ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΠΡΟΓΡΑΜΜΑ ΠΡΟΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΝΑΠΤΥΞΗ ΜΟΝΤΕΛΟΥ ΑΝΑΓΝΩΡΙΣΗΣ ΠΙΝΑΚΙΔΩΝ ΤΟΥ ΚΩΔΙΚΑ ΟΔΙΚΗΣ ΚΥΚΛΟΦΟΡΙΑΣ ΚΑΙ ΠΑΡΑΓΩΓΗΣ ΑΝΤΙΣΤΟΙΧΗΣ ΗΧΗΤΙΚΗΣ ΠΡΟΕΙΔΟΠΟΙΗΣΗΣ»

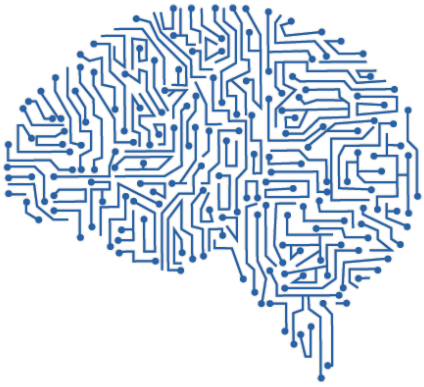
ΕΡΓΑΣΙΑ ΤΩΝ ΦΟΙΤΗΤΩΝ:

ΧΑΤΖΗΣΤΕΦΑΝΟΥ ΕΥΣΤΑΘΙΟΥ 164835

ΓΩΓΟΥ ΙΩΑΝΝΗ 517304

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΚΩΤΣΑΚΗΣ ΡΗΓΑΣ



ΑΥΓΟΥΣΤΟΣ 2023

INTERNATIONAL HELLENIC UNIVERSITY

UNDERGRADUATE PROGRAM

DEPARTMENT OF INFORMATION AND ELECTRONIC ENGINEERING

THESIS

“DEVELOPMENT OF A MODEL FOR THE RECOGNITION OF ROAD TRAFFIC SIGNS AND THE PRODUCTION OF A CORRESPONDING AUDIBLE WARNING”

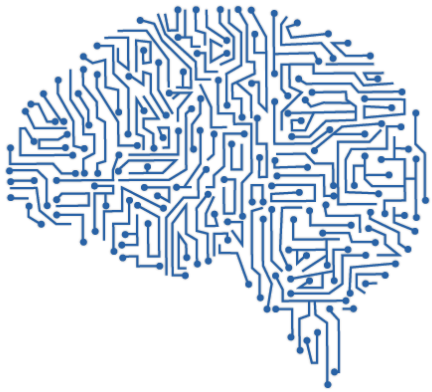
WORK OF THE STUDENTS:

CHATZISTEFANOU EFSTATHIOS 164835

GOGOS IOANNIS 517304

SUPERVISING PROFESSOR:

KOTSAKIS RIGAS



AUGUST 2023

Τίτλος Π.Ε.:

Ανάπτυξη μοντέλου αναγνώρισης πινακίδων του Κώδικα Οδικής Κυκλοφορίας και παραγωγής αντίστοιχης ηχητικής προειδοποίησης/ σήμανσης

Κωδικός Π.Ε.:

23211

Ονοματεπώνυμο φοιτητών:

Χατζηστεφάνου Ευστάθιος 165835

Γώγος Ιωάννης 517304

Ονοματεπώνυμο εισηγητή:

Ρήγας Κωτσάκης

Ημερομηνία ανάληψης Π.Ε.:

31-03-2023

Ημερομηνία περάτωσης Π.Ε.:

31-8-2023

Βεβαιώνουμε πως είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία εκπονήθηκε από τους προπτυχιακούς φοιτητές Χατζηστεφάνου Ευστάθιο και Γώγο Ιωάννη του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου Ελλάδος, κατά το ακαδημαϊκό έτος 2022-2023, υπό την επίβλεψη του καθηγητή Κωτσάκη Ρήγα.

Θα θέλαμε να εκφράσουμε την ευγνωμοσύνη μας στον καθηγητή μας για την ανάθεση του θέματος, για τις πολύτιμες γνώσεις που μοιράστηκε μαζί μας, την ανεκτίμητη βοήθειά του, ως επίσης το ενδιαφέρον του αλλά καθώς και το χρόνο που διέθεσε, ώστε να καταστεί δυνατή η από μέρους μας διεκπεραίωση της πτυχιακής εργασίας. Θα θέλαμε επίσης να ευχαριστήσουμε τους καθηγητές του τμήματος, που με το περιεχόμενο των μαθημάτων τους έδωσαν, αρκετές ιδέες και ώθησαν με το δικό τους τρόπο στην επιλογή και διεκπεραίωση της συγκεκριμένης εργασίας.

Τέλος, θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας για την συνεχή προσφορά και υποστήριξη κατά την διάρκεια των φοιτητικών μας χρόνων, χρόνων που υπήρξαν δυσκολίες, απογοητεύσεις αλλά ταυτόχρονα και επιτυχίες, χαρές και χαμόγελα.

Χατζηστεφάνου Ευστάθιος,

Γώγος Ιωάννης,

Θεσσαλονίκη, Αύγουστος 2023

ΠΕΡΙΛΗΨΗ

Οι πινακίδες κυκλοφορίας έχουν καθοριστικό ρόλο στη διαχείριση της κυκλοφορίας στο δρόμο, καθώς πειθαρχούν τους οδηγούς, αποτρέποντας έτσι τραυματισμούς, υλικές ζημιές αλλά ακόμα και θανάτους. Η διαχείριση των πινακίδων κυκλοφορίας με αυτόματη ανίχνευση και αναγνώριση αποτελεί σημαντικό μέρος κάθε ευφυούς συστήματος μεταφορών. Για αυτό το λόγο δημιουργήθηκε η ανάγκη για αναγνώριση πινακίδων κυκλοφορίας με τη χρήση μηχανικής μάθησης. Η χρήση τεχνικών μηχανικής μάθησης για την αυτόματη αναγνώριση πινακίδων κυκλοφορίας παρέχει πολλαπλά οφέλη και εφαρμογές σε διάφορους τομείς της καθημερινής ζωής. Βοηθά στην επίτευξη αυξημένης οδικής ασφάλειας, στην αυτόματη οδήγηση, στη βελτίωση του συστήματος στάθμευσης και στην επιβολή του οδικού κώδικα. Ωστόσο, παρά τα οφέλη και τη χρησιμότητα της αναγνώρισης πινακίδων κυκλοφορίας με τη χρήση μηχανικής μάθησης, υπάρχουν και προκλήσεις που πρέπει να αντιμετωπιστούν. Μεταξύ αυτών περιλαμβάνονται η ποικιλία των πινακίδων σε διάφορες περιοχές και χώρες, η αντίθεση και η σκίαση που μπορεί να επηρεάσουν την ακρίβεια της αναγνώρισης, καθώς και η προστασία της ιδιωτικότητας δεδομένων. Συνολικά, η αναγνώριση πινακίδων κυκλοφορίας με τη χρήση μηχανικής μάθησης αποτελεί ένα σημαντικό πεδίο έρευνας και ανάπτυξης, με ευρεία εφαρμογή και δυνατότητες βελτίωσης της οδικής ασφάλειας και της καθημερινής μας εμπειρίας στον τομέα της κυκλοφορίας.

Η παρούσα πτυχιακή εργασία με τίτλο *«Ανάπτυξη μοντέλου αναγνώρισης πινακίδων του Κώδικα Οδικής Κυκλοφορίας και παραγωγής αντίστοιχης ηχητικής προειδοποίησης/ σήμανσης»* αναφέρεται στον σχεδιασμό και την υλοποίηση ενός μοντέλου μηχανικής μάθησης το οποίο έχει αποδειχθεί αποτελεσματικό στην αναγνώριση πινακίδων κυκλοφορίας με βάση την εικόνα και στη συνέχεια παράγει αντίστοιχες ηχητικές προειδοποιήσεις ή σημάνσεις για τους οδηγούς. Για την ανάπτυξη του μοντέλου απαιτείται η συλλογή και επεξεργασία μεγάλου όγκου δεδομένων που περιέχουν πληροφορίες για διάφορες πινακίδες κυκλοφορίας. Το μοντέλο εκπαιδεύεται με βάση αυτά τα δεδομένα, τα οποία αποτελούνται από εικόνες πινακίδων κυκλοφορίας και τις αντίστοιχες ετικέτες τους. Αφού ολοκληρωθεί η εκπαίδευση, το μοντέλο μπορεί να χρησιμοποιηθεί για την αναγνώριση πινακίδων και την παραγωγή αντίστοιχης ηχητικής

προειδοποίησης, ώστε να ενημερώνει τους χρήστες για τις πινακίδες κυκλοφορίας που εντοπίζονται. Στη συνέχεια γίνεται αναλυτική παρουσίαση των διαφόρων τεχνολογιών που επιλέχτηκαν και χρησιμοποιήθηκαν, αλλά και των αποτελεσμάτων της εφαρμογής που προκύπτουν.

Λέξεις κλειδιά: αναγνώριση πινακίδων, Κώδικας Οδικής Κυκλοφορίας, μηχανική μάθηση, μοντέλο αναγνώρισης, ηχητική προειδοποίηση, σήμανση, επεξεργασία δεδομένων, εκπαίδευση, πληροφορίες, εικόνες, αποδοτικότητα, ανάπτυξη.

ABSTRACT

Traffic signs play a crucial role in managing road traffic as they discipline drivers, preventing injuries, material damages, and even fatalities. The management of traffic signs through automatic detection and recognition is an essential part of any intelligent transportation system. This is why there is a need for traffic sign recognition using machine learning. The use of machine learning techniques for automatic traffic sign recognition provides multiple benefits and applications in various aspects of everyday life. It helps achieve enhanced road safety, automated driving, improvement in parking systems, and enforcement of traffic regulations. However, despite the benefits and utility of traffic sign recognition using machine learning, there are challenges that need to be addressed. These include the diversity of signs in different regions and countries, contrast and shading that can affect recognition accuracy, as well as data privacy protection. Overall, traffic sign recognition using machine learning constitutes a significant field of research and development with wide-ranging applications and the potential to enhance road safety and our daily experience in the field of traffic.

The present thesis, entitled "Development of a Traffic Sign Recognition Model based on the Highway Code and generation of corresponding auditory warnings/signals," focuses on the design and implementation of a machine learning model that has proven effective in recognizing traffic signs based on images and subsequently generating corresponding auditory warnings or signals for drivers. The development of the model requires the collection and processing of a large volume of data containing information about various traffic signs. The model is trained using this data, which consists of traffic sign images and their corresponding labels. Once the training is completed, the model can be used for traffic sign recognition and generating corresponding auditory warnings to inform users about the detected traffic signs. Furthermore, a detailed presentation is provided on the selected and utilized technologies, as well as the results obtained from the application.

Keywords: traffic sign recognition, Highway Code, machine learning, recognition model, auditory warning, signaling, data processing, training, information, images, efficiency, development.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΕΥΣΕΩΝ

Π.Ε.	Πτυχιακή εργασία
Κ.Ο.Κ.	Κώδικας Οδικής Κυκλοφορίας
API	Application Programming Interface
GTSRB	German Traffic Sign Recognition Benchmark
ROI	Region Of Interest
OS	Operating System
RGB	Red Green Blue
CNN	Convolutional Neural Network
SVM	Support Vector Machines
ReLU	Rectified Linear Unit
HTTP	HyperText Transfer Protocol
HTML	Hyper Text Markup Language

Κεφάλαιο 1: Εισαγωγή

1.1 Εισαγωγή

Οι ταχύτατες εξελίξεις στην τεχνολογία υπολογιστών και τεχνικές μηχανικής μάθησης έχουν ανοίξει νέες δυνατότητες για τη βελτίωση της οδικής ασφάλειας και των συστημάτων διαχείρισης της κυκλοφορίας. Οι πινακίδες κυκλοφορίας διαδραματίζουν έναν κρίσιμο ρόλο στον καθοδήγηση των οδηγών και στην επιβολή των κανόνων κυκλοφορίας, αποτρέποντας τα ατυχήματα και εξασφαλίζοντας την ομαλή ροή της κυκλοφορίας.

1.2 Πρόβλημα

Ωστόσο, η αποτελεσματικότητα των πινακίδων κυκλοφορίας εξαρτάται σε μεγάλο βαθμό από την προσοχή και την ευαισθησία των οδηγών. Σε ορισμένα σενάρια, όπως η κακή ορατότητα ή η αποσπασμένη προσοχή του οδηγού, είναι πιθανό οι οδηγοί να μην αντιλαμβάνονται ή να ερμηνεύουν σωστά τις πληροφορίες που παρουσιάζονται από τις πινακίδες κυκλοφορίας, με αποτέλεσμα τον κίνδυνο στον δρόμο.

1.3 Αντιμετώπιση Προβλήματος

Για να αντιμετωπιστεί αυτό το πρόβλημα, η ανάπτυξη ενός αξιόπιστου συστήματος αναγνώρισης πινακίδων κυκλοφορίας που μπορεί να αναγνωρίζει και να ερμηνεύει αποτελεσματικά τις πινακίδες κυκλοφορίας, βασιζόμενο στον Κώδικα Οδικής Κυκλοφορίας, έχει κερδίσει σημαντική προσοχή.

1.4 Στόχοι εργασίας

Αυτή η πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μοντέλων αναγνώρισης πινακίδων κυκλοφορίας χρησιμοποιώντας τεχνικές μηχανικής μάθησης, με στόχο τη βελτίωση της οδικής ασφάλειας παρέχοντας αντίστοιχες ηχητικές προειδοποιήσεις ή σήμανση στους οδηγούς βάσει των αναγνωρισμένων πινακίδων κυκλοφορίας.

1.5 Λειτουργικότητα

Περιλαμβάνεται η συλλογή και επεξεργασία μεγάλου όγκου δεδομένων, από εικόνες πινακίδων κυκλοφορίας και τις αντίστοιχες ετικέτες τους. Μέσω της διαδικασίας εκπαίδευσης, το μοντέλο μαθαίνει να αναγνωρίζει και να ταξινομεί διάφορα είδη πινακίδων κυκλοφορίας σύμφωνα με τον Κώδικα Οδικής Κυκλοφορίας. Αφού ολοκληρωθεί η εκπαίδευση, το μοντέλο μπορεί να χρησιμοποιηθεί σε πραγματικό χρόνο για την αναγνώριση πινακίδων κυκλοφορίας από εικόνες και την παραγωγή κατάλληλων ηχητικών προειδοποιήσεων ή σημάνσεων, ενισχύοντας την ευαισθητοποίηση των οδηγών και μειώνοντας τον κίνδυνο ατυχημάτων.

1.6 Συνεισφορά

Η αντιμετώπιση των προκλήσεων της αναγνώρισης πινακίδων κυκλοφορίας και η ενσωμάτωση ηχητικής ανατροφοδότησης, στοχεύει στη βελτίωση της οδικής ασφάλειας και της διαχείρισης της κυκλοφορίας, παρέχοντας στους οδηγούς βελτιωμένες πληροφορίες και δεδομένα για το περιβάλλον. Αυτή η έρευνα συνεισφέρει στην πρόοδο της τεχνολογίας της οδικής ασφάλειας και συμβάλλει στη δημιουργία πιο έξυπνων και ασφαλών οδικών συστημάτων για το μέλλον.

Κεφάλαιο 2: Βιβλιογραφική Επισκόπηση

Στο κεφάλαιο αυτό γίνεται μια γενική βιβλιογραφική επισκόπηση της πτυχιακής εργασίας. Αρχικά παρουσιάζονται τα σήματα του κώδικα οδικής κυκλοφορίας και η σημαντικότητά τους στην καθημερινότητα του ανθρώπου. Επίσης γίνεται αναφορά στη χρήση μηχανικής μάθησης και στην μέθοδο αξιολόγησης του μοντέλου.

2.1 Κώδικας Οδικής Κυκλοφορίας

Ο Κώδικας Οδικής Κυκλοφορίας (Κ.Ο.Κ.) αποτελεί το πιο σημαντικό νομικό κείμενο σχετικά με την ασφαλή και εύρυθμη κυκλοφορία στους δρόμους. Η καλή γνώση και η τήρηση των διατάξεών του είναι απαραίτητη όχι μόνο για τους οδηγούς, αλλά και για όλους τους χρήστες των οδών, αφού η ασφάλεια επηρεάζεται από τη συμπεριφορά όλων μας. Με αυτή τη λογική ο Κ.Ο.Κ. δεν είναι απλά ένα νομικό κείμενο αλλά κανόνας ζωής, διότι η τήρηση των διατάξεών του επιβάλλεται επί της ουσίας για την προστασία της ίδιας της ανθρώπινης ύπαρξης.[1]

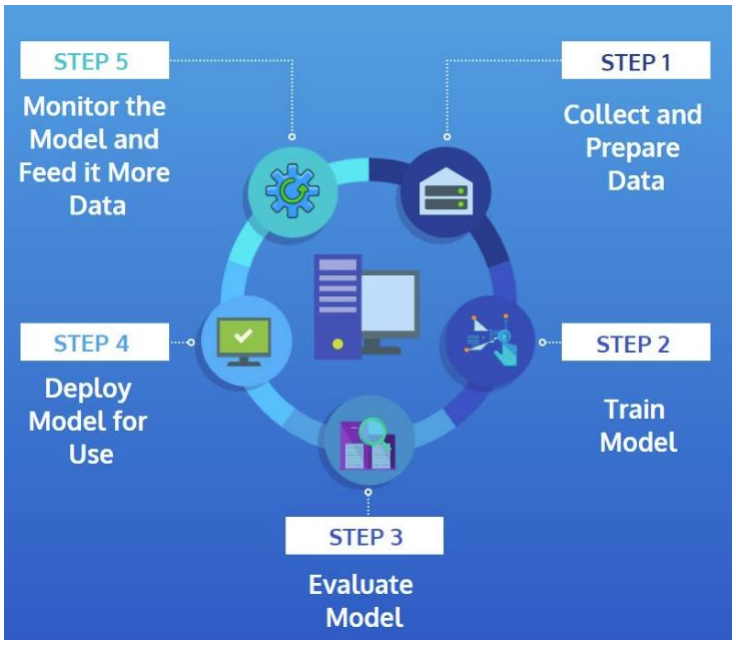
Εικόνα 2.1 Ρυθμιστικές Πινακίδες

Εικόνα 2.2: Πινακίδες Κινδύνου

2.2 Μηχανική Μάθηση

2.2.1 Τι είναι μηχανική μάθηση

Με τη συνεχή αύξηση του όγκου των δεδομένων σε ηλεκτρονική μορφή, η ανάγκη για αυτοματοποιημένες μεθόδους ανάλυσης δεδομένων συνεχίζει να αυξάνεται. Ο στόχος της μηχανικής μάθησης είναι να αναπτύξει μεθόδους που μπορούν αυτόματα να ανιχνεύουν μοτίβα στα δεδομένα και έπειτα να τα χρησιμοποιούν για την πρόβλεψη μελλοντικών δεδομένων ή άλλων αποτελεσμάτων.[2] Η μηχανική μάθηση σχετίζεται στενά με τα πεδία της στατιστικής και της εξόρυξης δεδομένων, αλλά διαφέρει ελαφρώς ως προς την έμφαση και τον ορολογία της. Τα συστήματα μηχανικής μάθησης μαθαίνουν αυτόματα προγράμματα από δεδομένα. Την τελευταία δεκαετία η χρήση της μηχανικής μάθησης έχει εξαπλωθεί με ραγδαίους ρυθμούς στην επιστήμη των υπολογιστών και όχι μόνο. Η μηχανική μάθηση χρησιμοποιείται στην αναζήτηση στον ιστό, στα φίλτρα ανεπιθύμητων μηνυμάτων, στα συστήματα σύστασης, στην τοποθέτηση διαφημίσεων, στην αξιολόγηση πιστοληπτικού κινδύνου, στον εντοπισμό απάτης, στις συναλλαγές με μετοχές, στο σχεδιασμό φαρμάκων και σε πολλές άλλες εφαρμογές[3] όπως φυσικά την αναγνώριση πινακίδων κυκλοφορίας.



Εικόνα 2.3: Αλγόριθμος Λειτουργίας Μηχανικής Μάθησης

2.2.2 Μηχανική μάθηση και έξυπνη οδήγηση

Ως άνθρωποι, η αναγνώριση αντικειμένων είναι μέρος της καθημερινής ζωής μας, βασιζόμενη στις αισθήσεις μας. Η αυτοματοποιημένη αναγνώριση αντικειμένων αποτελεί το μέλλον των αυτοκινήτων. Η μετάβαση από την ανθρώπινη αναγνώριση αντικειμένων στην αυτοματοποιημένη αναγνώριση αντικειμένων αποτελεί έναν τεράστιο βήμα. Κάθε χρόνο λαμβάνουν χώρα χιλιάδες οδικά ατυχήματα σε όλο τον κόσμο, που οφείλονται κυρίως σε ανθρώπινα λάθη. Για να μειωθεί αυτό, τα αυτοκίνητα μπορούν να γίνουν πλήρως αυτοματοποιημένα, χωρίς να απαιτείται καμία ανθρώπινη παρέμβαση. Τα αυτόνομα αυτοκίνητα φέρνουν επίσης πλεονεκτήματα όπως η αποδοτικότητα καυσίμων, η άνεση και η ευκολία, προκαλώντας έτσι έντονη έρευνα παγκοσμίως. Ένας καθοριστικός παράγοντας για την επίτευξη επιτυχίας σε αυτό τον τομέα είναι η δημιουργία καλύτερων αισθητήρων ανίχνευσης εμποδίων και σήμανσης, και η Τεχνητή Νοημοσύνη ανοίγει τον δρόμο για την ενσωμάτωσή τους. Η Τεχνητή Νοημοσύνη χρησιμοποιείται από έναν υπολογιστή με τον ίδιο τρόπο που οι άνθρωποι χρησιμοποιούν την νοημοσύνη τους. Αυτό είναι πολύ χρήσιμο για την ανίχνευση αντικειμένων/σήμανσης, τον αυτόματο έλεγχο ταχύτητας και την πλοήγηση.[4]

Εικόνα 2.4: Αναπαράσταση αναγνώρισης σήμανσης

Κεφάλαιο 3: Ανάλυση και Επεξήγηση του Θέματος

Στην παρούσα πτυχιακή εργασία γίνεται κατηγοριοποίηση Οδικών Σημάτων με την χρήση διάφορων τεχνολογιών μηχανικής μάθησης καθώς επίσης γίνεται και η σύγκριση των τεχνολογιών αυτών ως προς τον τρόπο λειτουργείας και την απόδοσή τους. Οι τεχνολογίες αυτές είναι τα συνελικτικά νευρωνικά δίκτυα - Convolutional neural network (CNN), η Γραμμική παλινδρόμηση (Linear regression), τα δέντρα αποφάσεων (Decision tree), η Λογιστική παλινδρόμηση (Logistic regression), Μηχανές διανυσμάτων υποστήριξης - Support Vector Machines (SVM) και ο αλγόριθμος Naive Bayes. Το σύνολο δεδομένων που έχει χρησιμοποιηθεί για την υλοποίηση της εργασίας είναι το German Traffic Sign Recognition Benchmark (GTSRB). Για κάθε κατηγορία πινακίδων δημιουργήθηκαν ηχητικά αποσπάσματα με χρήση online εργαλείου για μετατροπή κειμένου σε ομιλία. Παρακάτω γίνεται αναλυτικός σχολιασμός της κάθε τεχνολογίας με τη σειρά ξεχωριστά, παρουσιάζεται ο τρόπος δομής και υλοποίησης του εκάστου κώδικα καθώς επίσης γίνεται η σύγκριση τους με σκοπό την αξιολόγηση της απόδοσης του καθενός.

3.1: Ανάλυση της τεχνολογίας CNN

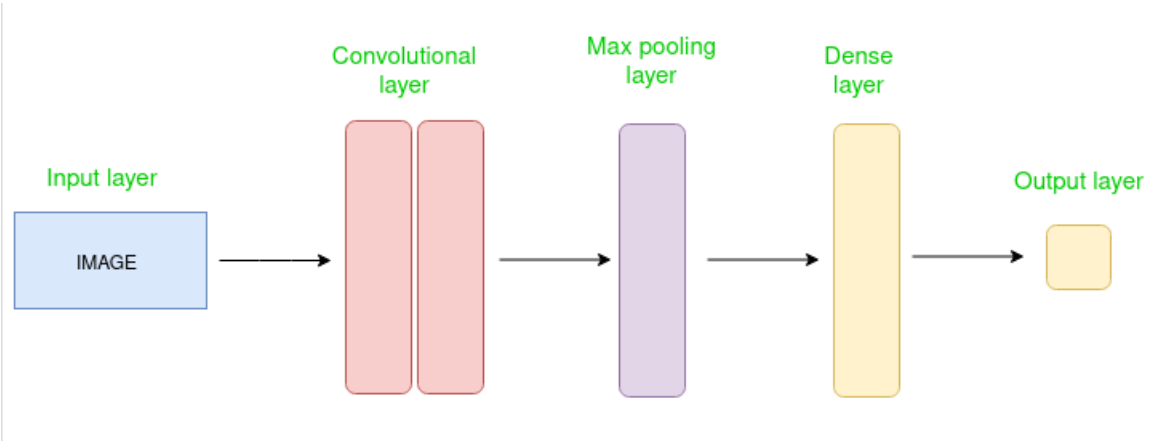
Το Συνελικτικό Νευρωνικό Δίκτυο είναι ένας αλγόριθμος βαθιάς μάθησης ειδικά σχεδιασμένος για την επεξεργασία εικόνων και βίντεο. Λαμβάνει εικόνες ως είσοδο, εξάγει και μαθαίνει τα χαρακτηριστικά της εικόνας και τις ταξινομεί με βάση τα χαρακτηριστικά που έμαθε.

Αυτός ο αλγόριθμος είναι εμπνευσμένος από τη λειτουργία ενός τμήματος του ανθρώπινου εγκεφάλου που είναι ο οπτικός φλοιός. Ο οπτικός φλοιός είναι ένα τμήμα του ανθρώπινου εγκεφάλου το οποίο είναι υπεύθυνο για την επεξεργασία των οπτικών πληροφοριών από τον εξωτερικό κόσμο. Διαθέτει διάφορα επίπεδα και κάθε επίπεδο έχει τη δική του λειτουργία, δηλαδή κάθε επίπεδο εξάγει κάποιες πληροφορίες από την εικόνα ή οποιοδήποτε οπτικό υλικό και τελικά όλες οι πληροφορίες που λαμβάνονται από κάθε επίπεδο συνδυάζονται και η εικόνα/οπτικό υλικό ερμηνεύεται ή ταξινομείται.

Ομοίως, το CNN διαθέτει διάφορα φίλτρα και κάθε φίλτρο εξάγει κάποιες πληροφορίες από την εικόνα, όπως ακμές, διάφορα είδη σχημάτων (κάθετα, οριζόντια, στρογγυλά), και στη συνέχεια όλα αυτά συνδυάζονται για την αναγνώριση της εικόνας.[5]

Τα CNN είναι συγκεκριμένα εμπνευσμένα από τον βιολογικό οπτικό φλοιό. Ο φλοιός διαθέτει μικρές περιοχές κυττάρων που είναι ευαίσθητες σε συγκεκριμένες περιοχές του οπτικού πεδίου. Αυτή η ιδέα επεκτάθηκε από ένα συναρπαστικό πείραμα που έκαναν οι Hubel και Wiesel το 1962. Σε αυτό το πείραμα, οι ερευνητές έδειξαν ότι ορισμένοι μεμονωμένοι νευρώνες στον εγκέφαλο ενεργοποιούνταν ή πυροδοτούνταν μόνο με την παρουσία ακμών συγκεκριμένου προσανατολισμού, όπως κάθετες ή οριζόντιες ακμές. Για παράδειγμα, ορισμένοι νευρώνες πυροδοτήθηκαν όταν εκτέθηκαν σε κάθετες πλευρές και ορισμένοι όταν τους εμφανίστηκε μια οριζόντια ακμή. Οι Hubel και Wiesel διαπίστωσαν ότι όλοι αυτοί οι νευρώνες ήταν καλά διατεταγμένοι σε μορφή στήλης και ότι όλοι μαζί ήταν σε θέση να παράγουν οπτική αντίληψη. Αυτή την ιδέα των εξειδικευμένων συστατικών μέσα σε ένα σύστημα που έχουν συγκεκριμένα καθήκοντα χρησιμοποιούν και οι μηχανές και την οποία μπορείτε επίσης να βρείτε πίσω στα CNN.[6]

Ένα νευρωνικό δίκτυο συνελικτικής μάθησης (CNN ή ConvNet) είναι μια αρχιτεκτονική δικτύου για βαθιά μάθηση που μαθαίνει απευθείας από τα δεδομένα. Τα CNN είναι ιδιαίτερα χρήσιμα για την εύρεση μοτίβων σε εικόνες για την αναγνώριση αντικειμένων, κλάσεων και κατηγοριών. Μπορούν επίσης να είναι αρκετά αποτελεσματικά για την ταξινόμηση δεδομένων ήχου, χρονοσειρών και σημάτων.



Εικόνα 3.1: Simple CNN architecture

Το συνεπτυγμένο νευρωνικό δίκτυο αποτελείται από πολλαπλά στρώματα όπως το στρώμα εισόδου, το συνεπτυγμένο στρώμα, το στρώμα συγκέντρωσης και τα πλήρως συνδεδεμένα στρώματα.[7]

Το στρώμα Convolutional εφαρμόζει φίλτρα στην εικόνα εισόδου για την εξαγωγή χαρακτηριστικών, το στρώμα Pooling μειώνει τα δείγματα της εικόνας για να μειώσει τον υπολογισμό και το πλήρως συνδεδεμένο στρώμα κάνει την τελική πρόβλεψη. Το δίκτυο μαθαίνει τα βέλτιστα φίλτρα μέσω οπισθοδιάδοσης και καθόδου κλίσης.[8]

Κεφάλαιο 3.1.1 Keras & Tensorflow

Για την κατασκευή του μοντέλου με την τεχνολογία CNN χρησιμοποιούνται βιβλιοθήκες Keras και Tensorflow.

Keras

Το Keras είναι ένα API βαθιάς μάθησης γραμμένο σε Python, το οποίο εκτελείται πάνω στην πλατφόρμα μηχανικής μάθησης TensorFlow. Αναπτύχθηκε με επίκεντρο τη δυνατότητα γρήγορου πειραματισμού. Η δυνατότητα μετάβασης από την ιδέα στο αποτέλεσμα το συντομότερο δυνατό είναι το κλειδί για την πραγματοποίηση καλής έρευνας.[9]

TensorFlow

Το TensorFlow είναι μια δημοφιλής βιβλιοθήκη ανοικτού κώδικα που κυκλοφόρησε το 2015 από την ομάδα Google Brain για την κατασκευή μοντέλων μηχανικής μάθησης και βαθιάς μάθησης. Βασίζεται στη γλώσσα προγραμματισμού Python και εκτελεί αριθμητικούς υπολογισμούς χρησιμοποιώντας γραφήματα ροής δεδομένων για τη δημιουργία μοντέλων.[10]

3.2 Υλοποίηση του μοντέλου

Ο κώδικας όπως προαναφέραμε αφορά την εκπαίδευση ενός νευρωνικού δικτύου συνελικτικών νευρώνων (CNN) για την κατηγοριοποίηση εικόνων χρησιμοποιώντας τις βιβλιοθήκες TensorFlow και Keras. Ο κώδικας περιλαμβάνει διάφορα βήματα, όπως προεπεξεργασία δεδομένων, κατασκευή της αρχιτεκτονικής του μοντέλου, σύνθεση του μοντέλου και εκπαίδευσή του χρησιμοποιώντας έναν data generator για εικόνες.

3.2.1 Εισαγωγή βιβλιοθηκών

Οι βιβλιοθήκες που χρησιμοποιούνται στον κώδικα παρέχουν τις απαραίτητες λειτουργίες για την επεξεργασία, ανάλυση και εκπαίδευση του μοντέλου μηχανικής μάθησης. Ας δούμε πιο αναλυτικά τις βιβλιοθήκες που εισάγονται:

Tensorflow: Αυτή είναι η βασική βιβλιοθήκη για μηχανική μάθηση και βαθιά μάθηση. Χρησιμοποιείται για τον ορισμό του μοντέλου, την εκπαίδευσή του και την αξιολόγησή του. Η βιβλιοθήκη NumPy παρέχει υποστήριξη για πίνακες και πράξεις πινάκων. Χρησιμοποιείται για την αποθήκευση και την επεξεργασία των δεδομένων όπως οι εικόνες και οι ετικέτες. Η βιβλιοθήκη os παρέχει λειτουργίες για τη διαχείριση των λειτουργικών συστημάτων, όπως τη διαχείριση διαδρομών αρχείων και φακέλων. Το OpenCV (CV2) είναι μια βιβλιοθήκη για επεξεργασία εικόνων και υπολογιστική όραση. Χρησιμοποιείται για να διαβάσει εικόνες από τον δίσκο και να τις μετατρέψει σε πίνακες. Το Keras είναι ένα υψηλού επιπέδου πλαίσιο εργασίας για τη δημιουργία και την εκπαίδευση νευρωνικών δικτύων. Χρησιμοποιείται για την κατασκευή, την εκπαίδευση και την αξιολόγηση του μοντέλου. Η βιβλιοθήκη Python Imaging Library (PIL) παρέχει λειτουργίες για την επεξεργασία εικόνων. Χρησιμοποιείται για να μετατρέψει εικόνες σε αντικείμενα Image και να τις αλλάξει μέγεθος. Η υπο-βιβλιοθήκη της scikit-learn χρησιμοποιείται για τον διαχωρισμό των δεδομένων σε σύνολο εκπαίδευσης και επικύρωσης. Η βιβλιοθήκη Matplotlib χρησιμοποιείται για τη δημιουργία γραφικών.

```python
import tensorflow as tf
print(tf.__version__)
import numpy as np
import os
import cv2
from tensorflow import keras
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator

np.random.seed(42)

from matplotlib import style
```

Εικόνα 3.2: Εισαγωγή βιβλιοθηκών CNN

3.2.2 Διαδρομές Φακέλων Δεδομένων

Ορίζονται οι διαδρομές προς τους φακέλους με τα δεδομένα εκπαίδευσης και δοκιμής χρησιμοποιώντας τις μεταβλητές train_path και test_path.

```python
## ορισμός διαδρομής των δεδομένων
data_dir = r"C:\Users\stathis\Desktop\archive"
train_path = r"C:\Users\stathis\Desktop\archive\Train"
test_path = r"C:\Users\stathis\Desktop\archive\Test"
```

Εικόνα 3.3: Διαδρομές Φακέλων

3.2.3 Διαστάσεις Εικόνων

Στο κομμάτι αυτό του κώδικα γίνεται ορισμός των διαστάσεων των εικόνων που θα χρησιμοποιηθούν κατά τη διάρκεια της επεξεργασίας και της εκπαίδευσης του μοντέλου. Αυτό είναι σημαντικό για την ορθή είσοδο των εικόνων στο μοντέλο. Συγκεκριμένα:

IMG_HEIGHT: Πρόκειται για το ύψος των εικόνων, δηλαδή τον αριθμό των pixels στον κατακόρυφο άξονα της εικόνας. Αυτή η παράμετρος ορίζει πόσες γραμμές από pixels υπάρχουν σε μια εικόνα.

IMG_WIDTH: Πρόκειται για το πλάτος των εικόνων, δηλαδή τον αριθμό των pixels στον οριζόντιο άξονα της εικόνας. Αυτή η παράμετρος ορίζει πόσες στήλες από pixels υπάρχουν σε μια εικόνα.

channels: Αναφέρεται στον αριθμό των χρωματικών καναλιών της εικόνας. Στις περισσότερες περιπτώσεις, οι εικόνες έχουν τρία χρωματικά κανάλια: ένα για το κόκκινο (Red), ένα για το πράσινο (Green) και ένα για το μπλε (Blue), συνθέτοντας την έγχρωμη εικόνα. Αυτό το στοιχείο συνήθως αναπαρίσταται με τον αριθμό 3, δείχνοντας τα τρία βασικά χρωματικά κανάλια.

```
23  ## Αλλαγή μεγέθους εικόνων σε 30x30x3
24  IMG_HEIGHT = 30
25  IMG_WIDTH = 30
26  channels = 3
```

Εικόνα 3.4: Καθορισμός μεγέθους εικόνων

3.2.4 Αριθμός των Κατηγοριών

Αναφέρεται στον τρόπο με τον οποίο υπολογίζεται ο αριθμός των διαφορετικών κατηγοριών ή κλάσεων που πρέπει να αναγνωρίσει το μοντέλο μας κατά τη διάρκεια της εκπαίδευσης.

Κατηγορίες: Οι κατηγορίες, επίσης γνωστές ως κλάσεις, αντιπροσωπεύουν τα διαφορετικά σύνολα δεδομένων που θέλουμε το μοντέλο να αναγνωρίσει. Κάθε κατηγορία αντιστοιχεί σε ένα είδος ή μοτίβο εικόνας που θα προσπαθήσουμε να εντοπίσουμε. Για παράδειγμα, σε ένα πρόβλημα αναγνώρισης οδικών πινακίδων, οι κατηγορίες θα μπορούσαν να είναι οι διάφορες τύποι πινακίδων (π.χ. "Μέγιστη Ταχύτητα 30km/h", "Στοπ", κλπ).

Υποφάκελοι: Σε αυτή την περίπτωση, η παράγραφος αναφέρεται στους υποφακέλους που βρίσκονται μέσα στον φάκελο εκπαίδευσης. Ο κάθε υποφάκελος αντιστοιχεί σε μία κατηγορία ή κλάση που πρέπει το μοντέλο να αναγνωρίσει. Δηλαδή, κάθε υποφάκελος περιέχει τις εικόνες που ανήκουν σε μία συγκεκριμένη κατηγορία.

Καθώς το μοντέλο μας θα εκπαιδευτεί να αναγνωρίζει τις εικόνες από κάθε υποφάκελο, ο αριθμός των διαφορετικών υποφακέλων (και, επομένως, κατηγοριών) που υπάρχουν στον φάκελο εκπαίδευσης θα καθορίσει τον αριθμό των κατηγοριών που πρέπει να ορίσουμε

στο μοντέλο μας. Αυτό είναι σημαντικό για την ορθή εκπαίδευση και αξιολόγηση του μοντέλου μας, καθώς πρέπει να έχουμε τη σωστή αντιστοίχιση μεταξύ των εικόνων και των αντίστοιχων κατηγοριών.

```
29  ## Ανίχνευση κάθε κλάσης
30  NUM_CATEGORIES = len(os.listdir(train_path))
31  NUM_CATEGORIES
```

Εικόνα 3.5: Ανίχνευση κλάσεων

3.2.5 Ετικέτες Κλάσεων

Οι ετικέτες κλάσεων αναφέρονται στις ετικέτες που αναθέτουμε στα δεδομένα μας για να δηλώσουμε σε ποια κατηγορία ανήκουν. Αποτελούν τον τρόπο με τον οποίο επισημαίνουμε ποια κλάση αντιπροσωπεύει κάθε εικόνα. Στον κώδικα, οι ετικέτες κλάσεων περιγράφονται με ένα λεξικό που αντιστοιχεί τον αριθμό της κλάσης στην αντίστοιχη περιγραφή της. Οι ετικέτες κλάσεων περιγράφονται με τη χρήση ενός λεξικού (classes). Το λεξικό περιέχει ζευγάρια "κλειδί-τιμή", όπου το "κλειδί" είναι ο αριθμός της κλάσης και η "τιμή" είναι η περιγραφή της κλάσης. Για κάθε κλάση, υπάρχει ένας αριθμός και μία αντίστοιχη περιγραφή που περιγράφει την κατηγορία της εικόνας. Για παράδειγμα, η κλάση 0 αντιπροσωπεύει την "Μέγιστη ταχύτητα(20km/h)", η κλάση 1 αντιπροσωπεύει την "Μέγιστη ταχύτητα(30km/h)", και ούτω καθεξής. Ο λόγος για τον οποίο οι ετικέτες κλάσεων είναι σημαντικές είναι γιατί επιτρέπουν στο μοντέλο να κατανοήσει ποιες κατηγορίες αναπαριστά κάθε εικόνα κατά την εκπαίδευση και τον έλεγχο. Επίσης, οι ετικέτες κλάσεων είναι απαραίτητες για την αξιολόγηση της απόδοσης του μοντέλου, καθώς μας επιτρέπουν να συγκρίνουμε τις προβλέψεις του μοντέλου με τις πραγματικές κλάσεις των εικόνων.

```
33  # Δημιουργία ετικετών
34  classes = {0: 'Μέγιστη ταχύτητα(20km/h)',
35             1: 'Μέγιστη ταχύτητα(30km/h)',
36             2: 'Μέγιστη ταχύτητα(50km/h)',
37             3: 'Μέγιστη ταχύτητα(60km/h)',
38             4: 'Μέγιστη ταχύτητα(70km/h)',
39             5: 'Μέγιστη ταχύτητα(80km/h)',
40             6: 'Τέλος ορίου ταχύτητας που έχει επιβληθεί με απαγορευτική πινακίδα (80km/h)',
41             7: 'Μέγιστη ταχύτητα(100km/h)',
42             8: 'Μέγιστη ταχύτητα(120km/h)',
43             9: 'Απαγορεύεται το προσπέρασμα των μηχανοκινήτων οχημάτων, πλήν των δίτροχων μοτοσυκλετών χωρίς κάνιστρο',
44             10: 'Απαγορεύεται στους οδηγούς φορτηγών αυτοκινήτων μεγίστου επιτρεπόμενου βάρους που υπερβαίνει τους 3,5 τόννους να προσπερνούν άλλα οχήματα',
45             11: 'Διασταύρωση με οδό, πάνω στην οποία αυτοί που κινούνται ωφείλουν να παραχωρήσουν προτεραιότητα',
46             12: 'Οδός προτεραιότητας',
```

Εικόνα 3.6: Δημιουργία Ετικετών

3.2.6 Συλλογή Δεδομένων Εκπαίδευσης

Η "Συλλογή Δεδομένων Εκπαίδευσης" αναφέρεται σε ένα σημαντικό βήμα κατά τη διαδικασία επεξεργασίας και προετοιμασίας των δεδομένων πριν αρχίσει η εκπαίδευση του μοντέλου, δηλαδή στη διαδικασία συλλογής, επεξεργασίας και διαμόρφωσης των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. Συγκεκριμένα, δημιουργούνται δύο κενές λίστες με τα ονόματα image_data και image_labels. Αυτές οι λίστες χρησιμοποιούνται για την αποθήκευση των εικόνων και των αντίστοιχων ετικετών κλάσεων. Χρησιμοποιείται ένας βρόχος για κάθε κατηγορία (φάκελο) εικόνων. Για κάθε φάκελο, διαβάζονται τα ονόματα των αρχείων εικόνων και γίνεται επεξεργασία τους. Κάθε εικόνα ανοίγει χρησιμοποιώντας τη βιβλιοθήκη OpenCV (CV2). Στη συνέχεια, μετατρέπεται σε μορφή PIL και αλλάζει το μέγεθός της σε 30x30 πίξελ, όπως ορίζεται από τις σταθερές IMG_HEIGHT και IMG_WIDTH. Η μετατροπή της εικόνας σε πίνακα NumPy γίνεται με τη χρήση της np.array(). Ο πίνακας που αντιπροσωπεύει την εικόνα προστίθεται στη λίστα image_data, ενώ η ετικέτα της κατηγορίας προστίθεται στη λίστα image_labels. Οι πίνακες με τις εικόνες και τις ετικέτες αποτελούν τα δεδομένα εκπαίδευσης που το μοντέλο θα "δει" κατά τη διάρκεια της εκπαίδευσης για την εκμάθηση των συσχετίσεων μεταξύ των χαρακτηριστικών και των ετικετών.

```
79  ## Συλλογή των δεδομένων εκπαίδευσης
80  image_data = []
81  image_labels = []
82
83  for i in range(NUM_CATEGORIES):
84      path = data_dir + '/Train/' + str(i)
85      images = os.listdir(path)
86
87      for img in images:
88          try:
89              image = cv2.imread(path + '/' + img)
90              image_fromarray = Image.fromarray(image, mode: 'RGB')
91              resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
92              image_data.append(np.array(resize_image))
93              image_labels.append(i)
94          except:
95              print("Error in " + img)
```

Εικόνα 3.7: Συλλογή δεδομένων εκπαίδευσης

3.2.7 Ανακάτεμα Δεδομένων

Το "Ανακάτεμα Δεδομένων" αναφέρεται σε ένα βήμα που εφαρμόζεται μετά τη συλλογή και την προετοιμασία των δεδομένων εκπαίδευσης. Αυτό το βήμα έχει ως στόχο το ανακάτεμα των δεδομένων πριν αρχίσει η εκπαίδευση του μοντέλου. Είναι σημαντικό για να διασφαλιστεί ότι τα δεδομένα παρουσιάζονται στο μοντέλο με τυχαία σειρά κατά τη διάρκεια της εκπαίδευσης και πραγματοποιείται με τα εξής βήματα: Αρχικά δημιουργείται ένας πίνακας shuffle_indexes που περιέχει τα δείγματα (εικόνες) ταξινομημένα με βάση τον αριθμό της θέσης τους. Εφαρμόζεται μια τυχαία αναδιάταξη των δειγμάτων με τη χρήση της συνάρτησης np.random.shuffle(). Αυτό δημιουργεί μια τυχαία σειρά για τον πίνακα shuffle_indexes. Οι πίνακες image_data και image_labels αναδιατάσσονται βάσει της τυχαίας σειράς που έχει δημιουργηθεί στον πίνακα shuffle_indexes. Αυτό σημαίνει ότι τα δείγματα ανακατεύονται και έχουν μια τυχαία σειρά κατά τη διάρκεια της εκπαίδευσης. Αυτό είναι σημαντικό για να αποτρέψει το μοντέλο από το να μάθει πρότυπα που συνδέονται με τη σειρά των δειγμάτων. Αν τα δείγματα παρουσιάζονταν στο μοντέλο με σταθερή σειρά, θα μπορούσε να επηρεαστεί η ικανότητα του μοντέλου να γενικεύει και να αναγνωρίζει τα μοτίβα ανεξάρτητα από τη σειρά των δειγμάτων.

```
103  ## Ανακάτεμα των δεδομένων εκπαίδευσης
104  shuffle_indexes = np.arange(image_data.shape[0])
105  np.random.shuffle(shuffle_indexes)
106  image_data = image_data[shuffle_indexes]
107  image_labels = image_labels[shuffle_indexes]
```

Εικόνα 3.8: Ανακάτεμα δεδομένων εκπαίδευσης

3.2.8 Διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και επικύρωσης

Ο "Διαχωρισμός Εκπαίδευσης-Επικύρωσης" είναι ένα βήμα στη διαδικασία της εκπαίδευσης του μοντέλου, το οποίο αφορά τον χωρισμό του διαθέσιμου συνόλου δεδομένων σε δύο υποσύνολα: το σύνολο εκπαίδευσης και το σύνολο επικύρωσης.

Το σύνολο εκπαίδευσης χρησιμοποιείται για να εκπαιδεύσει το μοντέλο. Κατά τη διάρκεια της εκπαίδευσης το μοντέλο προσαρμόζει τις παραμέτρους του, ώστε να μπορεί

να αναγνωρίζει τα πρότυπα και τις σχέσεις στα δεδομένα. Καθώς το μοντέλο εκπαιδεύεται, χρησιμοποιεί τα δείγματα από το σύνολο εκπαίδευσης για να βελτιστοποιήσει τις εσωτερικές του παραμέτρους, ώστε να μπορεί να παράγει ακριβές προβλέψεις σε νέα δεδομένα.

Το σύνολο επικύρωσης χρησιμοποιείται για να εκτιμήσουμε την απόδοση του μοντέλου κατά τη διάρκεια της εκπαίδευσης. Κατά τη διάρκεια κάθε εποχής (ένα πέρασμα από όλα τα δεδομένα εκπαίδευσης) της εκπαίδευσης, το μοντέλο αξιολογείται στο σύνολο επικύρωσης και υπολογίζονται μετρικές απόδοσης, όπως η ακρίβεια και το σφάλμα. Αυτό βοηθά στην παρακολούθηση της εξέλιξης της απόδοσης του μοντέλου κατά τη διάρκεια της εκπαίδευσης και μπορεί να βοηθήσει στην αποφυγή υπερεκπαίδευσης.

Ο διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και σύνολο επικύρωσης είναι σημαντικός για να αξιολογήσουμε τη γενικότητα του μοντέλου σε νέα, μη γνωστά δεδομένα. Αυτός ο διαχωρισμός μας επιτρέπει να ελέγξουμε αν το μοντέλο μας έχει μάθει να γενικεύει και να κάνει καλές προβλέψεις σε ανεξάρτητα δείγματα, πέρα από αυτά που χρησιμοποιούνται για την εκπαίδευσή του.

```
109  ## Διαχωρισμός των δεδομένων σε σύνολο εκπαίδευσης και επικύρωσης
110  X_train, X_val, y_train, y_val = train_test_split( *arrays: image_data, image_labels, test_size=0.3, random_state=42,
111                                                      shuffle=True)
112
113  X_train = X_train / 255
114  X_val = X_val / 255
115
116  print("X_train.shape", X_train.shape)
117  print("X_valid.shape", X_val.shape)
118  print("y_train.shape", y_train.shape)
119  print("y_valid.shape", y_val.shape)
```

Εικόνα 3.9: Σύνολο εκπαίδευσης και επικύρωσης

3.2.9 Κωδικοποίηση των ετικετών

Η "Κωδικοποίηση των ετικετών" αναφέρεται στη διαδικασία μετατροπής των κατηγοριών ετικετών ή κλάσεων σε μορφή που μπορεί να χρησιμοποιηθεί από μοντέλα μηχανικής μάθησης. Αυτό είναι σημαντικό γιατί πολλά μοντέλα μηχανικής μάθησης απαιτούν τις ετικέτες να είναι σε μορφή αριθμητικών τιμών.

Στον κώδικα, η κωδικοποίηση των ετικετών γίνεται χρησιμοποιώντας τη βιβλιοθήκη keras.utils της TensorFlow. Συγκεκριμένα, χρησιμοποιείται η συνάρτηση to_categorical(). Η διαδικασία αυτή μετατρέπει τις ετικέτες κλάσεων από τη μορφή τους (π.χ. ονόματα κλάσεων) σε μια δυαδική μορφή όπου κάθε κλάση αναπαρίσταται από ένα διάνυσμα με μηδενικά και άσσους.

Αυτή η κωδικοποίηση είναι σημαντική όταν το μοντέλο χρησιμοποιείται για την εκπαίδευση, καθώς επιτρέπει στο μοντέλο να αντιληφθεί την δομή των κατηγοριών και τις σχέσεις μεταξύ τους. Επιπλέον, αυτή η κωδικοποίηση διασφαλίζει ότι το μοντέλο μπορεί να παράγει προβλέψεις σε μορφή αριθμητικών τιμών, το οποίο είναι απαραίτητο για τη σύγκριση με τις πραγματικές ετικέτες κατά τη διάρκεια της αξιολόγησης της απόδοσης του μοντέλου.

```
121  ## Κωδικοποίηση των ετικετών
122  y_train = keras.utils.to_categorical(y_train, NUM_CATEGORIES)
123  y_val = keras.utils.to_categorical(y_val, NUM_CATEGORIES)
124
125  print(y_train.shape)
126  print(y_val.shape)
```

Εικόνα 3.10: Κωδικοποίηση ετικετών

3.2.10 Κατασκευή του μοντέλου

Η "Κατασκευή του μοντέλου" αναφέρεται στη δημιουργία της αρχιτεκτονικής του μοντέλου μηχανικής μάθησης, που θα χρησιμοποιηθεί για την επίλυση ενός συγκεκριμένου προβλήματος. Αυτή η αρχιτεκτονική καθορίζει πώς θα είναι οργανωμένα τα επίπεδα του μοντέλου, ποια είδη επιπέδων θα περιλαμβάνονται και πώς θα συνδέονται μεταξύ τους. Η κατασκευή του μοντέλου γίνεται με τη χρήση της βιβλιοθήκης TensorFlow και της υποβιβλιοθήκης Keras. Το μοντέλο που δημιουργείται είναι ένα συνελικτικό νευρωνικό δίκτυο (Convolutional Neural Network - CNN), το οποίο είναι εξαιρετικά αποτελεσματικό για την αναγνώριση εικόνων.

Δημιουργία του Μοντέλου: Το μοντέλο δημιουργείται ως ένα συνεχόμενο οριζόντιο stack των συνεκτικών στρωμάτων του νευρωνικού δικτύου. Ο σκοπός είναι να δημιουργηθεί μια σειρά από στρώματα που θα αντιπροσωπεύουν τη δομή του νευρωνικού δικτύου.

Συνελικτικά Επίπεδα: Ο κώδικας ξεκινά με την προσθήκη δύο συνελικτικών επιπέδων. Κάθε συνελικτικό επίπεδο (Conv2D) περιλαμβάνει μια ομάδα φίλτρων που αναγνωρίζουν διάφορα χαρακτηριστικά στις εικόνες. Κάθε φίλτρο εφαρμόζεται πάνω στις εικόνες με μια συγκεκριμένη παράθεση (stride) και δίνει τον χάρτη χαρακτηριστικών της εικόνας. Η συνάρτηση ενεργοποίησης "relu" χρησιμοποιείται για την ενεργοποίηση των νευρώνων στο συνελικτικό επίπεδο.

Επίπεδα Υποδειγματοληψίας: Στη συνέχεια, προστίθενται δύο επίπεδα υποδειγματοληψίας (MaxPool2D), τα οποία χρησιμοποιούνται για τη μείωση των διαστάσεων των χαρακτηριστικών με την επιλογή του μέγιστου τιμών σε ένα παράθυρο των διαστάσεων (2,2).

Επίπεδα Κανονικοποίησης Δέσμης: Τα επίπεδα κανονικοποίησης δέσμης (BatchNormalization) προστίθενται για τη βελτίωση της σταθερότητας και της ταχύτητας της εκπαίδευσης, καθώς και για τη μείωση της υπερεκπαίδευσης.

Επίπεδο Πλήρωσης: Ακολουθεί το επίπεδο πλήρωσης (Flatten), το οποίο μετατρέπει τις πολυδιάστατες παραστάσεις των χαρακτηριστικών σε ένα επίπεδο πίνακα, έτοιμο για την εισαγωγή σε ένα πλήρες συνδεδεμένο επίπεδο.

Επίπεδα Εξασθένισης (Dropout): Χρησιμοποιείται ένα επίπεδο εξασθένισης (Dropout) για να απενεργοποιήσει τυχαία ένα ποσοστό των νευρώνων κατά την εκπαίδευση. Αυτό βοηθά στη μείωση της υπερεκπαίδευσης και στη βελτίωση της γενίκευσης του μοντέλου.

Επίπεδο Εξόδου: Το τελευταίο πλήρες συνδεδεμένο επίπεδο έχει 43 νευρώνες, αντιπροσωπεύοντας τις 43 διαφορετικές κατηγορίες των οδικών σημάτων που πρέπει να αναγνωριστούν. Η συνάρτηση ενεργοποίησης "softmax" χρησιμοποιείται εδώ για να εξάγει μια πιθανοτική κατανομή πάνω στις κατηγορίες, δίνοντας μια πρόβλεψη για κάθε κατηγορία.

Βελτιστοποίηση και Συντελεστές: Ο κώδικας ορίζει τον βελτιστοποιητή Adam με έναν αρχικό ρυθμό εκπαίδευσης (learning rate) του 0.001 και ένα συντελεστή απόσβεσης (decay) που μειώνει το ρυθμό εκπαίδευσης κατά το τέλος των εποχών για τη βελτίωση της σύγκλισης. Επίσης, ορίζεται η συνάρτηση απώλειας (categorical_crossentropy) για την εκπαίδευση του μοντέλου και οι μετρικές αξιολόγησης ακρίβειας (accuracy).

```
## Κατασκευή του μοντέλου
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu',
                        input_shape=(IMG_HEIGHT, IMG_WIDTH, channels)),
    keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense( units: 512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense( units: 43, activation='softmax')
])

from tensorflow.keras.optimizers import legacy as legacy_optimizers

lr = 0.001
epochs = 1
decay = lr / (epochs * 0.5)

opt = legacy_optimizers.Adam(lr=lr, decay=decay)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

```

Εικόνα 3.11: Κατασκευή Μοντέλου

3.2.11 Συμπλήρωση των δεδομένων και εκπαίδευση

Ο κώδικας αφορά την εκπαίδευση του μοντέλου με επαυξημένα δεδομένα. Τα επαυξημένα δεδομένα είναι μια τεχνική που χρησιμοποιείται για τη βελτίωση της απόδοσης του μοντέλου εκπαίδευσης, εισάγοντας μικρές μεταστολές και μετατροπές στις εικόνες εκπαίδευσης πριν τις δώσει στο μοντέλο για εκπαίδευση. Αυτό βοηθά στη βελτίωση της γενίκευσης του μοντέλου και την αντιμετώπιση της υπερεκπαίδευσης. Ας δούμε τι κάνει κάθε παράμετρος του ImageDataGenerator:

rotation_range: Καθορίζει τη γωνία περιστροφής των εικόνων. Σε αυτήν την περίπτωση, οι εικόνες μπορεί να περιστραφούν μέχρι 10 μοίρες.

zoom_range: Ορίζει την περιοχή μεταβολής του μεγέθους των εικόνων. Οι εικόνες μπορούν να μεγεθυνθούν ή να σμικρυνθούν κατά μέγιστο 15%.

width_shift_range και height_shift_range: Ορίζουν το εύρος μετατόπισης των εικόνων σε οριζόντια και κατακόρυφη κατεύθυνση αντίστοιχα. Οι εικόνες μπορούν να μετακινηθούν μεταξύ -10% και +10% του πλάτους και του ύψους τους.

shear_range: Καθορίζει τη γωνία κάμψης των εικόνων. Οι εικόνες μπορούν να αλλοιωθούν κάθετα κατά μέγιστο 15%.

horizontal_flip και vertical_flip: Καθορίζουν εάν οι εικόνες θα αναστρέφονται οριζόντια ή κατακόρυφα.

fill_mode: Καθορίζει πώς θα γεμίζονται οι κενοί χώροι που προκύπτουν μετά την μετατόπιση ή αλλαγή μεγέθους των εικόνων. Στην περίπτωση "nearest", οι κενοί χώροι γεμίζουν με την πλησιέστερη τιμή.

Έπειτα, ο κώδικας χρησιμοποιεί τον aug.flow για να δημιουργήσει μια διαδοχή των επαυξημένων εικόνων από το σετ εκπαίδευσης και τις σχετικές ετικέτες. Αυτός ο διαχειριστής ροής είναι σημαντικός γιατί παρέχει επαυξημένα δεδομένα κατά τη διάρκεια της εκπαίδευσης, ώστε το μοντέλο να έχει ποικιλία και να μην υπερεκπαιδεύεται στα αρχικά δεδομένα.

Τέλος, το μοντέλο εκπαιδεύεται με τη χρήση του fit με τα επαυξημένα δεδομένα (aug.flow) για το σετ εκπαίδευσης και τις αντίστοιχες ετικέτες. Οι εποχές και τα δεδομένα της εκπαίδευσης έχουν οριστεί προηγουμένως. Επιπλέον, η εκπαίδευση γίνεται με τη χρήση του βελτιστοποιητή που έχει δημιουργηθεί, τους ορισμένους

χρησιμοποιούμενους μετρικές (σε αυτή την περίπτωση την ακρίβεια) και τον καθορισμένο αριθμό εποχών. Τέλος, το μοντέλο αποθηκεύεται σε ένα αρχείο .h5 με ένα όνομα που επιθυμούμε.

```
159 ## Συμπλήρωση των δεδομένων και εκπαίδευση
160 aug = ImageDataGenerator(
161     rotation_range=10,
162     zoom_range=0.15,
163     width_shift_range=0.1,
164     height_shift_range=0.1,
165     shear_range=0.15,
166     horizontal_flip=False,
167     vertical_flip=False,
168     fill_mode="nearest")
169
170 history = model.fit(aug.flow(X_train, y_train, batch_size=32), epochs=epochs, validation_data=(X_val, y_val))
171 model.save("testing1.h5")
172
```

Εικόνα 3.12: Εκπαίδευση

3.3 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης CNN

Ο κώδικας δέχεται ως είσοδο μια εικόνα οδικής σήμανσης, στη συνέχεια γίνεται αναγνώριση με βάση το προεκπαιδευμένο μοντέλο CNN που κατασκευάσαμε και αναλύσαμε στο κεφάλαιο 3.2 και τέλος εμφανίζεται η κλάση στην οποία ανήκει η εικόνα καθώς και ένα σύντομο μήνυμα με την περιγραφή της εικόνας αλλά και ένα ηχητικό απόσπασμα που περιγράφει την κατηγορία του σήματος.

3.3.1 Εισαγωγή Βιβλιοθηκών

Ο κώδικας ξεκινά με την εισαγωγή των απαραίτητων βιβλιοθηκών, όπως το NumPy για υπολογισμούς με πίνακες, το CV2 για επεξεργασία εικόνας, το pygame για αναπαραγωγή ήχου, το tkinter για τη δημιουργία γραφικού περιβάλλοντος και το tensorflow.keras για τη φόρτωση του μοντέλου του νευρωνικού δικτύου.

```
1 import numpy as np
2 import cv2
3 import pygame
4 import tkinter as tk
5 from tkinter import filedialog
6 from tensorflow.keras.models import load_model
```

Εικόνα 3.13: Εισαγωγή Βιβλιοθηκών

3.3.2 Συνάρτηση Προ-επεξεργασίας Εικόνας

Ορίζεται η συνάρτηση preprocess_image, η οποία δέχεται μια διαδρομή μιας εικόνας ως είσοδο. Η εικόνα φορτώνεται και μετατρέπεται σε μια μορφή που είναι κατάλληλη για την τροφοδότηση στο μοντέλο του νευρωνικού δικτύου. Η εικόνα αλλάζει μέγεθος σε 30x30 pixels και κανονικοποιείται σε τιμές από 0 έως 1.

```
## Συνάρτηση προεπεξεργασίας της εικόνας εισόδου
1 usage
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, dsize: (30, 30))
    img = img / 255.0  # Normalize the image
    return img
```

Εικόνα 3.14: Συνάρτηση Προ-επεξεργασίας

3.3.3 Φόρτωση Μοντέλου

Το μοντέλο του νευρωνικού δικτύου φορτώνεται από το αρχείο "testing.h5" με τη χρήση της συνάρτησης load_model. Αφού φορτωθεί το μοντέλο, εμφανίζεται ένα μήνυμα επιβεβαίωσης.

```
## Φόρτωση του αποθηκευμένου μοντέλου
model_filename = "testing.h5"
loaded_model = load_model(model_filename)
print("Model loaded successfully!")
```

Εικόνα 3.15: Φόρτωση μοντέλου

3.3.4 Δημιουργία Παραθύρου Tkinter και Επιλογή Εικόνας

Δημιουργείται ένα παράθυρο χρησιμοποιώντας το tk.Tk(), αλλά το παράθυρο είναι αόρατο καθώς είναι για εσωτερική χρήση. Χρησιμοποιείται η βιβλιοθήκη filedialog για να ανοίξει ένα παράθυρο περιήγησης αρχείων, όπου ο χρήστης μπορεί να επιλέξει μια εικόνα. Η διαδρομή της επιλεγμένης εικόνας αποθηκεύεται στη μεταβλητή input_image_path.

```
## Δημιουργία παραθύρου Tkinter (GUI)
root = tk.Tk()
root.withdraw()

## Χρήση file dialog για την επιλογή εικόνας
input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])

```

Εικόνα 3.16: Παράθυρο Tkinter & File dialog

3.3.5 Επεξεργασία και Πρόβλεψη Εικόνας

Αν υπάρχει μια εικόνα που επιλέχθηκε, τότε ξεκινά η διαδικασία πρόβλεψης. Η επιλεγμένη εικόνα υποβάλλεται στην συνάρτηση preprocess_image για προεπεξεργασία και κανονικοποίηση. Στη συνέχεια, προστίθεται μια διάσταση παρτίδας χρησιμοποιώντας το np.expand_dims και πραγματοποιείται πρόβλεψη της κλάσης χρησιμοποιώντας το φορτωμένο μοντέλο.

```
## Έλεγχος αν έχει επιλεγεί μια εικόνα
if input_image_path:
    print("Selected image:", input_image_path)

    # Προ-επεξεργασία της εικόνας εισόδου και δημιουργία πρόβλεψης
    preprocessed_img = preprocess_image(input_image_path)
    preprocessed_img = np.expand_dims(preprocessed_img, axis=0)  # Add batch dimension
    predicted_class = np.argmax(loaded_model.predict(preprocessed_img))  # Get the predicted class index
    print(predicted_class)

```

Εικόνα 3.17: Πρόβλεψη κλάσης

3.3.6 Κατηγορίες Μηνυμάτων

Δημιουργείται ένα λεξικό class_messages, το οποίο αντιστοιχεί κάθε προβλεπόμενη κλάση από το μοντέλο με ένα ζευγάρι πληροφοριών, το μήνυμα που περιγράφει την κλάση και το αρχείο ήχου που αντιστοιχεί σε αυτήν. Αυτό το λεξικό θα χρησιμοποιηθεί για να εμφανίσει το σωστό μήνυμα και να αναπαραγάγει τον σωστό ήχο που δημιουργήσαμε και αποθηκεύσαμε τοπικά νωρίτερα με χρήση online εργαλείου ανάλογα με την προβλεπόμενη κλάση.

Εικόνα 3.18: Ορισμός μηνυμάτων ανά κλάση

3.3.7 Εκτύπωση Προβλεπόμενου Μηνύματος

Εάν η προβλεπόμενη κλάση βρίσκεται στο λεξικό class_messages, τότε το κατάλληλο μήνυμα και το αντίστοιχο αρχείο ήχου ανακτώνται από το λεξικό. Στη συνέχεια, εμφανίζεται το προβλεπόμενο μήνυμα και εκκινεί η αναπαραγωγή του αντίστοιχου ήχου χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η προβλεπόμενη κλάση δεν υπάρχει στο λεξικό class_messages, εκτυπώνεται το μήνυμα "Unknown class", υποδηλώνοντας ότι δεν είναι γνωστή η σημασία αυτής της κλάσης.

```
91      ## Εκτύπωση του προβλεπόμενου μηνύματος
92      if predicted_class in class_messages:
93          class_message, audio_file = class_messages[predicted_class]
94          print("Predicted Class:", class_message)
95          pygame.mixer.init()
96          pygame.mixer.music.load(audio_file)
97          pygame.mixer.music.play()
98          while pygame.mixer.music.get_busy():
99              pygame.time.Clock().tick(10)
100     else:
101         print("Unknown class")
```

Εικόνα 3.19: Εκτύπωση Προβλεπόμενου Μηνύματος

3.4 Προβλέψεις μοντέλου CNN για χρήση σε HTML σελίδα

Έχει κατασκευαστεί ένας δεύτερος κώδικας προβλέψεων για το μοντέλο CNN με σκοπό την απεικόνιση της πρόβλεψης και του σχετικού ηχητικού αποσπάσματος σε μια ιστοσελίδα την οποία έχουμε ορίσει στον κώδικα AppCNN που θα αναλυθεί παρακάτω στο κεφάλαιο 3.5. Η διαφορά του κώδικα predictForHTML σε σχέση predictCNN είναι πως ο πρώτος επιστρέφει τα δεδομένα αφού τρέξει ώστε να μπορέσει να τα διαβάσει και να τα αξιοποιήσει ο κώδικας AppCNN ενώ ο κώδικας predictCNN απλά εμφανίζει τα δεδομένα.

3.4.1 Η δομή του predictForHTML

Αρχικά, εισάγονται οι παρακάτω απαραίτητες βιβλιοθήκες: NumPy, PIL, CV2 (OpenCV), tensorflow, και pygame.mixer.

```
1  import numpy as np
2  from PIL import Image
3  import cv2
4  import tensorflow as tf
5  import pygame.mixer
```

Εικόνα 3.20: Εισαγωγή Βιβλιοθηκών

Δημιουργεί μια κλάση με το όνομα traffic που αναγνωρίζει το σήμα και προβάλει μια περιγραφή καθώς αναπαράγει και ένα ηχητικό απόσπασμα που αντιστοιχεί στο συγκεκριμένο σήμα κυκλοφορίας. Συγκεκριμένα, η κλάση traffic έχει τις εξής βασικές λειτουργίες: Η __init__ δέχεται ως όρισμα το όνομα ενός αρχείου εικόνας και το αποθηκεύει ως μέλος της κλάσης, η μέθοδος trafficsign η οποία εκτελεί την πραγματική ανάλυση της εικόνας για τη αναγνώριση του σήματος και την αναπαραγωγή του αντίστοιχου ήχου. Ακολουθεί η trafficsign που φορτώνει το προ-εκπαιδευμένο μοντέλο CNN που δημιουργήσαμε νωρίτερα στο κεφάλαιο 3.2. Έπειτα φορτώνει την προς αναγνώριση εικόνα, την μετατρέπει σε μορφή που είναι κατάλληλη για το μοντέλο (μετασχηματισμός μεγέθους, κλιμάκωση κ.λπ.) Κάνει τις απαραίτητες επεξεργασίες στην εικόνα προτού την δώσει στο μοντέλο. Πιο συγκεκριμένα, κανονικοποιεί τις τιμές των pixel για να βρίσκονται μεταξύ 0 και 1. Περνάει την εικόνα μέσα από το μοντέλο για να πάρει μια πρόβλεψη για το ποιο σήμα αναγνωρίζει. Βρίσκει την κατηγορία σήματος με το

υψηλότερο βαθμό εμπιστοσύνης από τις προβλέψεις και τέλος βάσει της κατηγορίας αυτής, επιστέφει το κείμενο περιγραφής και το αντίστοιχο ηχητικό απόσπασμα.

```
2 usages
class traffic:
    def __init__(self,filename):
        self.filename =filename


    1 usage
    def trafficsign(self):

        model_path = "testing.h5"
        loaded_model = tf.keras.models.load_model(model_path)

        imagename = self.filename
        image = cv2.imread(imagename)

        image_fromarray = Image.fromarray(image, mode: 'RGB')
        resize_image = image_fromarray.resize((30, 30))
        expand_input = np.expand_dims(resize_image,axis=0)
        input_data = np.array(expand_input)
        input_data = input_data/255
        pred = loaded_model.predict(input_data)
        result = pred.argmax()
        pygame.mixer.init()
```

Εικόνα 3.21: Δημιουργία Πρόβλεψης

3.5 Κατασκευή Web Server

Στο κεφάλαιο αυτό γίνεται η ανάλυση της κατασκευής ενός Flask Web Server με σκοπό την δημοσίευση μιας HTML σελίδας για τοπική χρήση και έχει σαν όνομα AppCNN. Η λειτουργία της ιστοσελίδας θα αναλυθεί στο κεφάλαιο 3.6

3.5.1 Ανάλυση του AppCNN

Το σενάριο ξεκινά με την εισαγωγή απαραίτητων βιβλιοθηκών, όπως το Flask, το CORS (για τον έλεγχο του Cross-Origin Resource Sharing) και άλλες που χρησιμοποιούνται για τη διαχείριση των αιτημάτων και την αλληλεπίδραση με τον client. Αρχικά το Flask είναι ένα ελαφρύ πλαίσιο εφαρμογών που χρησιμοποιείται για την δημιουργία ιστοσελίδων web. Βοηθά στον χειρισμό των αιτημάτων από τους clients και την παραγωγή αποκρίσεων. Συγκεκριμένα στον κώδικα, χρησιμοποιείται για τη δημιουργία του Flask app, και να ορίσει τις διαδρομές και τις συναρτήσεις που εξυπηρετούν τα αιτήματα. Το Flask-CORS είναι μια βιβλιοθήκη που χρησιμοποιείται για τη διαχείριση του Cross-Origin Resource Sharing (CORS). Βοηθά στη διαχείριση των προβλημάτων ασφάλειας που προκύπτουν όταν ένας client προσπαθεί να κάνει αιτήσεις σε διαφορετικό domain από το server. Η βιβλιοθήκη PIL (Python Imaging Library) χρησιμοποιείται για την εργασία με εικόνες, για τη μετατροπή ενός πίνακα εικονοστοιχείων σε αντικείμενο εικόνας που μετά μειώνεται σε μέγεθος. Το NumPy είναι μια βιβλιοθήκη για την επιστημονική επεξεργασία δεδομένων στην Python, χρησιμοποιείται για τη διαχείριση των πινάκων δεδομένων των εικόνων. Το TensorFlow όπως έχει αναλυθεί και σε προηγούμενες ενότητες είναι μια βιβλιοθήκη ανοιχτού κώδικα για την εκπαίδευση και την ανάπτυξη μοντέλων μηχανικής μάθησης και νευρωνικών δικτύων, χρησιμοποιείται για τη φόρτωση του εκπαιδευμένου μοντέλου πρόβλεψης. Η ενότητα pygame.mixer χρησιμοποιείται την αναπαραγωγή ήχων κατά την αναγνώριση των σημάτων κυκλοφορίας. Η βιβλιοθήκη os χρησιμοποιείται για τη διαχείριση λειτουργιών του λειτουργικού συστήματος, όπως περιβαλλοντικές μεταβλητές και άλλες λειτουργίες.

```
from flask import Flask, request, jsonify, render_te
import os
from flask_cors import CORS, cross_origin
from utils.utils import decodeImage
from predictForHTML import traffic

os.putenv( _name: 'LANG', _value: 'en_US.UTF-8')
os.putenv( _name: 'LC_ALL', _value: 'en_US.UTF-8')

app = Flask(__name__)
CORS(app)
```

Εικόνα 3.22: Εισαγωγή Βιβλιοθηκών

Η κλάση AppCNN υλοποιεί τις απαραίτητες αρχικοποιήσεις για την εφαρμογή. Στο εσωτερικό της, δημιουργείται ένα αντικείμενο AppCNN, το οποίο περιλαμβάνει το όνομα αρχείου inputImage.jpg και ένα αντικείμενο traffic για την αναγνώριση των σημάτων. Οι διαδρομές αντιστοιχούν στις βασικές σελίδες της εφαρμογής. Η 1η απλά εμφανίζει την αρχική σελίδα της εφαρμογής, ενώ η 2η δέχεται POST αιτήματα από τον client. Στο εσωτερικό της, η εικόνα αποκωδικοποιείται και αποθηκεύεται, και η μέθοδος trafficsign() καλείται για την αναγνώριση του σήματος στην εικόνα. Το αποτέλεσμα της αναγνώρισης επιστρέφεται στον client σε μορφή JSON. Το if __name__ == "__main__": είναι η είσοδος στο κυρίως μέρος του προγράμματος. Εδώ δημιουργείται ένα αντικείμενο AppCNN για να αρχικοποιηθούν οι απαραίτητες ρυθμίσεις. Κατόπιν, το Flask app εκκινεί μέσω της μεθόδου run(), όπου ορίζεται ως IP "0.0.0.0" και πόρτα 5000, επιτρέποντας στην εφαρμογή να είναι προσβάσιμη από οποιαδήποτε διεύθυνση IP.

```python
class AppCNN:
    def __init__(self):
        self.filename = "inputImage.jpg"
        self.classifier = traffic(self.filename)


@app.route( rule: "/", methods=['GET'])
@cross_origin()
def home():
    return render_template('index.html')


@app.route( rule: "/predict", methods=['POST'])
@cross_origin()
def predictRoute():
    image = request.json['image']
    decodeImage(image, clApp.filename)
    result = clApp.classifier.trafficsign()
    return jsonify(result)


# port = int(os.getenv("PORT"))
if __name__ == "__main__":
    clApp = AppCNN()
    # app.run(host='0.0.0.0', port=port)
    app.run(host='0.0.0.0', port=5000)
```

Εικόνα 3.23: Κατασκευή AppCNN

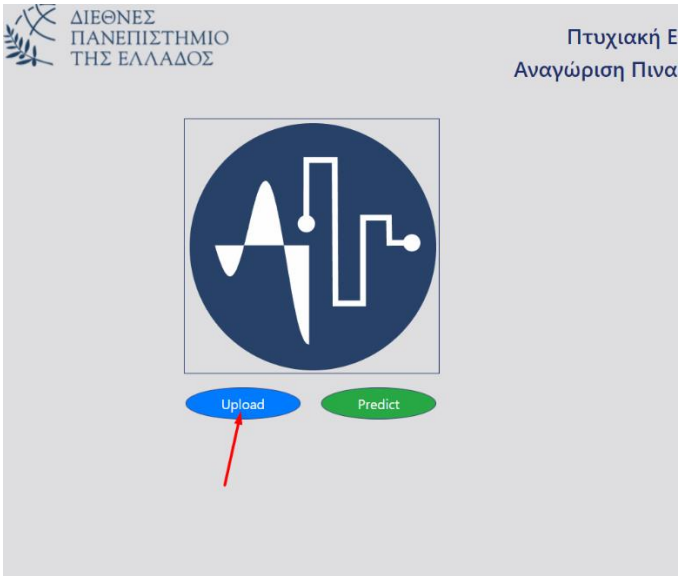
3.6 Ιστοσελίδα διεπαφής χρήστη

Ανάλυση και σχολιασμός της δομής της ιστοσελίδας καθώς και του τρόπου λειτουργίας της

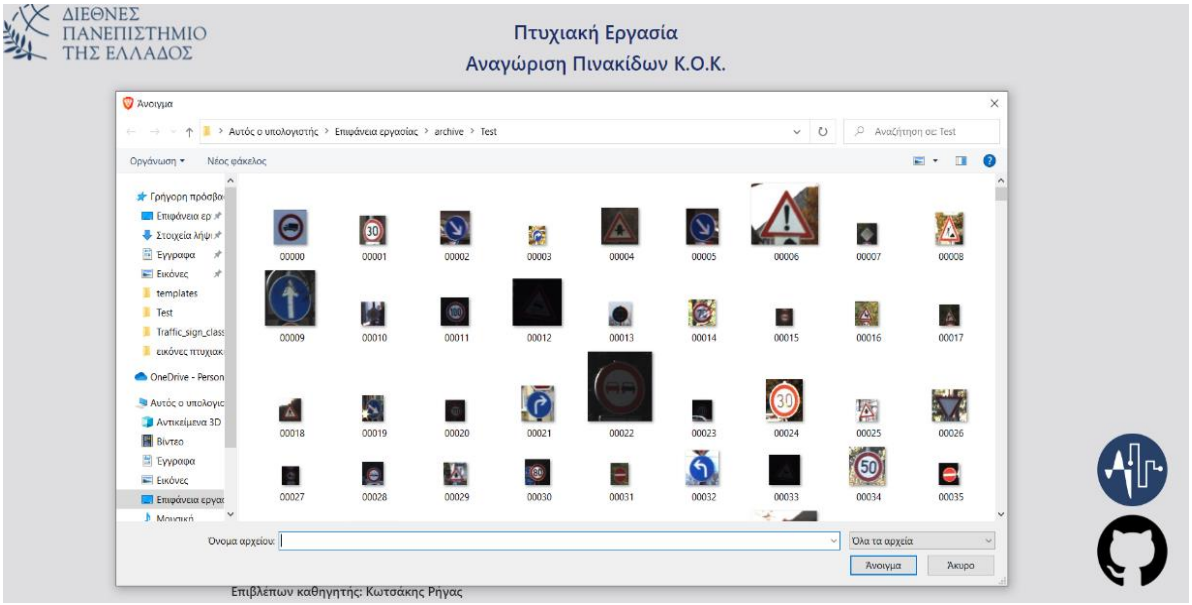
Εικόνα 3.24: Interface Εφαρμογής

Η ιστοσελίδα της εφαρμογής έχει την ιδιότητα να δέχεται εικόνες και να εμφανίζει μέσω του κώδικα Predict ένα αποτέλεσμα πρόβλεψης για την εικόνα που έχει δεχτεί αλλά και να κάνει αναπαραγωγή ενός αρχείου ήχου. Με το πάτημα του κουμπιού Upload η σελίδα εμφανίζει ένα παράθυρο για την εισαγωγή του αρχείου εικόνας που θέλουμε να εξετάσουμε. Αφού έχει επιλεχθεί το προς εξέταση αρχείο με το πάτημα του κουμπιού Predict γίνεται ο έλεγχος της εικόνας, και εμφανίζεται η περιγραφή της εικόνας που αναγνωρίστηκε και ο αντίστοιχος ήχος που έχει οριστεί στον κώδικα. Επίσης στην ιστοσελίδα υπάρχει ο τίτλος της εργασίας καθώς και τρία εικονίδια συνέσμων που οδηγούν στις ιστοσελίδες της σχολής, του τμήματος και του Github που βρίσκεται αναρτημένη η εργασία. Τέλος στο κάτω μέρος της σελίδας αναγρφαφονται τα ονόματα των φοιτητών που εκπόνησαν την εργασία και του επιβλέπωντα καθηγητή τους.

Ακολουθούν εικόνες για την επείδηξη του τρόπου λειτουργίας της ιστοσελίδας.



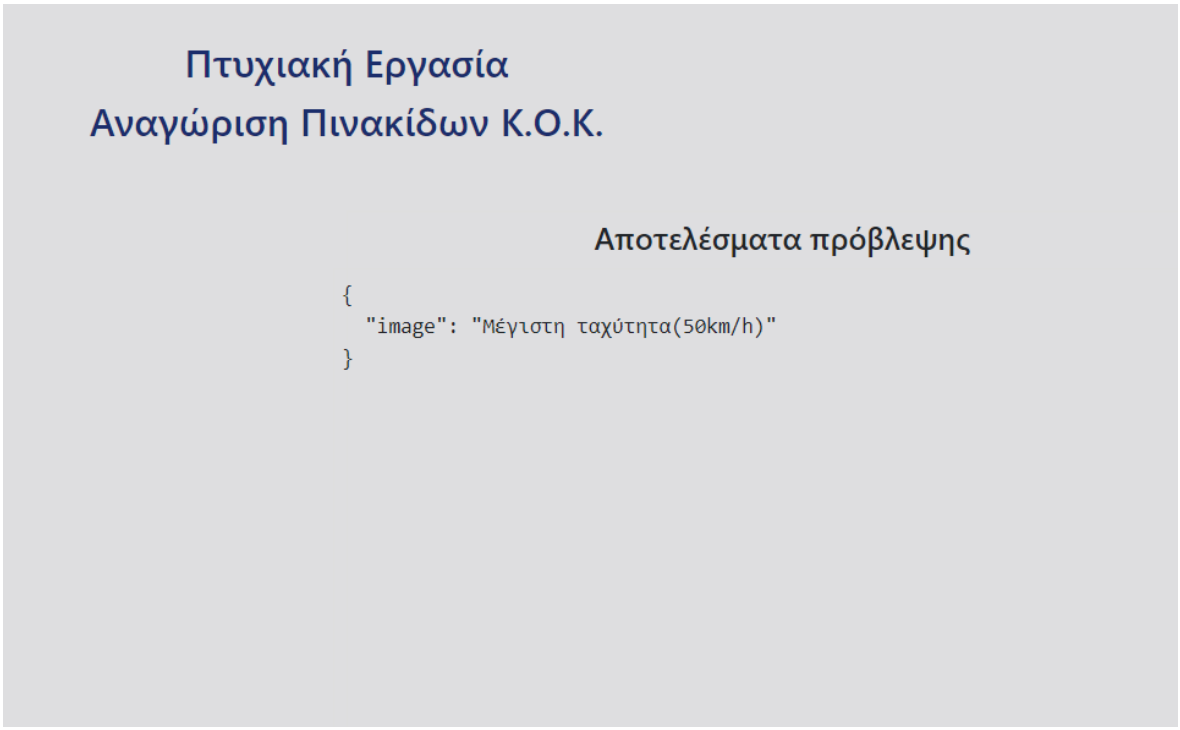
Εικόνα 3.25: Επιλογή κουμπιού Upload για επιλογή εικόνας



Εικόνα 3.26: Επιλογή τυχαίας εικόνας προς αναγνώριση

Εικόνα 3.27: Επιλογή κουμπιού Predict για την δημιουργία πρόβλεψης

Εικόνα 3.28: Αναγνώριση πινακίδας Κ.Ο.Κ. (όριο 50km/h)



Εικόνα 3.29: Εμφάνιση αποτελέσματος πρόβλεψης

Κεφάλαιο 4: Ανάλυση τεχνολογίας Linear Regression

Η γραμμική παλινδρόμηση είναι ίσως ένας από τους πιο γνωστούς και κατανοητούς αλγορίθμους στη στατιστική και τη μηχανική μάθηση. Η γραμμική παλινδρόμηση αναπτύχθηκε στον τομέα της στατιστικής και μελετάται ως μοντέλο για την κατανόηση της σχέσης μεταξύ αριθμητικών μεταβλητών εισόδου και εξόδου, αλλά με την πάροδο του χρόνου έχει γίνει αναπόσπαστο μέρος της σύγχρονης εργαλειοθήκης της μηχανικής μάθησης.[11]

Η γραμμική παλινδρόμηση χρησιμοποιεί τη σχέση μεταξύ των σημείων δεδομένων για να σχεδιάσει μια ευθεία γραμμή μέσω όλων αυτών. Αυτή η γραμμή μπορεί να χρησιμοποιηθεί για την πρόβλεψη μελλοντικών τιμών.[12]

Η ανάλυση γραμμικής παλινδρόμησης χρησιμοποιείται για την πρόβλεψη της τιμής μιας μεταβλητής με βάση την τιμή μιας άλλης μεταβλητής. Η μεταβλητή που θέλουμε να προβλέψουμε ονομάζεται εξαρτημένη μεταβλητή. Η μεταβλητή που χρησιμοποιείται για να προβλέψουμε την τιμή της άλλης μεταβλητής ονομάζεται ανεξάρτητη μεταβλητή.[13]

Αυτή η μορφή ανάλυσης εκτιμά τους συντελεστές της γραμμικής εξίσωσης, η οποία περιλαμβάνει μία ή περισσότερες ανεξάρτητες μεταβλητές που προβλέπουν καλύτερα την τιμή της εξαρτημένης μεταβλητής. Η γραμμική παλινδρόμηση προσαρμόζει μια ευθεία γραμμή ή επιφάνεια που ελαχιστοποιεί τις αποκλίσεις μεταξύ των προβλεπόμενων και των πραγματικών τιμών εξόδου. Υπάρχουν απλοί υπολογιστές γραμμικής παλινδρόμησης που χρησιμοποιούν μια μέθοδο "ελαχίστων τετραγώνων" για να ανακαλύψουν την καλύτερη δυνατή ευθεία για ένα σύνολο ζευγαρωμένων δεδομένων. Στη συνέχεια, εκτιμάται η τιμή του X (εξαρτημένη μεταβλητή) από το Y (ανεξάρτητη μεταβλητή).

Τα μοντέλα γραμμικής παλινδρόμησης είναι σχετικά απλά και παρέχουν έναν εύκολο στην ερμηνεία μαθηματικό τύπο που μπορεί να παράγει προβλέψεις. Η γραμμική παλινδρόμηση μπορεί να εφαρμοστεί σε διάφορους τομείς των επιχειρήσεων και των ακαδημαϊκών σπουδών.

Η γραμμική παλινδρόμηση χρησιμοποιείται στα πάντα, από τις βιολογικές, συμπεριφορικές, περιβαλλοντικές και κοινωνικές επιστήμες μέχρι τις επιχειρήσεις. Τα

μοντέλα γραμμικής παλινδρόμησης έχουν γίνει ένας αποδεδειγμένος τρόπος για την επιστημονική και αξιόπιστη πρόβλεψη του μέλλοντος. Επειδή η γραμμική παλινδρόμηση είναι μια μακροχρόνια καθιερωμένη στατιστική διαδικασία, οι ιδιότητες των μοντέλων γραμμικής παλινδρόμησης είναι καλά κατανοητές και μπορούν να εκπαιδευτούν πολύ γρήγορα.[13]

4.1 Υλοποίηση του μοντέλου με χρήση τεχνολογίας Linear regression

Ο κώδικας που ακολουθεί παρουσιάζει τον τρόπο φόρτωσης , επεξεργασίας και εκπαίδευσης ενός απλού γραμμικού μοντέλου παλινδρόμησης. Θα υπάρξει αναλυτική επεξήγηση καθώς και στιγμιότυπα του κώδικα.

4.1.1 Εισαγωγή απαραίτητων βιβλιοθηκών

Γίνεται πρώτα εισαγωγή των απαραίτητων βιβλιοθηκών όπως το os για τις λειτουργίες του συστήματος αρχείων, το CV2 για την επεξεργασία εικόνας, το NumPy για τους υπολογισμούς του πίνακα, το Pickle για την αποθήκευση και τη φόρτωση αντικειμένων και τα train_test_split, LinearRegression και mean_squared_error από το scikit-learn .

```
1  import os
2  import cv2
3  import numpy as np
4  import pickle
5  from sklearn.model_selection import train_test_split
6  from sklearn.linear_model import LinearRegression
7  from sklearn.metrics import mean_squared_error
```

Εικόνα 4.1: Εισαγωγή Βιβλιοθηκών

4.1.2 Ορισμός παραμέτρων εικόνας

Οι διαστάσεις εικόνας ορίζονται ως IMG_HEIGHT και IMG_WIDTH, με τιμές 30 pixels εκάστη.

```
9   # Ορισμός παραμέτρων εικόνας
10  IMG_HEIGHT = 30
11  IMG_WIDTH = 30
```

Εικόνα 4.2: Ορισμός Διαστάσεων Εικόνας

4.1.3 Φόρτωση και προ-επεξεργασία δεδομένων

Σε αυτό το τμήμα του κώδικα, οι εικόνες φορτώνονται από διάφορες κατηγορίες (φακέλους),υποβάλλονται σε προ-επεξεργασία και τα δεδομένα και οι αντίστοιχες ετικέτες αποθηκεύονται σε κατάλληλη μορφή που μπορεί να χρησιμοποιηθεί για την εκπαίδευση του μοντέλου. Ο βρόχος for ξεκινάει ανατρέχοντας σε κάθε φάκελο κατηγορίας μέσα στον φάκελο data_dir, όπου κάθε φάκελος αντιστοιχεί σε μια κατηγορία των εικόνων. Για κάθε εικόνα, η διαδρομή του αρχείου υπολογίζεται χρησιμοποιώντας τη συνάρτηση os.path.join (category_path, img_file), η εικόνα φορτώνεται χρησιμοποιώντας τη βιβλιοθήκη CV2 και αποθηκεύεται στη μεταβλητή image. Η εικόνα image αλλάζει μέγεθος στο ύψος και το πλάτος χρησιμοποιώντας τη συνάρτηση cv2.resize. Το νέο μέγεθος ορίζεται από τις σταθερές IMG_HEIGHT και IMG_WIDTH. Η εικόνα resized_image μετατρέπεται σε μονοδιάστατο πίνακα (επιπεδωποίηση) χρησιμοποιώντας την μέθοδο flatten(). Αυτός ο πίνακας αποθηκεύεται στον πίνακα image_data, που περιέχει όλα τα δεδομένα των εικόνων. Επίσης, η ετικέτα της κατηγορίας category προστίθεται στον πίνακα labels, που αντιστοιχεί στις ετικέτες των κατηγοριών των εικόνων. Τέλος, οι πίνακες image_data και labels αποθηκεύουν τα προ-επεξεργασμένα δεδομένα και τις αντίστοιχες ετικέτες που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου.

```python
# Φόρτωση και προ-επεξεργασία δεδομένων
1 usage
def load_and_preprocess_data(data_dir):
    image_data = []
    labels = []

    for category in os.listdir(data_dir):
        category_path = os.path.join(data_dir, category)
        for img_file in os.listdir(category_path):
            img_path = os.path.join(category_path, img_file)
            image = cv2.imread(img_path)
            resized_image = cv2.resize(image, dsize: (IMG_HEIGHT, IMG_WIDTH))
            image_data.append(resized_image.flatten())
            labels.append(int(category))

    image_data = np.array(image_data)
    labels = np.array(labels)
    return image_data, labels
```

Εικόνα 4.3: Φόρτωση και προ-επεξεργασία

Στη συνέχεια γίνεται η φόρτωση και προ-επεξεργασία των δεδομένων εικόνας μέσω της συνάρτησης load_and_preprocess_data. Ορίζεται η διαδρομή του φακέλου που περιέχει τα δεδομένα εικόνας. Το αντικείμενο X θα περιέχει τα προ-επεξεργασμένα δεδομένα εικόνας, ενώ το αντικείμενο y θα περιέχει τις αντίστοιχες ετικέτες των κατηγοριών των εικόνων. Η συνάρτηση load_and_preprocess_data(data_dir) καλείται για να εκτελέσει τις ενέργειες φόρτωσης και προ-επεξεργασίας των δεδομένων εικόνας μέσα από το φάκελο data_dir. Στη συνέχεια, τα προεπεξεργασμένα δεδομένα αποθηκεύονται στο αντικείμενο X και οι αντίστοιχες ετικέτες αποθηκεύονται στο αντικείμενο y, προετοιμάζοντας τα δεδομένα για την εκπαίδευση του μοντέλου. Μετά από αυτήν την ενέργεια, τα δεδομένα X και y είναι έτοιμα για χρήση στην εκπαίδευση και αξιολόγηση του μοντέλου.

4.1.4 Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης

Χρησιμοποιώντας τη συνάρτηση train_test_split, τα δεδομένα εικόνας X και οι ετικέτες y διαχωρίζονται σε σύνολα εκπαίδευσης και επικύρωσης. Σε αυτήν την περίπτωση, το 30% των δεδομένων χρησιμοποιείται για το σύνολο επικύρωσης.

```
34  # Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
35  X_train, X_val, y_train, y_val = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
```

Εικόνα 4.4: Σύνολα εκπαίδευσης και επικύρωσης

4.1.5 Δημιουργία και εκπαίδευση του μοντέλου

Ο συγκεκριμένος κώδικας δημιουργεί ένα αντικείμενο model τύπου LinearRegression και το εκπαιδεύει χρησιμοποιώντας τα δεδομένα εκπαίδευσης X_train και y_train. Το μοντέλο προσπαθεί να εκτιμήσει μια γραμμική σχέση μεταξύ των χαρακτηριστικών (πεπλατυσμένων εικόνων) και των ετικετών (κατηγοριών) των εικόνων.

4.1.6 Πρόβλεψη ετικετών για το σύνολο επικύρωσης

Χρησιμοποιώντας το εκπαιδευμένο μοντέλο, οι ετικέτες προβλέπονται για το σύνολο επικύρωσης X_val.

```
41  # Πρόβλεψη ετικετών για το σύνολο επικύρωσης
42  y_pred = model.predict(X_val)
```

Εικόνα 4.5: Πρόβλεψη ετικετών

4.1.7 Υπολογισμός του μέσου τετραγωνικού σφάλματος (MSE) για την αξιολόγηση

Το MSE υπολογίζεται μεταξύ της πραγματικής ετικέτας επικύρωσης y_val και της ετικέτας πρόβλεψης y_pred το MSE αντιπροσωπεύει το μέσο όρο των τετραγώνων των αποκλίσεων μεταξύ των πραγματικών και των προβλεπόμενων τιμών ένα χαμηλότερο MSE υποδηλώνει μεγαλύτερη ακρίβεια του μοντέλου.

```
44  # Υπολογισμός του μέσου τετραγωνικού σφάλματος (MSE) για την αξιολόγηση
45  mse = mean_squared_error(y_val, y_pred)
46  print("Mean Squared Error:", mse)
```

Εικόνα 4.6: Υπολογισμός MSE

4.1.8 Αποθήκευση του εκπαιδευμένου μοντέλου χρησιμοποιώντας το Pickle

Το εκπαιδευμένο μοντέλο αποθηκεύεται σε ένα αρχείο, στην περίπτωση μας με όνομα trained_linear_regression_model.pkl με τη χρήση του Pickle. Το αρχείο δημιουργείται με δυαδικό τρόπο εγγραφής ('wb' για δυαδικό τρόπο εγγραφής).

```
48  # Αποθήκευση του εκπαιδευμένου μοντέλου χρησιμοποιώντας το pickle
49  model_filename = "trained_linear_regression_model.pkl"
50  with open(model_filename, 'wb') as model_file:
51      pickle.dump(model, model_file)
52  print("Model saved as", model_filename)
```

Εικόνα 4.7: Αποθήκευση Μοντέλου

4.2 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης Linear Regression

Ο παρακάτω κώδικας αναλύει τις ενέργειες που πρέπει να εκτελεστούν για την πρόβλεψη των κατηγοριών των πινακίδων σήμανσης και την εμφάνιση των αντίστοιχων μηνυμάτων καθώς και των ηχητικών αποσπασμάτων.

4.2.1 Φόρτωση απαραίτητων βιβλιοθηκών

Εισάγονται οι απαραίτητες βιβλιοθήκες για την φόρτωση και επεξεργασία εικόνας (OpenCV), την φόρτωση εκπαιδευμένου μοντέλου (Pickle), την αναπαραγωγή ήχου (pygame), καθώς και την δημιουργία γραφικής διεπαφής (tkinter) για την επιλογή της εικόνας.

```
1  import cv2
2  import pickle
3  import pygame
4  import tkinter as tk
5  from tkinter import filedialog
```

Εικόνα 4.8: Εισαγωγή Βιβλιοθηκών

4.2.2 Ορισμός συνάρτησης για προ-επεξεργασία εικόνας

Η συνάρτηση αυτή δέχεται ως όρισμα το μονοπάτι μιας εικόνας, τη φορτώνει και την επεξεργάζεται μειώνοντας το μέγεθος της σε 30x30 pixel. Το αποτέλεσμα είναι ένας πίνακας που περιέχει τις τιμές των pixel της εικόνας.

4.2.3 Φόρτωση του εκπαιδευμένου μοντέλου

Το εκπαιδευμένο μοντέλο φορτώνεται από το αρχείο trained_linear_regression_model.pkl χρησιμοποιώντας τη βιβλιοθήκη Pickle.

```
14  # Φόρτωση του αποθηκευμένου μοντέλου με χρήση του pickle
15  model_filename = "trained_linear_regression_model.pkl"
16  with open(model_filename, 'rb') as model_file:
17      loaded_model = pickle.load(model_file)
18  print("Model loaded successfully!")
```

Εικόνα 4.9: Φόρτωση Μοντέλου

4.2.4 Δημιουργία παραθύρου Tkinter (GUI)

Δημιουργείται ένα παράθυρο γραφικής διεπαφής της βιβλιοθήκης tkinter και το κύριο παράθυρο κρύβεται (withdrawn) από την προβολή.

```
20  # Δημιουργία παραθύρου Tkinter (GUI)
21  root = tk.Tk()
22  root.withdraw()
```

Εικόνα 4.10: Παράθυρο Tkinter

4.2.5 Επιλογή εικόνας με το file dialog

Χρησιμοποιείται το file dialog για να επιλεγεί μια εικόνα. Ο χρήστης έχει τη δυνατότητα να επιλέξει αρχεία εικόνας μεταξύ jpg και png.

```
24  # Χρήση file dialog για επιλογή εικόνας
25  input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])
```

Εικόνα 4.11: Επιλογή εικόνας

4.2.6 Έλεγχος αν έχει επιλεγεί εικόνα

Γίνεται έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα ώστε να συνεχίσει ο κώδικας παρακάτω στην πρόβλεψη κατηγορίας.

```
27  # Έλεγχος αν έχει επιλεγεί μια εικόνα
28  if input_image_path:
29      print("Selected image:", input_image_path)
```

Εικόνα 4.12: Έλεγχος αν επιλέχθηκε αρχείο

4.2.7 Προ-επεξεργασία της εικόνας εισόδου και δημιουργία πρόβλεψης κλάσης

Η εικόνα επεξεργάζεται από τη συνάρτηση preprocess_image και το φορτωμένο μοντέλο χρησιμοποιείται για την πρόβλεψη της κλάσης της εικόνας.

```
32  # Προ-επεξεργασία της εικόνας εισόδου και δημιουργία πρόβλεψης κλάσης
33  preprocessed_img = preprocess_image(input_image_path)
34  predicted_class = loaded_model.predict([preprocessed_img])[0]
35  print(predicted_class)
```

Εικόνα 4.13: Πρόβλεψη κλάσης

4.2.8 Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης

Κάθε κατηγορία σήμανσης έχει έναν αριθμό κλάσης που την αντιπροσωπεύει. Για παράδειγμα η κλάση 0 αντιστοιχεί στην πινακίδα που επιβάλει όριο ταχύτητας 20 χιλιόμετρα ανά ώρα και περιλαμβάνει το αντίστοιχο μήνυμα και ηχητικό απόσπασμα. Συνολικά έχουμε 43 κλάσεις με την τελευταία να αντιστοιχεί στην πινακίδα "Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα" και στο αντίστοιχο ηχητικό απόσπασμα.

```
# Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης
class_messages = {
    0: ("Μέγιστη ταχύτητα(20km/h)", "sounds/20kmh.mp3"),
    1: ("Μέγιστη ταχύτητα(30km/h)", "sounds/30kmh.mp3"),
    2: ("Μέγιστη ταχύτητα(50km/h)", "sounds/50kmh.mp3"),
    3: ("Μέγιστη ταχύτητα(60km/h)", "sounds/60kmh.mp3"),
    4: ("Μέγιστη ταχύτητα(70km/h)", "sounds/70kmh.mp3"),
    5:  ("Μέγιστη ταχύτητα(80km/h)", "sounds/80kmh.mp3"),
    6: ("Τέλος ορίου ταχύτητας που έχει επιβληθεί με απαγορευτική πινακίδα (80km/h)", "sounds/telos80.mp3"),
    7: ("Μέγιστη ταχύτητα(100km/h)", "sounds/100kmh.mp3"),
    8: ("Μέγιστη ταχύτητα(120km/h)", "sounds/120kmh.mp3"),
    9: ("Απαγορεύεται το προσπέρασμα των μηχανοκινήτων οχημάτων, πλήν των διτρόχων μοτοσυκλετών χωρίς κάνιστρο", "sounds/passing.mp3"),
    10: ("Απαγορεύεται στους οδηγούς φορτηγών αυτοκινήτων μεγίστου επιτρεπόμενου βάρους που υπερβαίνει τους 3,5 τόννους να προσπερνούν άλλα οχήματα", "sounds/passing3,5.mp3"),
    11: ("Διασταύρωση με οδό, πάνω στην οποία αυτοί που κινούνται ωφείλουν να παραχωρήσουν προτεραιότητα", "sounds/diastavrosiProteraiotita.mp3"),
    12: ("Οδός προτεραιότητας", "sounds/priorityRoad.mp3"),
    13: ("Υποχρεωτική παραχώρηση προτεραιότητας", "sounds/yield.mp3"),
    31: ("Κίνδυνος απο διέλευση άγριων ζώων", "sounds/31.mp3"),
    32: ("Τέλος όλων των τοπικών απαγορεύσεων οι οποίες έχουν επιβληθεί με απαγορευτικές πινακίδες στα κινούμενα οχήματα", "sounds/32.mp3"),
    33: ("Υποχρεωτική κατεύθυνση πορείας με στροφή δεξιά", "sounds/33.mp3"),
    34: ("Υποχρεωτική κατεύθυνση πορείας με στροφή αριστερά", "sounds/34.mp3"),
    35: ("Υποχρεωτική κατεύθυνση πορείας πρός τα εμπρός", "sounds/35.mp3"),
    36: ("Υποχρεωτική κατεύθυνση πορείας εμπρός η δεξιά", "sounds/36.mp3"),
    37: ("Υποχρεωτική κατεύθυνση πορείας εμπρός ή αριστερά", "sounds/37.mp3"),
    38: ("Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου", "sounds/38.mp3"),
    39: ("Υποχρεωτική διέλευση μόνο από την αριστερή πλευρά της νησίδος η του εμποδίου", "sounds/39.mp3"),
    40: ("Κυκλική υποχρεωτική διαδρομή", "sounds/40.mp3"),
    41: ("Τέλος απαγόρευσης προσπεράσματος το οποίο έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/41.mp3"),
    42: ("Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/42.mp3")
}
```

Εικόνα 4 14: Ορισμός μηνύματος κάθε κλάσης

4.2.9 Εκτύπωση του προβλεπόμενου μηνύματος κλάσης

Αν η πρόβλεψη της κατηγορίας που προκύπτει από το μοντέλο είναι συμπεριλαμβανόμενη στο λεξικό class_messages, τότε το αντίστοιχο μήνυμα και αρχείο ήχου λαμβάνονται και εμφανίζονται στην οθόνη. Επιπλέον, ο ήχος αναπαράγεται χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η πρόβλεψη δεν αντιστοιχεί σε καμία κατηγορία που ορίζεται στο class_messages, τότε εμφανίζεται το μήνυμα "Unknown class".

```
# Εκτύπωση του προβλεπόμενου μηνύματος κλάσης
if predicted_class in class_messages:
    class_message, audio_file = class_messages[predicted_class]
    print("Predicted Class:", class_message)
    # Load and play audio using pygame
    pygame.mixer.init()
    pygame.mixer.music.load(audio_file)
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy():
        pygame.time.Clock().tick(10)
else:
    print("Unknown class")
```

Εικόνα 4.15: Εκτύπωση Προβλεπόμενου Μηνύματος

Κεφάλαιο 5: Ανάλυση τεχνολογίας Decision Tree

Ένα δέντρο αποφάσεων είναι μια δενδροειδής δομή που μοιάζει με διάγραμμα ροής, όπου ένας εσωτερικός κόμβος αντιπροσωπεύει ένα χαρακτηριστικό ,ο κλάδος αντιπροσωπεύει έναν κανόνα απόφασης και κάθε κόμβος φύλλου αντιπροσωπεύει το αποτέλεσμα. [14]

Ο κορυφαίος κόμβος σε ένα δέντρο αποφάσεων είναι γνωστός ως κόμβος ρίζας. Μαθαίνει να χωρίζει με βάση την τιμή του χαρακτηριστικού. Κατανέμει το δέντρο με αναδρομικό τρόπο που ονομάζεται αναδρομική κατάτμηση. Αυτή η δομή που μοιάζει με διάγραμμα ροής σας βοηθά στη λήψη αποφάσεων. Είναι οπτικοποίηση όπως ένα διάγραμμα διαγράμματος ροής που μιμείται εύκολα τη σκέψη σε ανθρώπινο επίπεδο. Αυτός είναι ο λόγος για τον οποίο τα δέντρα αποφάσεων είναι εύκολο να κατανοηθούν και να ερμηνευτούν.[14]

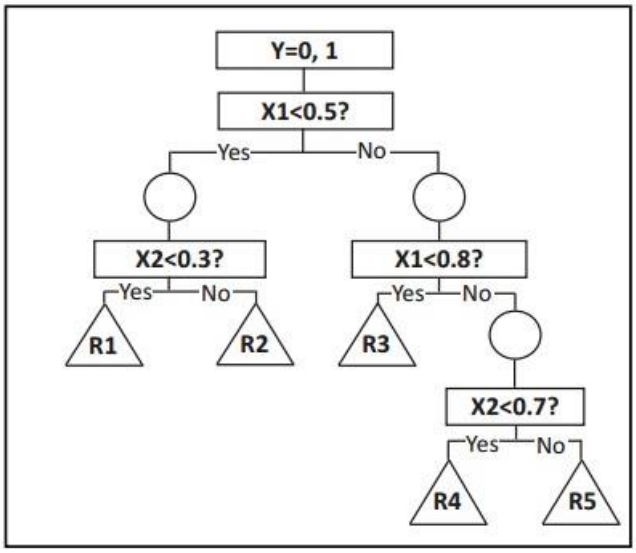
Το δέντρο αποφάσεων είναι ένας αλγόριθμος ML τύπου white box. Μοιράζεται την εσωτερική λογική λήψης αποφάσεων, η οποία δεν είναι διαθέσιμη στους αλγορίθμους τύπου μαύρου κουτιού, όπως με ένα νευρωνικό δίκτυο. Ο χρόνος εκπαίδευσής του είναι ταχύτερος σε σύγκριση με τον αλγόριθμο του νευρωνικού δικτύου.[14]

Η χρονική πολυπλοκότητα των δέντρων αποφάσεων είναι συνάρτηση του αριθμού των εγγραφών και των χαρακτηριστικών στα δεδομένα που δίνονται. Το δέντρο αποφάσεων είναι μια μέθοδος χωρίς κατανομή ή μη παραμετρική μέθοδος που δεν εξαρτάται από υποθέσεις κατανομής πιθανοτήτων. Τα δέντρα αποφάσεων μπορούν να χειριστούν δεδομένα υψηλών διαστάσεων με καλή ακρίβεια.[14]

Τα Δέντρα Αποφάσεων (ΔΑ) είναι μια μη παραμετρική μέθοδος μάθησης με επίβλεψη που χρησιμοποιείται για ταξινόμηση και παλινδρόμηση. Ο στόχος είναι η δημιουργία ενός μοντέλου που προβλέπει την τιμή μιας μεταβλητής-στόχου με την εκμάθηση απλών κανόνων απόφασης που προκύπτουν από τα χαρακτηριστικά των δεδομένων. Ένα δέντρο μπορεί να θεωρηθεί ως μια προσέγγιση σταθερών κατά τεμάχια.[15]

Τα δέντρα αποφάσεων μπορούν να εφαρμοστούν σε θέματα παλινδρόμησης ως προσέγγιση στην προγνωστική ανάλυση για την πρόβλεψη εξόδων από αόρατα δεδομένα.

Είναι δημοφιλή στην κοινότητα της μηχανικής μάθησης ως μορφές δομημένων μοντέλων. Η δενδροειδής δομή είναι εύκολα κατανοητή και μας επιτρέπει να αναλύσουμε γρήγορα τη διαδικασία λήψης αποφάσεων. Η επεξηγηματικότητα, η οποία αναφέρεται στην κατανόηση της εξόδου ενός μοντέλου, αποτελεί ζωτικό μέρος της μηχανικής μάθησης. Πρόκειται για ένα ισχυρό εργαλείο που εντοπίζει τις αδυναμίες του μοντέλου και επηρεάζει τα δεδομένα. Βοηθά στην επαλήθευση των προβλέψεων για τη βελτίωση της απόδοσης του μοντέλου και αποκτά νέες γνώσεις σχετικά με ένα πρόβλημα. Πρέπει να γνωρίζουμε την ακρίβεια ενός δέντρου για να καταλήξουμε σε μια απόφαση για τη δημιουργία στρατηγικών διαχωρισμών. Τα κριτήρια απόφασης διαφέρουν ανάλογα με τα δέντρα παλινδρόμησης και ταξινόμησης. Τα δέντρα αποφάσεων χρησιμοποιούν διαφορετικούς αλγορίθμους για να χωρίσουν έναν κόμβο σε διαφορετικούς υποκόμβους. Η δημιουργία υποκόμβων αυξάνει την ομοιογένεια των επερχόμενων υποκόμβων. Η καθαρότητα του κόμβου αυξάνεται ανάλογα με τη μεταβλητή-στόχο. Το δέντρο αποφάσεων θα χωρίσει τους κόμβους σε όλες τις μεταβλητές και στη συνέχεια θα επιλέξει τη διάσπαση που οδηγεί στους πιο ομοιογενείς υποκόμβους. Η επιλογή του αλγορίθμου εξαρτάται σε μεγάλο βαθμό από τον τύπο της μεταβλητής-στόχου.[16]



Εικόνα 5.1: Decision Tree

5.1 Υλοποίηση του μοντέλου με χρήση τεχνολογίας Decision Tree

Ο παρακάτω κώδικας εκτελεί μια προκαθορισμένη διαδικασία εκπαίδευσης και αξιολόγησης ενός μοντέλου Decision Tree (Δέντρου Αποφάσεων) για την αναγνώριση κατηγοριών εικόνων πινακίδων κυκλοφορίας.

5.1.1 Εισαγωγή απαραίτητων βιβλιοθηκών

Όπως είδαμε και προηγουμένως η ενότητα "Εισαγωγή απαραίτητων βιβλιοθηκών" αποτελεί το πρώτο μέρος του κώδικα και εξηγεί ποιες βιβλιοθήκες χρειάζονται για να εκτελεστεί ο υπόλοιπος κώδικας. Οι βιβλιοθήκες είναι συλλογές από προκατασκευασμένες συναρτήσεις, κλάσεις και εργαλεία που μπορούν να χρησιμοποιηθούν για συγκεκριμένες εργασίες. Ποιο συγκεκριμένα η os που παρέχει συναρτήσεις για τη διαχείριση του λειτουργικού συστήματος, όπως τη διαχείριση αρχείων και φακέλων, η CV2 (OpenCV) η οποία είναι μια βιβλιοθήκη επεξεργασίας εικόνας και βίντεο. Χρησιμοποιείται εδώ για τη φόρτωση και την επεξεργασία των εικόνων, η NumPy που είναι μια βιβλιοθήκη για τον υπολογισμό μεγάλων αριθμητικών πινάκων και προσφέρει πολλές χρήσιμες συναρτήσεις για την επεξεργασία αριθμητικών δεδομένων, η Pickle που χρησιμοποιείται για την αποθήκευση και τη φόρτωση αντικειμένων Python σε δυαδική μορφή συγκεκριμένα χρησιμοποιείται για την αποθήκευση του εκπαιδευμένου μοντέλου και τέλος η scikit-learn χρησιμοποιείται για την εκπαίδευση και αξιολόγηση του μοντέλου Δέντρου Αποφάσεων.

```
1  import os
2  import cv2
3  import numpy as np
4  import pickle
5  from sklearn.model_selection import train_test_split
6  from sklearn.tree import DecisionTreeClassifier
7  from sklearn.metrics import accuracy_score
```

Εικόνα 5.2: Εισαγωγή Βιβλιοθηκών

5.1.2 Καθορισμός παραμέτρων εικόνας

Εδώ ορίζονται οι παράμετροι IMG_HEIGHT και IMG_WIDTH, οι οποίες καθορίζουν τις διαστάσεις στις οποίες θα τροποποιηθεί η εικόνα για την εκπαίδευση και την αξιολόγηση. Και στις δυο παραμέτρους εκχωρούνται οι τιμές 30.

```
9   # Καθορισμός παραμέτρων εικόνας
10  IMG_HEIGHT = 30
11  IMG_WIDTH = 30
12
```

Εικόνα 5.3: Καθορισμός Παραμέτρων

5.1.3 Φόρτωση και προ-επεξεργασία δεδομένων

Η συνάρτηση load_and_preprocess_data φορτώνει δεδομένα εικόνας από τον καθορισμένο φάκελο data_dir και εκτελεί προ-επεξεργασία. Κάθε εικόνα φορτώνεται, παίρνει νέες διαστάσεις και αποθηκεύεται στη λίστα image_data και οι αντίστοιχες ετικέτες αποθηκεύονται στη λίστα image_labels.

```
13  # Φόρτωση και προ-επεξεργασία δεδομένων
14  def load_and_preprocess_data(data_dir):
15      image_data = []
16      image_labels = []
17
18      for category in os.listdir(data_dir):
19          category_path = os.path.join(data_dir, category)
20          for img_file in os.listdir(category_path):
21              img_path = os.path.join(category_path, img_file)
22              image = cv2.imread(img_path)
23              resized_image = cv2.resize(image, dsize: (IMG_HEIGHT, IMG_WIDTH))
24              image_data.append(resized_image)
25              image_labels.append(int(category))
26
27      image_data = np.array(image_data)
28      image_labels = np.array(image_labels)
29      return image_data, image_labels
```

Εικόνα 5.4: Φόρτωση και προ-επεξεργασία

5.1.4 Φόρτωση και προ-επεξεργασία δεδομένων

Σε αυτό το σημείο φορτώνονται τα δεδομένα εικόνας από τον καθορισμένο φάκελο data_dir. Η μεταβλητή data_dir δέχεται σαν παράμετρο την διαδρομή για τα δεδομένα εκπαίδευσης.

```
31  # Φόρτωση και προ-επεξεργασία δεδομένων
32  data_dir = r"C:\Users\stathis\Desktop\archive\Train"
33  X, y = load_and_preprocess_data(data_dir)
```

Εικόνα 5.5: Φόρτωση και προ-επεξεργασία

5.1.5 Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης

Χρησιμοποιείται η συνάρτηση train_test_split της βιβλιοθήκης scikit-learn (sklearn) για να χωριστεί το σύνολο δεδομένων σε δύο σύνολα, ένα εκπαίδευσης και ένα επικύρωσης. Με αυτόν τον τρόπο, ένα μοντέλο μπορεί να εκπαιδευτεί σε ένα υποσύνολο των δεδομένων και να αξιολογηθεί σε ένα άλλο υποσύνολο που δεν χρησιμοποιήθηκε κατά την εκπαίδευση. Η παράμετρος test_size=0.3 υποδηλώνει ότι το 30% των δεδομένων θα χρησιμοποιηθεί για το σύνολο επικύρωσης, ενώ το υπόλοιπο 70% για το σύνολο εκπαίδευσης. Η παράμετρος random_state=42 καθορίζει την κατάσταση τυχαιότητας, εξασφαλίζοντας ότι η διαίρεση των δεδομένων θα είναι συνεπής κάθε φορά που εκτελείτε ο κώδικας.

```
35  # Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
36  X_train, X_val, y_train, y_val = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
```

Εικόνα 5.6: Δεδομένα εκπαίδευσης και Επικύρωσης

5.1.6 Επιπεδοποίηση των εικόνων για το Decision Tree

Σε αυτό το τμήμα του κώδικα εκτελείται μια διαδικασία προ-επεξεργασίας των εικόνων πριν χρησιμοποιηθούν για την εκπαίδευση και την αξιολόγηση του μοντέλου. Έχουμε το X_train που είναι ένα σύνολο δεδομένων με εικόνες για εκπαίδευση. Κάθε εικόνα είναι ένας πίνακας που περιέχει τιμές πίξελ για κάθε σημείο της εικόνας. Το X_train είναι ένας τρισδιάστατος πίνακας, με διαστάσεις (αριθμός εικόνων, ύψος εικόνας,

πλάτος εικόνας). Το X_train μετατρέπεται σε έναν μονοδιάστατο πίνακα, όπου κάθε εικόνα απλοποιείται και γίνεται μια μονοδιάστατη σειρά τιμών. Αυτό κάνει τις εικόνες κατανοητές από το μοντέλο. Το X_train.shape[0] είναι ο αριθμός των εικόνων στο X_train. Ουσιαστικά, το reshape αλλάζει τη μορφή του πίνακα, αλλά διατηρεί τον αριθμό των στοιχείων του. Το -1 στη δεύτερη διάσταση του reshape σημαίνει "αυτόματος υπολογισμός". Αυτό επιτρέπει στο NumPy να υπολογίσει τον αριθμό των στηλών (των στοιχείων της μονοδιάστατης σειράς) ανάλογα με τον αριθμό των στοιχείων στην αρχική δισδιάστατη μορφή του X_train. Με αυτόν τον τρόπο, κάθε εικόνα αντιπροσωπεύεται τώρα από έναν μονοδιάστατο πίνακα με σειριακές τιμές πίξελ. Ομοίως γίνεται και για το σύνολο επικύρωσης X_val.

```
38  # Επιπεδοποίηση των εικόνων για το Decision Tree
39  X_train_flat = X_train.reshape(X_train.shape[0], -1)
40  X_val_flat = X_val.reshape(X_val.shape[0], -1)
```

Εικόνα 5.7: Επιπεδοποίηση

5.1.7 Δημιουργία και εκπαίδευση μοντέλου Decision Tree

Η κλάση DecisionTreeClassifier από τη βιβλιοθήκη sklearn χρησιμοποιείται για τη δημιουργία ενός μοντέλου δέντρου αποφάσεων. Αυτό το μοντέλο εκπαιδεύεται στα δεδομένα εκπαίδευσης (x_train_flat, y_train).

```
42  # Δημιουργία και εκπαίδευση μοντέλου
43  model = DecisionTreeClassifier(random_state=42)
44  model.fit(X_train_flat, y_train)
```

Εικόνα 5.8: Εκπαίδευση Μοντέλου

5.1.8 Αξιολόγηση του μοντέλου στο σύνολο επικύρωσης

Εδώ προβλέπονται οι κατηγορίες για το σύνολο επικύρωσης (X_val_flat), συγκρίνονται οι προβλέψεις με τις πραγματικές ετικέτες (y_val) και υπολογίζεται η ακρίβεια του μοντέλου. Τέλος εμφανίζεται το ποσοστό ακρίβειας στην οθόνη.

```
46  # Αξιολόγηση του μοντέλου στο σύνολο επικύρωσης
47  y_pred = model.predict(X_val_flat)
48  accuracy = accuracy_score(y_val, y_pred)
49  print("Validation Accuracy:", accuracy)
```

Εικόνα 5.9: Αξιολόγηση στο σύνολο επικύρωσης

5.1.9 Αποθήκευση του εκπαιδευμένου μοντέλου

Το εκπαιδευμένο μοντέλο αποθηκεύεται σε ένα αρχείο χρησιμοποιώντας τη βιβλιοθήκη Pickle. Αυτό επιτρέπει την μελλοντική χρήση του χωρίς την ανάγκη επανεκπαίδευσης.

```
51  # Αποθήκευση του εκπαιδευμένου μοντέλου
52  model_filename = "trained_decision_tree_model.pkl"
53  with open(model_filename, 'wb') as model_file:
54      pickle.dump(model, model_file)
55  print("Model saved as", model_filename)
```

Εικόνα 5.10: Αποθήκευση Μοντέλου

5.2 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης Decision Tree

Ο κώδικας που ακολουθεί χρησιμοποιείται για την αναγνώριση και πρόβλεψη σημάτων οδικής κυκλοφορίας βάσει του προ-εκπαιδευμένου μοντέλου Decision Tree που δημιουργήθηκε και παρουσιάστηκε στην ενότητα 5.1. Παρακάτω παρουσιάζεται η δομή του κώδικα αναλυτικά.

5.2.1 Εισαγωγή απαιτούμενων βιβλιοθηκών

Όπως στους προηγούμενους κώδικες που αναλύθηκαν παραπάνω έτσι και σε αυτόν γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών για την ορθή λειτουργεία του. Οι βιβλιοθήκες αυτές είναι οι NumPy (np), CV2, Pickle, pygame, tkinter (tk) και filedialog.

```
1  import numpy as np
2  import cv2
3  import pickle
4  import pygame
5  import tkinter as tk
6  from tkinter import filedialog
```

Εικόνα 5.11: Εισαγωγή Βιβλιοθηκών

5.2.2 Ορισμός συνάρτησης για προ-επεξεργασία εικόνας

Είναι το τμήμα του κώδικα που περιέχει την συνάρτηση που χρησιμοποιείται για την προ-επεξεργασία μιας εικόνας πριν χρησιμοποιηθεί από το μοντέλο αναγνώρισης, πιο συγκεκριμένα δηλώνεται η συνάρτηση def preprocess_image(image_path), "preprocess_image" είναι το όνομα της συνάρτησης και "image_path" είναι η παράμετρος που αναμένει η συνάρτηση, δηλαδή η διαδρομή προς το αρχείο εικόνας για επεξεργασία. Η γραμμή img = cv2.imread(image_path) διαβάζει την εικόνα από το αρχείο που περιέχεται στο image_path και αποθηκεύει την εικόνα ως έναν πίνακα (array) τιμών που αντιπροσωπεύει τα χρώματα των εικονοστοιχείων. Η γραμμή img = cv2.resize(img, (30, 30)) αλλάζει το μέγεθος της εικόνας σε 30x30 pixels. Η γραμμή img = img.flatten() κάνει επιπεδοποίηση τον πίνακα της εικόνας, μετατρέπει δηλαδή τον

δισδιάστατο πίνακα εικονοστοιχείων σε έναν μονοδιάστατο πίνακα. Τέλος επιστρέφει τον επεξεργασμένο πίνακα εικονοστοιχείων.

```
8   # Ορισμός συνάρτησης για προ-επεξεργασία εικόνας
9   def preprocess_image(image_path):
10      img = cv2.imread(image_path)
11      img = cv2.resize(img, dsize: (30, 30))
12      img = img.flatten()
13      return img
```

Εικόνα 5.12: Προ-επεξεργασία εικόνας

5.2.3 Φόρτωση του προ-εκπαιδευμένου μοντέλου Decision Tree με χρήση του Pickle

Το μοντέλο φορτώνει το προ-εκπαιδευμένο αρχείο που δημιουργήθηκε προηγουμένως στο κεφάλαιο 5.1.

```
15  # Φόρτωση αποθηκευμένου μοντέλου χρησιμοποιώντας το pickle
16  model_filename = "trained_decision_tree_model.pkl"
17  with open(model_filename, 'rb') as model_file:
18      loaded_model = pickle.load(model_file)
19  print("Model loaded successfully!")
```

Εικόνα 5.13: Φόρτωση Μοντέλου

5.2.4 Δημιουργία παραθύρου Tkinter (GUI)

Σε αυτόν του κώδικα, το root = tk.Tk() δημιουργεί ένα νέο γραφικό παράθυρο, που στην ουσία δημιουργείται προσωρινά για την επιλογή αρχείου. Ενώ το root.withdraw() καλείται για να αποκρύψει το παράθυρο αυτό.

```
21  # Δημιουργία παραθύρου Tkinter (GUI)
22  root = tk.Tk()
23  root.withdraw()
```

Εικόνα 5.14: Παράθυρο Tkinter

5.2.5 Χρήση του file dialog για να επιλεγεί μια εικόνα

Χρησιμοποιείται το file dialog για να επιλεγεί μια εικόνα. Ο χρήστης έχει τη δυνατότητα να επιλέξει αρχεία εικόνας μεταξύ jpg και png.

```
25  # Χρήση του file dialog για να επιλεγεί μια εικόνα
26  input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])
```

Εικόνα 5.15: Επιλογή Εικόνας

5.2.6 Έλεγχος εάν επιλέχθηκε εικόνα

Γίνεται έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα ώστε να συνεχίσει ο κώδικας παρακάτω στην πρόβλεψη κατηγορίας.

```
28  # Έλεγχος εάν επιλέχθηκε εικόνα
29  if input_image_path:
30      print("Selected image:", input_image_path)
```

Εικόνα 5.16: Έλεγχος Επιλογής Αρχείου

5.2.7 Προ-επεξεργασία της εικόνας εισόδου και δημιουργία πρόβλεψης κλάσης

Η εικόνα επεξεργάζεται από τη συνάρτηση preprocess_image και το φορτωμένο μοντέλο χρησιμοποιείται για την πρόβλεψη της κλάσης της εικόνας.

```
33  # Προ-επεξεργασία της εικόνας εισόδου και δημιουργία πρόβλεψης κλάσης
34  preprocessed_img = preprocess_image(input_image_path)
35  predicted_class = loaded_model.predict([preprocessed_img])[0]  # Get the predicted class
36  print(predicted_class)
```

Εικόνα 5.17: Πρόβλεψη Κλάσης

5.2.8 Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης

Κάθε κατηγορία σήμανσης έχει έναν αριθμό κλάσης που την αντιπροσωπεύει. Για παράδειγμα η κλάση 0 αντιστοιχεί στην πινακίδα που επιβάλει όριο ταχύτητας 20 χιλιόμετρα ανά ώρα και περιλαμβάνει το αντίστοιχο μήνυμα και ηχητικό απόσπασμα. Συνολικά έχουμε 43 κλάσεις με την τελευταία να αντιστοιχεί στην πινακίδα "Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα" και στο αντίστοιχο ηχητικό απόσπασμα.

```
# Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης
class_messages = {
    0: ("Μέγιστη ταχύτητα(20km/h)", "sounds/20kmh.mp3"),
    1: ("Μέγιστη ταχύτητα(30km/h)", "sounds/30kmh.mp3"),
    2: ("Μέγιστη ταχύτητα(50km/h)", "sounds/50kmh.mp3"),
    3: ("Μέγιστη ταχύτητα(60km/h)", "sounds/60kmh.mp3"),
    4: ("Μέγιστη ταχύτητα(70km/h)", "sounds/70kmh.mp3"),
    5:  ("Μέγιστη ταχύτητα(80km/h)", "sounds/80kmh.mp3"),
    6: ("Τέλος ορίου ταχύτητας που έχει επιβληθεί με απαγορευτική πινακίδα (80km/h)", "sounds/telos80.mp3"),
    7: ("Μέγιστη ταχύτητα(100km/h)", "sounds/100kmh.mp3"),
    8: ("Μέγιστη ταχύτητα(120km/h)", "sounds/120kmh.mp3"),
    9: ("Απαγορεύεται το προσπέρασμα των μηχανοκινήτων οχημάτων, πλήν των δίτροχων μοτοσυκλετών χωρίς κάνιστρο", "sounds/passing.mp3"),
    10: ("Απαγορεύεται στους οδηγούς φορτηγών αυτοκινήτων μεγίστου επιτρεπόμενου βάρους που υπερβαίνει τους 3,5 τόννους να προσπερνούν άλλα οχήματα", "sounds/passing3,5.mp3"),
    11: ("Διασταύρωση με οδό, πάνω στην οποία αυτοί που κινούνται υφείλουν να παραχωρήσουν προτεραιότητα", "sounds/diastavrosiProteraiotita.mp3"),
    12: ("Οδός προτεραιότητας", "sounds/priorityRoad.mp3"),
    13: ("Υποχρεωτική παραχώρηση προτεραιότητας", "sounds/yield.mp3"),
    31: ("Κίνδυνος απο διέλευση άγριων ζώων", "sounds/31.mp3"),
    32: ("Τέλος όλων των τοπικών απαγορεύσεων οι οποίες έχουν επιβληθεί με απαγορευτικές πινακίδες στα κινούμενα οχήματα", "sounds/32.mp3"),
    33: ("Υποχρεωτική κατεύθυνση πορείας με στροφή δεξιά", "sounds/33.mp3"),
    34: ("Υποχρεωτική κατεύθυνση πορείας με στροφή αριστερά", "sounds/34.mp3"),
    35: ("Υποχρεωτική κατεύθυνση πορείας πρός τα εμπρός", "sounds/35.mp3"),
    36: ("Υποχρεωτική κατεύθυνση πορείας εμπρός η δεξιά", "sounds/36.mp3"),
    37: ("Υποχρεωτική κατεύθυνση πορείας εμπρός ή αριστερά", "sounds/37.mp3"),
    38: ("Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου", "sounds/38.mp3"),
    39: ("Υποχρεωτική διέλευση μόνο από την αριστερή πλευρά της νησίδος η του εμποδίου", "sounds/39.mp3"),
    40: ("Κυκλική υποχρεωτική διαδρομή", "sounds/40.mp3"),
    41: ("Τέλος απαγόρευσης προσπεράσματος το οποίο έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/41.mp3"),
    42: ("Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/42.mp3")
}
```

Εικόνα 5.18: Ορισμός μηνύματος για κάθε κλάση

5.2.9 Εκτύπωση του προβλεπόμενου μηνύματος κλάσης

Αν η πρόβλεψη της κατηγορίας που προκύπτει από το μοντέλο είναι συμπεριλαμβανόμενη στο λεξικό class_messages, τότε το αντίστοιχο μήνυμα και αρχείο ήχου λαμβάνονται και εμφανίζονται στην οθόνη. Επιπλέον, ο ήχος αναπαράγεται χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η πρόβλεψη δεν αντιστοιχεί σε καμία κατηγορία που ορίζεται στο class_messages, τότε εμφανίζεται το μήνυμα "Unknown class".

```
86  # Εκτύπωση του προβλεπόμενου μηνύματος κλάσης
87  if predicted_class in class_messages:
88      class_message, audio_file = class_messages[predicted_class]
89      print("Predicted Class:", class_message)
90      # Load and play audio using pygame
91      pygame.mixer.init()
92      pygame.mixer.music.load(audio_file)
93      pygame.mixer.music.play()
94      while pygame.mixer.music.get_busy():
95          pygame.time.Clock().tick(10)
96  else:
97      print("Unknown class")
98
```

Εικόνα 5.19: Εκτύπωση Προβλεπόμενου Μηνύματος

Κεφάλαιο 6: Ανάλυση τεχνολογίας Logistic Regression

Η λογιστική παλινδρόμηση αποσκοπεί στην επίλυση προβλημάτων ταξινόμησης. Αυτό το επιτυγχάνει προβλέποντας κατηγορικά αποτελέσματα, σε αντίθεση με τη γραμμική παλινδρόμηση που προβλέπει ένα συνεχές αποτέλεσμα. Η λογιστική παλινδρόμηση αποσκοπεί στην επίλυση προβλημάτων ταξινόμησης. Αυτό το επιτυγχάνει προβλέποντας κατηγορικά αποτελέσματα, σε αντίθεση με τη γραμμική παλινδρόμηση που προβλέπει ένα συνεχές αποτέλεσμα.

Η λογιστική παλινδρόμηση μπορεί να χρησιμοποιηθεί για διάφορα προβλήματα ταξινόμησης, όπως η ανίχνευση ανεπιθύμητης αλληλογραφίας. πρόβλεψη διαβήτη, αν ένας συγκεκριμένος πελάτης θα αγοράσει ένα συγκεκριμένο προϊόν ή θα αλλάξει έναν άλλο ανταγωνιστή, αν ο χρήστης θα κάνει κλικ σε έναν συγκεκριμένο διαφημιστικό σύνδεσμο ή όχι, και πολλά άλλα παραδείγματα υπάρχουν στον κάδο.[18]

Η λογιστική παλινδρόμηση είναι ένας από τους πιο απλούς και συχνά χρησιμοποιούμενους αλγορίθμους μηχανικής μάθησης για ταξινόμηση δύο κατηγοριών. Είναι εύκολο να υλοποιηθεί και μπορεί να χρησιμοποιηθεί ως βασική γραμμή για οποιοδήποτε πρόβλημα δυαδικής ταξινόμησης. Οι βασικές θεμελιώδεις έννοιες της είναι επίσης εποικοδομητικές στη βαθιά μάθηση. Η λογιστική παλινδρόμηση περιγράφει και εκτιμά τη σχέση μεταξύ μιας εξαρτημένης δυαδικής μεταβλητής και ανεξάρτητων μεταβλητών.

Η ονομασία "λογιστική παλινδρόμηση" προέρχεται από την έννοια της λογιστικής συνάρτησης που χρησιμοποιεί. Η λογιστική συνάρτηση είναι επίσης γνωστή ως σιγμοειδής συνάρτηση. Η τιμή αυτής της λογιστικής συνάρτησης βρίσκεται μεταξύ του μηδενός και του ενός.[19]

6.1 Υλοποίηση του μοντέλου με χρήση τεχνολογίας Logistic Regression

Ο παρακάτω κώδικας εκτελεί μια προκαθορισμένη διαδικασία εκπαίδευσης και αξιολόγησης ενός μοντέλου λογιστικής παλινδρόμησης (Logistic Regression) για την αναγνώριση κατηγοριών εικόνων πινακίδων κυκλοφορίας.

6.1.1 Εισαγωγή απαραίτητων βιβλιοθηκών

Όπως και στους στις τεχνολογίες που αναλύθηκαν παραπάνω στα προηγούμενα κεφάλαια έτσι και για την Logistic Regression κατά την κατασκευή του κώδικα εκπαίδευσης γίνεται εισαγωγή των απαραίτητων και κατάλληλων βιβλιοθηκών. Οι βιβλιοθήκες αυτές είναι οι NumPy, os, CV2, train_test_split από τη βιβλιοθήκη sklearn, LogisticRegression από τη βιβλιοθήκη sklearn και η Pickle.

```
1  import numpy as np
2  import os
3  import cv2
4  from sklearn.model_selection import train_test_split
5  from sklearn.linear_model import LogisticRegression
6  import pickle
```

Εικόνα 6.1: Εισαγωγή Βιβλιοθηκών

6.1.2 Ορισμός διαδρομής για το dataset

Η διαδρομή για το dataset ορίζεται στην μεταβλητή data_dir. Αυτή είναι η διαδρομή προς τον φάκελο που περιέχει τα δεδομένα του dataset.

```
8  # Ορισμός διαδρομής για το dataset
9  data_dir = r"C:\Users\stathis\Desktop\archive"
```

Εικόνα 6.2: Ορισμός Διαδρομής DataSet

6.1.3 Ορισμός συνάρτησης φόρτωσης και προ-επεξεργασίας των δεδομένων

Η συνάρτηση load_data αναλαμβάνει τη φόρτωση των εικόνων και των αντίστοιχων ετικετών από τον φάκελο "Train". Για κάθε εικόνα, φορτώνεται, μετατρέπεται σε grayscale και σε συνέχεια σε ένα διάνυσμα επίπεδης μορφής. Οι ετικέτες αναπαριστούν την κατηγορία της εικόνας.

```python
# Ορισμός συνάρτησης φόρτωσης και προ-επεξεργασίας των δεδομένων
1 usage
def load_data(data_dir):
    images = []
    labels = []

    for label in os.listdir(os.path.join(data_dir, 'Train')):
        path = os.path.join(data_dir, 'Train', label)
        for img_name in os.listdir(path):
            img_path = os.path.join(path, img_name)
            img = cv2.imread(img_path)
            img = cv2.resize(img, dsize: (IMG_HEIGHT, IMG_WIDTH))
            images.append(img.flatten())
            labels.append(int(label))

    return np.array(images), np.array(labels)
```

Εικόνα 6.3: Φόρτωση και Προ-επεξεργασία

6.1.4 Ορισμός διαστάσεων εικόνων

Οι διαστάσεις των εικόνων ορίζονται στις μεταβλητές IMG_HEIGHT και IMG_WIDTH. Οι εικόνες θα ανασχηματιστούν σε αυτές τις διαστάσεις κατά την προ-επεξεργασία δηλαδή 30x30 Pixels.

```python
# Ορισμός διαστάσεων εικόνων
IMG_HEIGHT = 30
IMG_WIDTH = 30
```

Εικόνα 6.4: Ορισμός Διαστάσεων

6.1.5 Φόρτωση και προ-επεξεργασία των δεδομένων

Τα δεδομένα φορτώνονται και προ-επεξεργάζονται χρησιμοποιώντας τη συνάρτηση load_data. Οι εικόνες μετατρέπονται σε επίπεδη μορφή και αποθηκεύονται στη μεταβλητή X, ενώ οι ετικέτες αποθηκεύονται στη μεταβλητή y.

```
31  # Φόρτωση και προ-επεξεργασία των δεδομένων
32  X, y = load_data(data_dir)
```

Εικόνα 6.5: Φόρτωση και Προ-επεξεργασία

6.1.6 Διαίρεση δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης

Χρησιμοποιείται η συνάρτηση train_test_split για να διαχωριστούν τα δεδομένα σε σύνολα εκπαίδευσης και επικύρωσης. Το 70% των δεδομένων χρησιμοποιείται για την εκπαίδευση, ενώ το 30% για την επικύρωση. Τα δεδομένα εκπαίδευσης αποθηκεύονται στις μεταβλητές X_train και y_train, ενώ τα δεδομένα επικύρωσης αποθηκεύονται στις μεταβλητές X_val και y_val.

```
34  # Διαίρεση των δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
35  X_train, X_val, y_train, y_val = train_test_split( *arrays: X, y, test_size=0.3, random_state=42, shuffle=True)
```

Εικόνα 6.6: Σύνολα εκπαίδευσης και επικύρωσης

6.1.7 Αρχικοποίηση και εκπαίδευση του μοντέλου λογιστικής παλινδρόμησης

Δημιουργείται ένα αντικείμενο μοντέλου λογιστικής παλινδρόμησης χρησιμοποιώντας την κλάση LogisticRegression από τη βιβλιοθήκη sklearn. Το μοντέλο αρχικοποιείται με τον αριθμό μέγιστων επαναλήψεων max_iter. Έπειτα, το μοντέλο εκπαιδεύεται με τα δεδομένα εκπαίδευσης (X_train και y_train).

```
37  # Αρχικοποίηση και εκπαίδευση του μοντέλου λογιστικής παλινδρόμησης
38  model = LogisticRegression(max_iter=10000)
39  model.fit(X_train, y_train)
```

Εικόνα 6.7: Εκπαίδευση Μοντέλου

6.1.8 Αποθήκευση του εκπαιδευμένου μοντέλου με χρήση του Pickle

Το εκπαιδευμένο μοντέλο αποθηκεύεται σε ένα αρχείο με το όνομα trained_logistic_regression_model.pkl χρησιμοποιώντας το Pickle. Αυτό το αρχείο

περιέχει τα στοιχεία του μοντέλου που είναι απαραίτητα για να γίνει η επαναφορά του στο μέλλον ή η χρήση του για δημιουργία προβλέψεων.

```
41  # Αποθήκευση του εκπαιδευμένου μοντέλου με χρήση του pickle
42  model_filename = "trained_logistic_regression_model.pkl"
43  with open(model_filename, 'wb') as model_file:
44      pickle.dump(model, model_file)
45  print("Model saved as", model_filename)
46
```

Εικόνα 6 8: Αποθήκευση Μοντέλου

6.2 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης Logistic Regression

Ο κώδικας που ακολουθεί χρησιμοποιείται για την αναγνώριση και πρόβλεψη σημάτων οδικής κυκλοφορίας βάσει του προ-εκπαιδευμένου μοντέλου Logistic Regression που δημιουργήθηκε και παρουσιάστηκε στην ενότητα 6.1. Παρακάτω παρουσιάζεται η δομή του κώδικα αναλυτικά.

6.2.1 Εισαγωγή Βιβλιοθηκών

Εισάγονται οι απαραίτητες βιβλιοθήκες για την ορθή λειτουργία του κώδικα. Αυτές είναι οι CV2 (OpenCV) για επεξεργασία εικόνων, η Pickle για τη φόρτωση και την αποθήκευση αντικειμένων σε αρχεία, η pygame για την αναπαραγωγή ήχου, και η tkinter για τη διεπαφή χρήστη.

6.2.2 Συνάρτηση Προ-επεξεργασίας Εικόνας

Ορίζεται η συνάρτηση preprocess_image που επιτελεί την προ-επεξεργασία μιας εικόνας, παίρνει μια εικόνα ως είσοδο και επιτελεί διάφορες επεξεργασίες για να προετοιμάσει την εικόνα για την πρόβλεψη. Πιο συγκεκριμένα στο img = cv2.imread(image_path) η συνάρτηση cv2.imread αναγνωρίζει το μονοπάτι της εικόνας (το image_path) και φορτώνει την εικόνα σε μορφή πίνακα. Ο πίνακας περιέχει τις πληροφορίες για τις τιμές των χρωμάτων (RGB) κάθε pixel της εικόνας.

Στο κομμάτι img = cv2.resize(img, (30, 30)) η συνάρτηση cv2.resize ρυθμίζει το μέγεθος της εικόνας στις διαστάσεις 30x30 pixels. Αυτό γίνεται για να εξασφαλιστεί ότι οι εικόνες που θα χρησιμοποιηθούν για την πρόβλεψη έχουν συγκεκριμένο μέγεθος. Στο img = img.flatten() η μέθοδος flatten μετατρέπει τον δισδιάστατο πίνακα των χρωμάτων σε έναν μονοδιάστατο πίνακα. Αυτό είναι απαραίτητο για να μπορεί το μοντέλο να δεχθεί την προ-επεξεργασμένη εικόνα ως είσοδο. Το τελικό αποτέλεσμα της λειτουργίας προ-επεξεργασίας είναι ο μονοδιάστατος πίνακας img, ο οποίος περιέχει τις τιμές των pixel της εικόνας που έχουν μετατραπεί και σχεδιαστεί κατάλληλα για να δοθούν ως είσοδο στο μοντέλο.

```
# Συνάρτηση Προ-επεξεργασίας Εικόνας
1 usage
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, dsize: (30, 30))
    img = img.flatten()
    return img
```

Εικόνα 6:9: Προ-επεξεργασία εικόνας

6.2.3 Φόρτωση του Εκπαιδευμένου Μοντέλου

Φορτώνεται το εκπαιδευμένο μοντέλο από το αρχείο

```
# Φόρτωση του Εκπαιδευμένου Μοντέλου
model_filename = "trained_logistic_regression_model.pkl"
with open(model_filename, 'rb') as model_file:
    loaded_model = pickle.load(model_file)
print("Model loaded successfully!")
```

Εικόνα 6.10: Φόρτωση Μοντέλου

6.2.4 Δημιουργία GUI και Επιλογή Εικόνας

Δημιουργείται ένα GUI παράθυρο με χρήση της βιβλιοθήκης tkinter, και μέσω του filedialog ο χρήστης μπορεί να επιλέξει μια εικόνα

```
# Δημιουργία GUI και Επιλογή Εικόνας
root = tk.Tk()
root.withdraw()

input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])

```

Εικόνα 6.11: Tkinter παράθυρο και επιλογή εικόνας

6.2.5 Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα

Γίνεται έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα κατά το προηγούμενο βήμα από το αρχείο. Εάν όντως έχει γίνει επιλογή εικόνας ο κώδικας προχωράει παρακάτω στην πρόβλεψη εικόνας.

```
# Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα
if input_image_path:
    print("Selected image:", input_image_path)
```

Εικόνα 6.12: Έλεγχος επιλογής αρχείου

6.2.6 Προ-επεξεργασία και Πρόβλεψη της Εικόνας

Η εικόνα που επιλέχθηκε προ-επεξεργάζεται και στη συνέχεια προβλέπεται και εκτυπώνεται η κλάση της.

```
# Προ-επεξεργασία και δημιουργία Πρόβλεψης της Εικόνας
preprocessed_img = preprocess_image(input_image_path)
predicted_class = loaded_model.predict([preprocessed_img])[0]  # Get the predicted class
print(predicted_class)
```

Εικόνα 6.13: Δημιουργία Πρόβλεψης Κλάσης

6.2.7 Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης

Κάθε κατηγορία σήμανσης έχει έναν αριθμό κλάσης που την αντιπροσωπεύει. Για παράδειγμα η κλάση 0 αντιστοιχεί στην πινακίδα που επιβάλει όριο ταχύτητας 20 χιλιόμετρα ανά ώρα και περιλαμβάνει το αντίστοιχο μήνυμα και ηχητικό απόσπασμα. Συνολικά έχουμε 43 κλάσεις με την τελευταία να αντιστοιχεί στην πινακίδα "Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα" και στο αντίστοιχο ηχητικό απόσπασμα.

```
# Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης
class_messages = {
    0: ("Μέγιστη ταχύτητα(20km/h)", "sounds/20kmh.mp3"),
    1: ("Μέγιστη ταχύτητα(30km/h)", "sounds/30kmh.mp3"),
    2: ("Μέγιστη ταχύτητα(50km/h)", "sounds/50kmh.mp3"),
    3: ("Μέγιστη ταχύτητα(60km/h)", "sounds/60kmh.mp3"),
    4: ("Μέγιστη ταχύτητα(70km/h)", "sounds/70kmh.mp3"),
    5:  ("Μέγιστη ταχύτητα(80km/h)", "sounds/80kmh.mp3"),
    6: ("Τέλος ορίου ταχύτητας που έχει επιβληθεί με απαγορευτική πινακίδα (80km/h)", "sounds/telos80.mp3"),
    7: ("Μέγιστη ταχύτητα(100km/h)", "sounds/100kmh.mp3"),
    8: ("Μέγιστη ταχύτητα(120km/h)", "sounds/120kmh.mp3"),
    9: ("Απαγορεύεται το προσπέρασμα των μηχανοκινήτων οχημάτων, πλήν των δίτροχων μοτοσυκλετών χωρίς κάνιστρο", "sounds/passing.mp3"),
    10: ("Απαγορεύεται στους οδηγούς φορτηγών αυτοκινήτων μεγίστου επιτρεπόμενου βάρους που υπερβαίνει τους 3,5 τόννους να προσπερνούν άλλα οχήματα", "sounds/passing3,5.mp3"),
    11: ("Διασταύρωση με οδό, πάνω στην οποία αυτοί που κινούνται ωφείλουν να παραχωρήσουν προτεραιότητα", "sounds/diastavrosiProteraiotita.mp3"),
    12: ("Οδός προτεραιότητας", "sounds/priorityRoad.mp3"),
    13: ("Υποχρεωτική παραχώρηση προτεραιότητας", "sounds/yield.mp3"),
    31: ("Κίνδυνος απο διέλευση άγριων ζώων", "sounds/31.mp3"),
    32: ("Τέλος όλων των τοπικών απαγορεύσεων οι οποίες έχουν επιβληθεί με απαγορευτικές πινακίδες στα κινούμενα οχήματα", "sounds/32.mp3"),
    33: ("Υποχρεωτική κατεύθυνση πορείας με στροφή δεξιά", "sounds/33.mp3"),
    34: ("Υποχρεωτική κατεύθυνση πορείας με στροφή αριστερά", "sounds/34.mp3"),
    35: ("Υποχρεωτική κατεύθυνση πορείας πρός τα εμπρός", "sounds/35.mp3"),
    36: ("Υποχρεωτική κατεύθυνση πορείας εμπρός η δεξιά", "sounds/36.mp3"),
    37: ("Υποχρεωτική κατεύθυνση πορείας εμπρός ή αριστερά", "sounds/37.mp3"),
    38: ("Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου", "sounds/38.mp3"),
    39: ("Υποχρεωτική διέλευση μόνο από την αριστερή πλευρά της νησίδος η του εμποδίου", "sounds/39.mp3"),
    40: ("Κυκλική υποχρεωτική διαδρομή", "sounds/40.mp3"),
    41: ("Τέλος απαγόρευσης προσπεράσματος το οποίο έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/41.mp3"),
    42: ("Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/42.mp3")
}
```

Εικόνα 6.14: Ορισμός Μηνύματος Κλάσης

6.2.8 Εκτύπωση του προβλεπόμενου μηνύματος κλάσης

Αν η πρόβλεψη της κατηγορίας που προκύπτει από το μοντέλο είναι συμπεριλαμβανόμενη στο λεξικό class_messages, τότε το αντίστοιχο μήνυμα και αρχείο ήχου λαμβάνονται και εμφανίζονται στην οθόνη. Επιπλέον, ο ήχος αναπαράγεται χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η πρόβλεψη δεν αντιστοιχεί σε καμία κατηγορία που ορίζεται στο class_messages, τότε εμφανίζεται το μήνυμα "Unknown class".

```
86  # Εκτύπωση του προβλεπόμενου μηνύματος κλάσης
87  if predicted_class in class_messages:
88      class_message, audio_file = class_messages[predicted_class]
89      print("Predicted Class:", class_message)
90      # Load and play audio using pygame
91      pygame.mixer.init()
92      pygame.mixer.music.load(audio_file)
93      pygame.mixer.music.play()
94      while pygame.mixer.music.get_busy():
95          pygame.time.Clock().tick(10)
96  else:
97      print("Unknown class")
98
```

Εικόνα 6 15: Εκτύπωση Μηνύματος

Κεφάλαιο 7: Ανάλυση τεχνολογίας Naive Bayes

Η τεχνολογία Naive Bayes είναι ένας αλγόριθμος μηχανικής μάθησης που βασίζεται στο θεώρημα του Bayes. Αυτός ο αλγόριθμος είναι ευρέως χρησιμοποιούμενος για προβλήματα ταξινόμησης και κατηγοριοποίησης. [20]

Ο αλγόριθμος Naive Bayes κάνει την υπόθεση της "αφελούς ανεξαρτησίας", δηλαδή υποθέτει ότι τα χαρακτηριστικά που περιγράφουν ένα δείγμα είναι ανεξάρτητα μεταξύ τους, δεδομένης της κατηγορίας του δείγματος. Παρόλο που αυτή η υπόθεση σπάνια ισχύει στην πραγματικότητα, η αφελής ανεξαρτησία απλοποιεί τον υπολογισμό και επιτρέπει την αποτελεσματική εκπαίδευση του μοντέλου.[21]

Υπάρχουν διάφορες παραλλαγές του αλγορίθμου Naive Bayes, όπως:

Multinomial Naive Bayes: Χρησιμοποιείται κυρίως για την ταξινόμηση κειμένων με πολλές λέξεις, όπως τα έγγραφα.

Gaussian Naive Bayes: Χρησιμοποιείται όταν τα χαρακτηριστικά ακολουθούν κανονική κατανομή (κανονική κατανομή).

Bernoulli Naive Bayes: Κατάλληλο για δυαδικά δεδομένα (0/1), όπως τα προφίλ χρηστών σε κοινωνικά δίκτυα.

Ένα σημαντικό πλεονέκτημα του Naive Bayes είναι ότι μπορεί να λειτουργήσει αποτελεσματικά ακόμη και με μικρό αριθμό δειγμάτων εκπαίδευσης. Παρά την αφελή υπόθεση ανεξαρτησίας, συχνά παρέχει αξιόλογες προβλέψεις, ειδικά για προβλήματα ταξινόμησης κειμένων όπως τα spam emails, η κατηγοριοποίηση κειμένων και άλλες παρόμοιες εφαρμογές.[21]

7.1 Υλοποίηση του μοντέλου με χρήση τεχνολογίας Naive Bayes

Ο παρακάτω κώδικας εκτελεί μια προκαθορισμένη διαδικασία εκπαίδευσης και αξιολόγησης ενός μοντέλου με χρήση της τεχνολογίας Naive Bayes για την αναγνώριση κατηγοριών εικόνων πινακίδων κυκλοφορίας.

7.1.1 Εισαγωγή των απαραίτητων βιβλιοθηκών

Όπως και στις τεχνολογίες που αναλύθηκαν παραπάνω στα προηγούμενα κεφάλαια, έτσι και για την Naive Bayes κατά την κατασκευή του κώδικα εκπαίδευσης γίνεται εισαγωγή των απαραίτητων και κατάλληλων βιβλιοθηκών. Οι βιβλιοθήκες αυτές είναι οι os, CV2, NumPy, Pickle, train_test_split από sklearn.model_selection, GaussianNB από sklearn.naive_bayes και accuracy_score από sklearn.metrics.

```
1  import os
2  import cv2
3  import numpy as np
4  import pickle
5  from sklearn.model_selection import train_test_split
6  from sklearn.naive_bayes import GaussianNB
7  from sklearn.metrics import accuracy_score
```

Εικόνα 7.1: Εισαγωγή Βιβλιοθηκών

7.1.2 Ορισμός των παραμέτρων των εικόνων

Ορίζονται οι διαστάσεις των εικόνων που θα χρησιμοποιηθούν στον επεξεργασμένο κώδικα. Συγκεκριμένα, καθορίζονται το ύψος και το πλάτος των εικόνων.

```
9  # Ορισμός των παραμέτρων των εικόνων
10 IMG_HEIGHT = 30
11 IMG_WIDTH = 30
```

Εικόνα 7.2: Ορισμός παραμέτρων

7.1.3 Συνάρτηση φόρτωσης και Προ-επεξεργασίας Δεδομένων

Ορίζεται η συνάρτηση load_and_preprocess_data(data_dir), η οποία αναλαμβάνει τη φόρτωση και προ-επεξεργασία των δεδομένων. Οι εικόνες φορτώνονται από τον φάκελο data_dir, μετατρέπονται σε μονοδιάστατα διανύσματα με χρήση της μεθόδου flatten(), και τα αντίστοιχα labels τους αποθηκεύονται. Οι δεδομένα εικόνας και οι ετικέτες μετατρέπονται σε πίνακες της NumPy.

```python
# Συνάρτηση Φόρτωσης και Προ-επεξεργασίας Δεδομένων
1 usage
def load_and_preprocess_data(data_dir):
    image_data = []
    labels = []

    for category in os.listdir(data_dir):
        category_path = os.path.join(data_dir, category)
        for img_file in os.listdir(category_path):
            img_path = os.path.join(category_path, img_file)
            image = cv2.imread(img_path)
            resized_image = cv2.resize(image, dsize: (IMG_HEIGHT, IMG_WIDTH))
            image_data.append(resized_image.flatten())
            labels.append(int(category))

    image_data = np.array(image_data)
    labels = np.array(labels)
    return image_data, labels
```

Εικόνα 7.3: Φόρτωση και Προ-επεξεργασία

7.1.4 Φόρτωση και Προ-επεξεργασία Δεδομένων

Τα δεδομένα φορτώνονται από τον φάκελο data_dir και αποθηκεύονται στις μεταβλητές X (δεδομένα εικόνας) και y (ετικέτες). Έπειτα, το σύνολο δεδομένων διαιρείται σε σύνολα εκπαίδευσης και επικύρωσης με χρήση της συνάρτησης train_test_split.

```python
# Φόρτωση και Προ-επεξεργασία Δεδομένων
data_dir = r"C:\Users\stathis\Desktop\archive\Train"
X, y = load_and_preprocess_data(data_dir)

# Διαχωρισμός δεδομένων σε σύνολα εκπαίδευσης και επικύρωσης
X_train, X_val, y_train, y_val = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
```

Εικόνα 7.4: Φόρτωση και Προ-επεξεργασία

7.1.5 Δημιουργία και Εκπαίδευση Μοντέλου Naive Bayes

Δημιουργείται ένα αντικείμενο της κλάσης GaussianNB, το οποίο αντιπροσωπεύει το μοντέλο Naive Bayes. Το μοντέλο εκπαιδεύεται χρησιμοποιώντας τα δεδομένα εκπαίδευσης (X_train και y_train) με τη μέθοδο fit(). Κατά την εκπαίδευση, το μοντέλο "μαθαίνει" τις πιθανότητες κάθε κλάσης βάσει των χαρακτηριστικών των δεδομένων.

```
38  # Δημιουργία και Εκπαίδευση Μοντέλου Naive Bayes
39  model = GaussianNB()
40  model.fit(X_train, y_train)
```

Εικόνα 7.5: Εκπαίδευση Μοντέλου

7.1.6 Πρόβλεψη Ετικετών για το Σύνολο Επικύρωσης

Χρησιμοποιείται το εκπαιδευμένο μοντέλο για να προβλέψει τις ετικέτες των δεδομένων επικύρωσης (X_val) με τη μέθοδο predict(). Οι προβλεπόμενες ετικέτες αποθηκεύονται στη μεταβλητή y_pred.

```
42  # Πρόβλεψη Ετικετών για το Σύνολο Επικύρωσης
43  y_pred = model.predict(X_val)
```

Εικόνα 7.6: Πρόβλεψη Ετικετών Επικύρωσης

7.1.7 Υπολογισμός Ακρίβειας για την Αξιολόγηση

Υπολογίζεται η ακρίβεια του μοντέλου συγκρίνοντας τις πραγματικές ετικέτες του συνόλου επικύρωσης (y_val) με τις προβλεπόμενες ετικέτες (y_pred). Η ακρίβεια υπολογίζεται με την βοήθεια της συνάρτησης accuracy_score από τη βιβλιοθήκη sklearn.

```
45  # Υπολογισμός Ακρίβειας για την Αξιολόγηση
46  accuracy = accuracy_score(y_val, y_pred)
47  print("Validation Accuracy:", accuracy)
```

Εικόνα 7.7: Υπολογισμός Ακρίβειας

7.1.8 Αποθήκευση του Εκπαιδευμένου Μοντέλου με τη χρήση του Pickle

Το εκπαιδευμένο μοντέλο αποθηκεύεται χρησιμοποιώντας τη βιβλιοθήκη Pickle. Δημιουργείται ένα αρχείο με το όνομα trained_naive_bayes_model.pkl, στο οποίο το μοντέλο αποθηκεύεται με τη χρήση της μεθόδου dump() της κλάσης Pickle. Αυτό το αρχείο περιέχει όλες τις πληροφορίες που απαιτούνται για το εκπαιδευμένο μοντέλο Naive Bayes.

```
49  # Αποθήκευση του Εκπαιδευμένου Μοντέλου με τη χρήση του Pickle
50  model_filename = "trained_naive_bayes_model.pkl"
51  with open(model_filename, 'wb') as model_file:
52      pickle.dump(model, model_file)
53  print("Model saved as", model_filename)
```

Εικόνα 7.8: Αποθήκευση Μοντέλου

7.2 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης Naive Bayes

Ο κώδικας που ακολουθεί χρησιμοποιείται για την αναγνώριση και πρόβλεψη σημάτων οδικής κυκλοφορίας βάσει του προ-εκπαιδευμένου μοντέλου Naive Bayes που δημιουργήθηκε και παρουσιάστηκε στην ενότητα 7.1. Παρακάτω παρουσιάζεται η δομή του κώδικα αναλυτικά.

7.2.1 Εισαγωγή Βιβλιοθηκών

Εισάγονται οι απαραίτητες βιβλιοθήκες για την εκτέλεση του κώδικα. Αυτές περιλαμβάνουν την OpenCV (CV2), την Pickle (pickle), την pygame (pygame), καθώς και την tkinter (tk) για τη δημιουργία γραφικού περιβάλλοντος.

```
1  import cv2
2  import pickle
3  import pygame
4  import tkinter as tk
5  from tkinter import filedialog
```

Εικόνα 7.9: Εισαγωγή Βιβλιοθηκών

7.2.2 Συνάρτηση Προ-επεξεργασίας Εικόνας

Ορίζεται η συνάρτηση preprocess_image(image_path), η οποία φορτώνει μια εικόνα από ένα δοθέν μονοπάτι (image_path). Η εικόνα υποστέγεται σε μία προεπιλεγμένη διάσταση (30x30 pixels) για να χρησιμοποιηθεί στην πρόβλεψη.

```
7   # Συνάρτηση Προ-επεξεργασίας Εικόνας
    1 usage
8   def preprocess_image(image_path):
9       img = cv2.imread(image_path)
10      img = cv2.resize(img, dsize: (30, 30))
11      img = img.flatten()
12      return img
```

Εικόνα 7.10: Προ-επεξεργασία Εικόνας

7.2.3 Φόρτωση του Εκπαιδευμένου Μοντέλου

Το εκπαιδευμένο μοντέλο Naive Bayes φορτώνεται από το αρχείο που δημιουργήθηκε στον προηγούμενο κώδικα. Χρησιμοποιείται η Pickle.load() για να φορτώσει το μοντέλο από το αρχείο trained_naive_bayes_model.pkl

```
14  # Φόρτωση του Εκπαιδευμένου Μοντέλου με χρήση του pickle
15  model_filename = "trained_naive_bayes_model.pkl"
16  with open(model_filename, 'rb') as model_file:
17      loaded_model = pickle.load(model_file)
18  print("Model loaded successfully!")
```

Εικόνα 7.11: Φόρτωση Μοντέλου

7.2.4 Δημιουργία Παραθύρου Tkinter και επιλογή εικόνας

Δημιουργείται ένα παράθυρο Tkinter με τη γραμμή root = tk.Tk(). Το παράθυρο αυτό χρησιμοποιείται για να εμφανίζεται ένα παράθυρο διαλόγου επιλογής αρχείου και επιλέγεται μια εικόνα από τον χρήστη. Αφού ο χρήστης επιλέξει μια εικόνα, το μονοπάτι της εικόνας αποθηκεύεται στη μεταβλητή input_image_path.

```
20  # Δημιουργία Παραθύρου Tkinter και επιλογή εικόνας
21  root = tk.Tk()
22  root.withdraw()
23
24  input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])
```

Εικόνα 7.12: Παράθυρο Tkinter και επιλογή εικόνας

7.2.5 Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα

Γίνεται έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα κατά το προηγούμενο βήμα από το αρχείο. Εάν όντως έχει γίνει επιλογή εικόνας ο κώδικας προχωράει παρακάτω στην πρόβλεψη εικόνας.

```
26  # Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα
27  if input_image_path:
28      print("Selected image:", input_image_path)
```

Εικόνα 7.13: Έλεγχος επιλογής εικόνας

7.2.6 Προ-επεξεργασία και Πρόβλεψη της Εικόνας

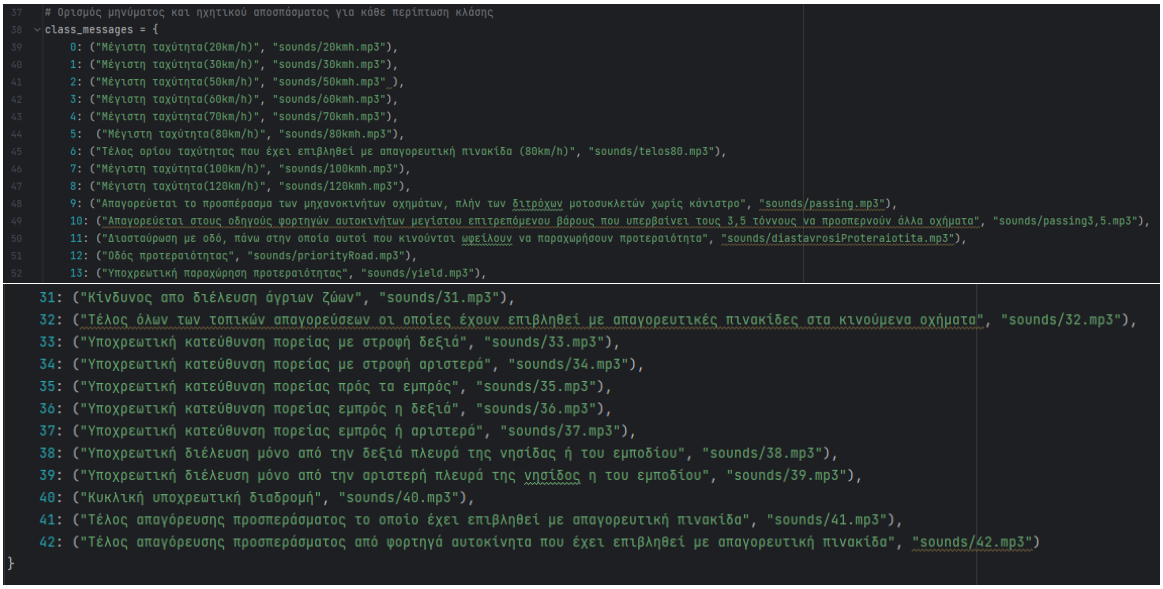
Η εικόνα που επιλέχθηκε προ-επεξεργάζεται και στη συνέχεια προβλέπεται και εκτυπώνεται η κλάση της.

```
31  # Προ-επεξεργασία και δημιουργία Πρόβλεψης της Εικόνας
32  preprocessed_img = preprocess_image(input_image_path)
33  predicted_class = loaded_model.predict([preprocessed_img])[0]  # Get the predicted class
34  print(predicted_class)
```

Εικόνα 7.14: Πρόβλεψη κλάσης

7.2.7 Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης

Κάθε κατηγορία σήμανσης έχει έναν αριθμό κλάσης που την αντιπροσωπεύει. Για παράδειγμα η κλάση 0 αντιστοιχεί στην πινακίδα που επιβάλει όριο ταχύτητας 20 χιλιόμετρα ανά ώρα και περιλαμβάνει το αντίστοιχο μήνυμα και ηχητικό απόσπασμα. Συνολικά έχουμε 43 κλάσεις με την τελευταία να αντιστοιχεί στην πινακίδα "Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα" και στο αντίστοιχο ηχητικό απόσπασμα.



Εικόνα 7.15: Ορισμός μηνύματος κλάσης

7.2.8 Εκτύπωση του προβλεπόμενου μηνύματος κλάσης

Αν η πρόβλεψη της κατηγορίας που προκύπτει από το μοντέλο είναι συμπεριλαμβανόμενη στο λεξικό class_messages, τότε το αντίστοιχο μήνυμα και αρχείο ήχου λαμβάνονται και εμφανίζονται στην οθόνη. Επιπλέον, ο ήχος αναπαράγεται χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η πρόβλεψη δεν αντιστοιχεί σε καμία κατηγορία που ορίζεται στο class_messages, τότε εμφανίζεται το μήνυμα "Unknown class".

```
# Εκτύπωση του προβλεπόμενου μηνύματος κλάσης
if predicted_class in class_messages:
    class_message, audio_file = class_messages[predicted_class]
    print("Predicted Class:", class_message)
    # Load and play audio using pygame
    pygame.mixer.init()
    pygame.mixer.music.load(audio_file)
    pygame.mixer.music.play()
    while pygame.mixer.music.get_busy():
        pygame.time.Clock().tick(10)
else:
    print("Unknown class")
```

Εικόνα 7.16: Εκτύπωση Προβλεπόμενου Μηνύματος

Κεφάλαιο 8: Ανάλυση τεχνολογίας Support Vector Machine – SVM

Το Support Vector Machines (SVMs) είναι ένας ισχυρός αλγόριθμος μηχανικής μάθησης, ευρέως χρησιμοποιούμενος για ταξινόμηση και παλινδρόμηση. Η βασική του ιδέα είναι να βρει ένα υπερ-επίπεδο στον χώρο των χαρακτηριστικών που καλύπτει το μέγιστο δυνατό περιθώριο μεταξύ των κλάσεων των δειγμάτων. Αυτό το υπερ-επίπεδο, που ονομάζεται "όριο απόφασης", μπορεί να επιτρέψει την αποτελεσματική ταξινόμηση των δειγμάτων σε διάφορες κατηγορίες.[22]

Τα βασικά σημεία που χαρακτηρίζουν την τεχνολογία SVM περιλαμβάνουν:

Μέγιστο Περιθώριο (Maximum Margin): Ο στόχος των SVM είναι να βρουν το υπερεπίπεδο που έχει το μέγιστο περιθώριο, δηλαδή τη μεγαλύτερη δυνατή απόσταση μεταξύ των δύο κλάσεων δειγμάτων. Αυτό μειώνει τον κίνδυνο υπερεκπαίδευσης και αυξάνει τη γενίκευση του μοντέλου.

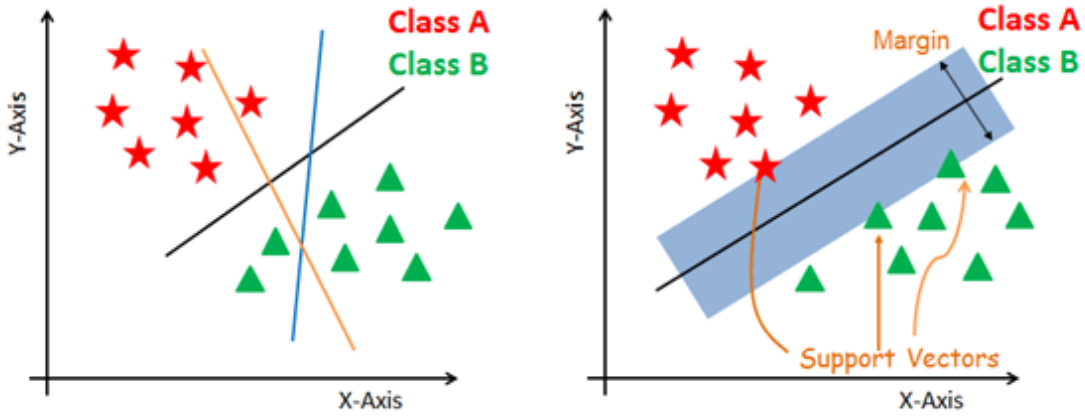
Υποστηρικτικά Διανύσματα (Support Vectors): Τα δείγματα που βρίσκονται πιο κοντά στο όριο απόφασης είναι τα υποστηρικτικά διανύσματα. Αυτά τα δείγματα επηρεάζουν τον υπολογισμό και τη θέση του όριου απόφασης.

Εφαρμογή Πυρήνων (Kernel Tricks): Οι πυρήνες είναι συναρτήσεις που μετασχηματίζουν τα δεδομένα σε έναν χώρο υψηλότερων διαστάσεων, όπου είναι πιο πιθανό να είναι γραμμικά διαχωρίσιμα. Αυτό επιτρέπει την αποτελεσματική χρήση των SVM και σε μη γραμμικά προβλήματα.

Παράμετρος C: Η παράμετρος C ελέγχει την ευελιξία του μοντέλου. Υψηλές τιμές του C επιτρέπουν λιγότερη παραβίαση του περιθωρίου και αυξημένη ακρίβεια εκπαίδευσης, ενώ μικρότερες τιμές δίνουν μεγαλύτερο περιθώριο σφάλματος αλλά μεγαλύτερη πολυπλοκότητα στο μοντέλο.[23]

Οι SVMs είναι ιδιαίτερα χρήσιμες για περίπλοκα προβλήματα ταξινόμησης και παλινδρόμησης, όπως η αναγνώριση εικόνων, η κατηγοριοποίηση κειμένων κ.α.[24]

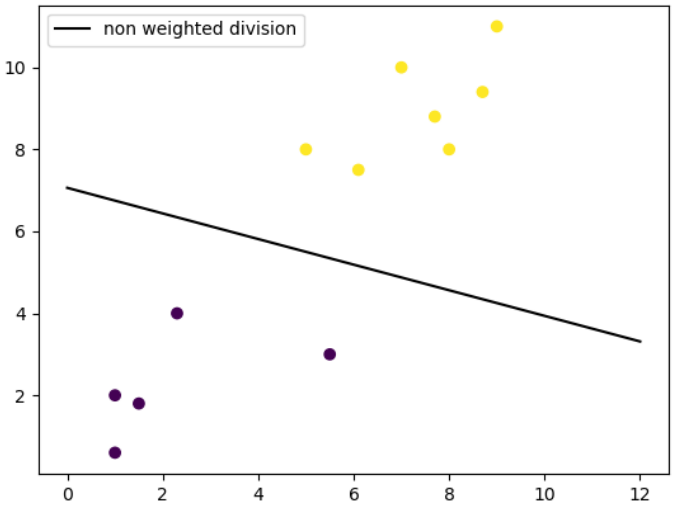
Δημιουργία υπερ-επιπέδων που διαχωρίζουν τις κλάσεις με τον καλύτερο τρόπο. Αριστερό σχήμα που δείχνει τρία υπερεπίπεδα μαύρο, μπλε και πορτοκαλί, το μπλε και το πορτοκαλί έχουν υψηλότερο σφάλμα ταξινόμησης, αλλά το μαύρο διαχωρίζει σωστά τις δύο κλάσεις.



Εικόνα 8.1: SVM

Ένας απλός γραμμικός ταξινομητής SVM λειτουργεί κάνοντας μια ευθεία γραμμή μεταξύ δύο κλάσεων. Αυτό σημαίνει ότι όλα τα σημεία δεδομένων στη μία πλευρά της γραμμής θα αντιπροσωπεύουν μια κατηγορία και τα σημεία δεδομένων στην άλλη πλευρά της γραμμής θα τοποθετηθούν σε μια διαφορετική κατηγορία. Αυτό σημαίνει ότι μπορεί να υπάρχει άπειρος αριθμός γραμμών για να επιλέξετε. Δηλαδή άπειρες κατηγορίες.[24]

Ένα δισδιάστατο παράδειγμα βοηθά να γίνει κατανοητή όλη η ορολογία. Βασικά έχετε κάποια σημεία δεδομένων σε ένα πλέγμα. Προσπαθείτε να διαχωρίσετε αυτά τα σημεία δεδομένων με βάση την κατηγορία στην οποία θα έπρεπε να ανήκουν, αλλά δεν θέλετε να έχετε δεδομένα σε λάθος κατηγορία. Αυτό σημαίνει ότι προσπαθείτε να βρείτε τη γραμμή μεταξύ των δύο πλησιέστερων σημείων που διατηρεί τα άλλα σημεία δεδομένων χωριστά.[25]



Εικόνα 8 2: SVM Category

8.1 Υλοποίηση του μοντέλου με χρήση τεχνολογίας Support Vector Machine – SVM

Ο παρακάτω κώδικας εκτελεί μια προκαθορισμένη διαδικασία εκπαίδευσης και αξιολόγησης ενός μοντέλου με χρήση της τεχνολογίας SVM για την αναγνώριση κατηγοριών εικόνων πινακίδων κυκλοφορίας.

8.1.1 Εισαγωγή των απαραίτητων βιβλιοθηκών

Ο κώδικας ξεκινά με την εισαγωγή των βιβλιοθηκών που θα χρησιμοποιηθούν. Οι βιβλιοθήκες os, CV2, NumPy, Pickle, train_test_split, SVC και accuracy_score εισάγονται για τη διαχείριση του συστήματος, την επεξεργασία εικόνων, τις αριθμητικές διαδικασίες, την εκπαίδευση μοντέλου SVM και τον υπολογισμό ακρίβειας αντίστοιχα.

```
1  import os
2  import cv2
3  import numpy as np
4  import pickle
5  from sklearn.model_selection import train_test_split
6  from sklearn.svm import SVC
7  from sklearn.metrics import accuracy_score
```

Εικόνα 8.3: Εισαγωγή Βιβλιοθηκών

8.1.2 Ορισμός Παραμέτρων Εικόνας

Οι παράμετροι IMG_HEIGHT και IMG_WIDTH καθορίζονται για το ύψος και το πλάτος στα οποία θα μετασχηματιστούν οι εικόνες πριν από την εισαγωγή τους στο μοντέλο.

```
# Ορισμός Παραμέτρων Εικόνας
IMG_HEIGHT = 30
IMG_WIDTH = 30
```

Εικόνα 8.4: Ορισμός Παραμέτρων Εικόνας

8.1.3 Ορισμός Συνάρτησης load_and_preprocess_data

Η συνάρτηση αυτή φορτώνει και προεπεξεργάζεται τα δεδομένα εικόνας από έναν καθορισμένο φάκελο. Δημιουργεί δύο λίστες, μία για τις εικόνες (image_data) και μία για τις ετικέτες (labels). Χρησιμοποιεί τη βιβλιοθήκη OpenCV για να φορτώσει και ανασχεδιάσει κάθε εικόνα.

```
13  # Ορισμός Συνάρτησης load_and_preprocess_data
    1 usage
14  def load_and_preprocess_data(data_dir):
15      image_data = []
16      labels = []
17
18      for category in os.listdir(data_dir):
19          category_path = os.path.join(data_dir, category)
20          for img_file in os.listdir(category_path):
21              img_path = os.path.join(category_path, img_file)
22              image = cv2.imread(img_path)
23              resized_image = cv2.resize(image, dsize: (IMG_HEIGHT, IMG_WIDTH))
24              image_data.append(resized_image.flatten())
25              labels.append(int(category))
26
27      image_data = np.array(image_data)
28      labels = np.array(labels)
29      return image_data, labels
```

Εικόνα 8.5: Προ-επεξεργασία Δεδομένων

8.1.4 Φόρτωση και Προ-επεξεργασία των Δεδομένων

Ορίζεται το data_dir, ο φάκελος που περιέχει τα δεδομένα εικόνας. Καλείται η συνάρτηση load_and_preprocess_data να φορτώσει και προεπεξεργαστεί τις εικόνες. Τα δεδομένα εικόνας και οι ετικέτες αποθηκεύονται στις μεταβλητές X και y.

```
31  # Φόρτωση και Προ-επεξεργασία των Δεδομένων
32  data_dir = r"C:\Users\stathis\Desktop\archive\Train"
33  X, y = load_and_preprocess_data(data_dir)
```

Εικόνα 8.6: Φόρτωση και Προ-επεξεργασία

8.1.5 Διαίρεση των Δεδομένων σε Σύνολα Εκπαίδευσης και Επικύρωσης

Χρησιμοποιείται η συνάρτηση train_test_split για να διαχωρίσει τα δεδομένα σε τέσσερα σύνολα: X_train, X_val, y_train και y_val.

```
# Διαίρεση των Δεδομένων σε Σύνολα Εκπαίδευσης και Επικύρωσης
X_train, X_val, y_train, y_val = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
```

Εικόνα 8.7: Διαχωρισμός σε σύνολα εκπαίδευσης και επικύρωσης

8.1.6 Δημιουργία και Εκπαίδευση του Μοντέλου SVM

Σε αυτήν τη φάση του κώδικα, δημιουργείται και εκπαιδεύεται το μοντέλο Μηχανής Υποστήριξης Διανυσματικών Μηχανών (SVM). Το SVM είναι ένα αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται για την ταξινόμηση και την ανίχνευση προτύπων σε δεδομένα. Για τη Δημιουργία του Μοντέλου έχουμε στη γραμμή model = SVC(kernel='linear', C=1000, random_state=42) δημιουργείται ένα νέο αντικείμενο SVM με τις παραμέτρους που ακολουθούν. kernel='linear': Ορίζει τον τύπο πυρήνα που θα χρησιμοποιηθεί. Στην περίπτωση αυτή, χρησιμοποιείται γραμμικός πυρήνας. C=1000: Η παράμετρος C καθορίζει την ευελιξία του μοντέλου. Το μεγαλύτερο C δίνει περισσότερη έμφαση στην ακρίβεια, αλλά μπορεί να οδηγήσει και σε υπερ-εκπαίδευση. random_state=42: Αυτή η παράμετρος καθορίζει τον τυχαίο σπόρο που χρησιμοποιείται για την αναπαραγωγή των αποτελεσμάτων. Για την Εκπαίδευση του Μοντέλου: Με τη χρήση της μεθόδου fit(X_train, y_train) εκπαιδεύεται το μοντέλο στα δεδομένα εκπαίδευσης (X_train είναι οι εικόνες, και y_train είναι οι αντίστοιχες ετικέτες κατηγοριών).

```
# Δημιουργία και Εκπαίδευση του Μοντέλου SVM
model = SVC(kernel='linear', C=1000, random_state=42)
model.fit(X_train, y_train)
```

Εικόνα 8.8: Εκπαίδευση Μοντέλου

8.1.7 Πρόβλεψη Ετικετών για το Σύνολο Επικύρωσης

Χρησιμοποιώντας το εκπαιδευμένο μοντέλο, προβλέπονται οι ετικέτες για το σύνολο επικύρωσης X_val και αποθηκεύονται στη μεταβλητή y_pred.

```
42  # Πρόβλεψη Ετικετών για το Σύνολο Επικύρωσης
43  y_pred = model.predict(X_val)
```

Εικόνα 8.9: Πρόβλεψη ετικετών επικύρωσης

8.1.8 Υπολογισμός Ακρίβειας για Αξιολόγηση

Χρησιμοποιείται η συνάρτηση accuracy_score για να υπολογιστεί η ακρίβεια του μοντέλου. Αυτή η μέτρηση συγκρίνει τις προβλεπόμενες ετικέτες (y_pred) με τις πραγματικές ετικέτες του συνόλου επικύρωσης (y_val).

```
45  # Υπολογισμός Ακρίβειας για Αξιολόγηση
46  accuracy = accuracy_score(y_val, y_pred)
47  print("Validation Accuracy:", accuracy)
```

Εικόνα 8 10: Υπολογισμός Ακρίβειας

8.1.9 Αποθήκευση του Εκπαιδευμένου Μοντέλου με το Pickle

Το εκπαιδευμένο μοντέλο SVM αποθηκεύεται σε ένα αρχείο με τη χρήση της βιβλιοθήκης Pickle. Αυτό το αρχείο μπορεί να χρησιμοποιηθεί αργότερα για προβλέψεις χωρίς την ανάγκη να εκπαιδευτεί ξανά το μοντέλο.

```
49  # Αποθήκευση του Εκπαιδευμένου Μοντέλου με το Pickle
50  model_filename = "trained_svm_model.pkl"
51  with open(model_filename, 'wb') as model_file:
52      pickle.dump(model, model_file)
53  print("Model saved as", model_filename)
```

Εικόνα 8 11: Αποθήκευση Μοντέλου

8.2 Δημιουργία προβλέψεων για το μοντέλο εκπαίδευσης SVM

Ο κώδικας που ακολουθεί χρησιμοποιείται για την αναγνώριση και πρόβλεψη σημάτων οδικής κυκλοφορίας βάσει του προ-εκπαιδευμένου μοντέλου SVM που δημιουργήθηκε και παρουσιάστηκε στην ενότητα 8.1. Παρακάτω παρουσιάζεται η δομή του κώδικα αναλυτικά.

8.2.1 Εισαγωγή Βιβλιοθηκών

Αρχικά, εισάγονται ορισμένες βιβλιοθήκες, όπως η CV2 για την επεξεργασία εικόνων, η Pickle για τη φόρτωση του μοντέλου SVM, η pygame για την αναπαραγωγή ήχου, και η tkinter για τη δημιουργία γραφικού περιβάλλοντος χρήστη.

```
1  import cv2
2  import pickle
3  import pygame
4  import tkinter as tk
5  from tkinter import filedialog
```

Εικόνα 8.12: Εισαγωγή Βιβλιοθηκών

8.2.2 Συνάρτηση preprocess_image

Ορίζεται μια συνάρτηση με όνομα preprocess_image που δέχεται τη διαδρομή μιας εικόνας ως είσοδο και την προ-επεξεργάζεται. Η εικόνα φορτώνεται με τη χρήση της OpenCV (CV2), στη συνέχεια αλλάζει το μέγεθος της σε 30x30 pixels και επιπλέον επιπεδοποιείται (flatten) για να χρησιμοποιηθεί ως είσοδος για το μοντέλο SVM.

```
7  # Συνάρτηση preprocess_image
   1 usage
8  def preprocess_image(image_path):
9      img = cv2.imread(image_path)
10     img = cv2.resize(img, dsize: (30,30))
11     img = img.flatten()
12     return img
```

Εικόνα 8 13: Προ-επεξεργασία εικόνας

8.2.3 Φόρτωση Εκπαιδευμένου Μοντέλου

Φορτώνεται το εκπαιδευμένο μοντέλο SVM από ένα αρχείο χρησιμοποιώντας την βιβλιοθήκη Pickle.

```
# Φόρτωση Εκπαιδευμένου Μοντέλου
model_filename = "trained_svm_model.pkl"
with open(model_filename, 'rb') as model_file:
    loaded_model = pickle.load(model_file)
print("Model loaded successfully!")
```

Εικόνα 8 14: Φόρτωση Μοντέλου

8.2.4 Δημιουργία Παραθύρου Tkinter και επιλογή εικόνας

Δημιουργείται ένα παράθυρο Tkinter με τη γραμμή root = tk.Tk(). Το παράθυρο αυτό χρησιμοποιείται για να εμφανίζεται ένα παράθυρο διαλόγου επιλογής αρχείου και επιλέγεται μια εικόνα από τον χρήστη. Αφού ο χρήστης επιλέξει μια εικόνα, το μονοπάτι της εικόνας αποθηκεύεται στη μεταβλητή input_image_path.

```
# Δημιουργία Παραθύρου Tkinter και επιλογή εικόνας
root = tk.Tk()
root.withdraw()

input_image_path = filedialog.askopenfilename(title="Select an image", filetypes=[("Image files", "*.jpg *.png")])
```

Εικόνα 8.15: Tkinter Παράθυρο και επιλογή εικόνας

8.2.5 Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα

Γίνεται έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα κατά το προηγούμενο βήμα από το αρχείο. Εάν όντως έχει γίνει επιλογή εικόνας ο κώδικας προχωράει παρακάτω στην πρόβλεψη εικόνας.

```
# Έλεγχος για το εάν έχει επιλεγεί κάποια εικόνα
if input_image_path:
    print("Selected image:", input_image_path)
```

Εικόνα 8.16: Έλεγχος Επιλογής Εικόνας

8.2.6 Προ-επεξεργασία και Πρόβλεψη της Εικόνας

Η εικόνα που επιλέχθηκε προ-επεξεργάζεται και στη συνέχεια προβλέπεται και εκτυπώνεται η κλάση της.

```
# Προ-επεξεργασία και δημιουργία Πρόβλεψης της Εικόνας
preprocessed_img = preprocess_image(input_image_path)
predicted_class = loaded_model.predict([preprocessed_img])[0]  # Get the predicted class
print(predicted_class)
```

Εικόνα 8.17: Πρόβλεψη Κλάσης

8.2.7 Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης

Κάθε κατηγορία σήμανσης έχει έναν αριθμό κλάσης που την αντιπροσωπεύει. Για παράδειγμα η κλάση 0 αντιστοιχεί στην πινακίδα που επιβάλει όριο ταχύτητας 20 χιλιόμετρα ανά ώρα και περιλαμβάνει το αντίστοιχο μήνυμα και ηχητικό απόσπασμα. Συνολικά έχουμε 43 κλάσεις με την τελευταία να αντιστοιχεί στην πινακίδα "Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα" και στο αντίστοιχο ηχητικό απόσπασμα.

```
# Ορισμός μηνύματος και ηχητικού αποσπάσματος για κάθε περίπτωση κλάσης
class_messages = {
    0: ("Μέγιστη ταχύτητα(20km/h)", "sounds/20kmh.mp3"),
    1: ("Μέγιστη ταχύτητα(30km/h)", "sounds/30kmh.mp3"),
    2: ("Μέγιστη ταχύτητα(50km/h)", "sounds/50kmh.mp3"),
    3: ("Μέγιστη ταχύτητα(60km/h)", "sounds/60kmh.mp3"),
    4: ("Μέγιστη ταχύτητα(70km/h)", "sounds/70kmh.mp3"),
    5:  ("Μέγιστη ταχύτητα(80km/h)", "sounds/80kmh.mp3"),
    6: ("Τέλος ορίου ταχύτητας που έχει επιβληθεί με απαγορευτική πινακίδα (80km/h)", "sounds/telos80.mp3"),
    7: ("Μέγιστη ταχύτητα(100km/h)", "sounds/100kmh.mp3"),
    8: ("Μέγιστη ταχύτητα(120km/h)", "sounds/120kmh.mp3"),
    9: ("Απαγορεύεται το προσπέρασμα των μηχανοκινήτων οχημάτων, πλήν των διτρόχων μοτοσυκλετών χωρίς κάνιστρο", "sounds/passing.mp3"),
    10: ("Απαγορεύεται στους οδηγούς φορτηγών αυτοκινήτων μεγίστου επιτρεπόμενου βάρους που υπερβαίνει τους 3,5 τόννους να προσπερνούν άλλα οχήματα", "sounds/passing3,5.mp3"),
    11: ("Διασταύρωση με οδό, πάνω στην οποία αυτοί που κινούνται ωφείλουν να παραχωρήσουν προτεραιότητα", "sounds/diastavrosiProteraiotita.mp3"),
    12: ("Οδός προτεραιότητας", "sounds/priorityRoad.mp3"),
    13: ("Υποχρεωτική παραχώρηση προτεραιότητας", "sounds/yield.mp3"),
    31: ("Κίνδυνος απο διέλευση άγριων ζώων", "sounds/31.mp3"),
    32: ("Τέλος όλων των τοπικών απαγορεύσεων οι οποίες έχουν επιβληθεί με απαγορευτικές πινακίδες στα κινούμενα οχήματα", "sounds/32.mp3"),
    33: ("Υποχρεωτική κατεύθυνση πορείας με στροφή δεξιά", "sounds/33.mp3"),
    34: ("Υποχρεωτική κατεύθυνση πορείας με στροφή αριστερά", "sounds/34.mp3"),
    35: ("Υποχρεωτική κατεύθυνση πορείας πρός τα εμπρός", "sounds/35.mp3"),
    36: ("Υποχρεωτική κατεύθυνση πορείας εμπρός η δεξιά", "sounds/36.mp3"),
    37: ("Υποχρεωτική κατεύθυνση πορείας εμπρός ή αριστερά", "sounds/37.mp3"),
    38: ("Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου", "sounds/38.mp3"),
    39: ("Υποχρεωτική διέλευση μόνο από την αριστερή πλευρά της νησίδας η του εμποδίου", "sounds/39.mp3"),
    40: ("Κυκλική υποχρεωτική διαδρομή", "sounds/40.mp3"),
    41: ("Τέλος απαγόρευσης προσπεράσματος το οποίο έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/41.mp3"),
    42: ("Τέλος απαγόρευσης προσπεράσματος από φορτηγά αυτοκίνητα που έχει επιβληθεί με απαγορευτική πινακίδα", "sounds/42.mp3")
}
```

Εικόνα 8.18: Ορισμός Μηνύματος Κλάσης

8.2.8 Εκτύπωση του προβλεπόμενου μηνύματος κλάσης

Αν η πρόβλεψη της κατηγορίας που προκύπτει από το μοντέλο είναι συμπεριλαμβανόμενη στο λεξικό class_messages, τότε το αντίστοιχο μήνυμα και αρχείο ήχου λαμβάνονται και εμφανίζονται στην οθόνη. Επιπλέον, ο ήχος αναπαράγεται χρησιμοποιώντας τη βιβλιοθήκη pygame. Αν η πρόβλεψη δεν αντιστοιχεί σε καμία κατηγορία που ορίζεται στο class_messages, τότε εμφανίζεται το μήνυμα "Unknown class".

```
86  # Εκτύπωση του προβλεπόμενου μηνύματος κλάσης
87  if predicted_class in class_messages:
88      class_message, audio_file = class_messages[predicted_class]
89      print("Predicted Class:", class_message)
90      # Load and play audio using pygame
91      pygame.mixer.init()
92      pygame.mixer.music.load(audio_file)
93      pygame.mixer.music.play()
94      while pygame.mixer.music.get_busy():
95          pygame.time.Clock().tick(10)
96  else:
97      print("Unknown class")
98
```

Εικόνα 8.19: Εκτύπωση Προβλεπόμενου Μηνύματος

Κεφάλαιο 9: Το Σύνολο των Δεδομένων

Παρακάτω αναλύεται το σύνολο των δεδομένων που χρησιμοποιήθηκε για την υλοποίηση της εργασίας αλλά και κάποιες προκλήσεις που αφορούν την χρήση του.

9.1 Το σύνολο δεδομένων που χρησιμοποιήθηκε

Το σύνολο δεδομένων που έχει χρησιμοποιηθεί για την υλοποίηση της εργασίας είναι το German Traffic Sign Recognition Benchmark (GTSRB). Το σύνολο δεδομένων GTSRB αποτελείται από 43 κατηγορίες πινακίδων κυκλοφορίας και σχεδόν 50.000 εικόνες.

Εικόνα 9.1: Οι 43 κατηγορίες πινακίδων

Οι πινακίδες κυκλοφορίας έχουν προ-κοπεί για εμάς, υπονοώντας ότι οι σχολιαστές/δημιουργοί του συνόλου δεδομένων έχουν επισημάνει χειροκίνητα τις πινακίδες στις εικόνες και έχουν εξάγει την περιοχή ενδιαφέροντος (ROI) των πινακίδων κυκλοφορίας για εμάς, απλοποιώντας έτσι το έργο.[26]

Στον πραγματικό κόσμο, η αναγνώριση πινακίδων κυκλοφορίας είναι μια διαδικασία δύο σταδίων, του εντοπισμού και της αναγνώρισης. Κατά την διαδικασία του εντοπισμού γίνεται εντοπισμός του σημείου όπου σε μια εικόνα/ένα καρέ εισόδου βρίσκεται μια

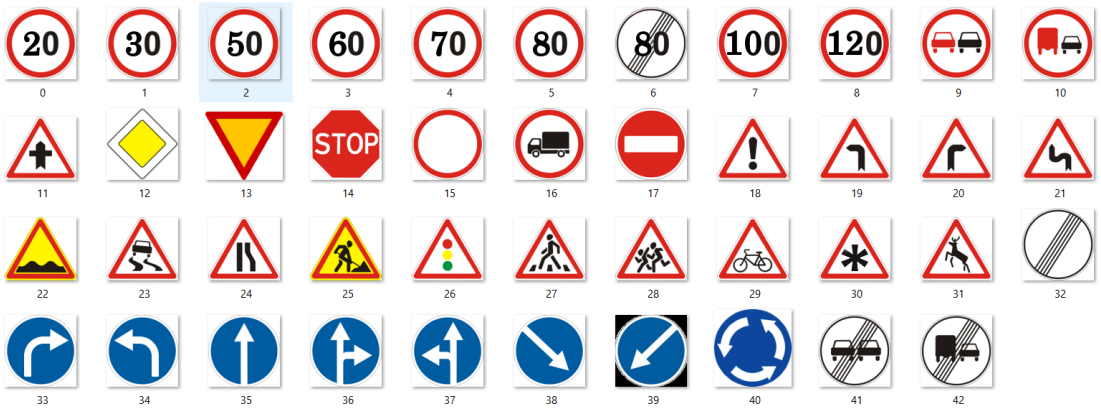
πινακίδα κυκλοφορίας. Κατά την διαδικασία της αναγνώρισης λαμβάνεται η εντοπισμένη περιοχή ενδιαφέροντος (ROI) και στη συνέχεια αναγνωρίζετε και ταξινομείται η πινακίδα κυκλοφορίας.[27]

9.2 Οι προκλήσεις που προκύπτουν με το σύνολο δεδομένων GTSRB

Υπάρχουν ορισμένες προκλήσεις στο σύνολο δεδομένων GTSRB, η πρώτη είναι ότι οι εικόνες είναι χαμηλής ανάλυσης ή ακόμη χειρότερα κάποιες έχουν χαμηλή αντίθεση. Αυτές οι εικόνες είναι pixelated, και σε ορισμένες περιπτώσεις, είναι εξαιρετικά δύσκολο, αν όχι αδύνατο, για το ανθρώπινο μάτι και τον εγκέφαλο να αναγνωρίσουν το σήμα.

Η δεύτερη πρόκληση με το σύνολο δεδομένων είναι ο χειρισμός της διαστρέβλωσης των τάξεων:

Η κορυφαία κατηγορία (Όριο ταχύτητας 50km/h) έχει πάνω από 2.000 παραδείγματα, ενώ η κατηγορία που εκπροσωπείται λιγότερο (Όριο ταχύτητας 20km/h) έχει λιγότερα από 200 παραδείγματα.



Εικόνα 9.2: Τα 43 σήματα της βάσης

Κεφάλαιο 10: Αξιολόγηση απόδοσης κάθε τεχνολογίας

Στο κεφάλαιο αυτό θα αξιολογηθεί η κάθε τεχνολογία που χρησιμοποιήθηκε για την κατασκευή και εκπαίδευση των μοντέλων αναγνώρισης πινακίδων κυκλοφορίας ως προς την απόδοσή τους. Κάθε μοντέλο υποβλήθηκε σε ένα τεστ δοκιμών. Το τεστ αφορά 12 κατηγορίες πινακίδων κυκλοφορίας, οι οποίες αποτελούν τις βασικότερες πινακίδες που συναντάει κανείς κατά πλειοψηφία στο δρόμο στην καθημερινότητα του. Αποτελούνται από την σήμανση "Μέγιστη ταχύτητα(30km/h)", "Μέγιστη ταχύτητα(50km/h)", "Μέγιστη ταχύτητα(60km/h)", "Μέγιστη ταχύτητα(70km/h)", "Μέγιστη ταχύτητα(80km/h)", "Μέγιστη ταχύτητα(120km/h)", "Οδός προτεραιότητας", "Υποχρεωτική παραχώρηση προτεραιότητας", "Υποχρεωτική διακοπή πορείας STOP", "Απαγορεύεται η είσοδος σε όλα τα οχήματα", "Προσοχή άλλοι κίνδυνοι", "Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου". Για κάθε μια από αυτές τις κατηγορίες δόθηκαν για αναγνώριση 10 διαφορετικές λήψεις. Η κάθε κατηγορία αντιστοιχεί σε έναν αριθμό κλάσης:

Μέγιστη ταχύτητα(30km/h): Αντιστοιχεί στην κατηγορία 1

Μέγιστη ταχύτητα(50km/h): Αντιστοιχεί στην κατηγορία 2

Μέγιστη ταχύτητα(60km/h): Αντιστοιχεί στην κατηγορία 3

Μέγιστη ταχύτητα(70km/h): Αντιστοιχεί στην κατηγορία 4

Μέγιστη ταχύτητα(80km/h): Αντιστοιχεί στην κατηγορία 5

Μέγιστη ταχύτητα(120km/h): Αντιστοιχεί στην κατηγορία 8

Οδός προτεραιότητας: Αντιστοιχεί στην κατηγορία 12

Υποχρεωτική παραχώρηση προτεραιότητας: Αντιστοιχεί στην κατηγορία 13

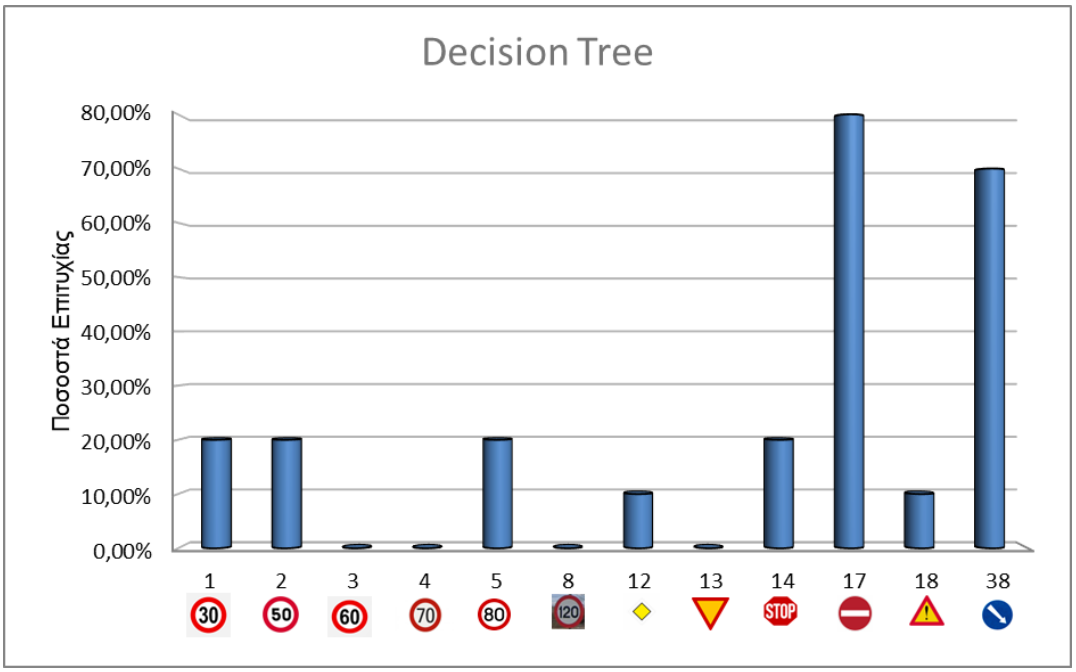
Υποχρεωτική διακοπή πορείας STOP: Αντιστοιχεί στην κατηγορία 14

Απαγορεύεται η είσοδος σε όλα τα οχήματα: Αντιστοιχεί στην κατηγορία 17

Προσοχή άλλοι κίνδυνοι: Αντιστοιχεί στην κατηγορία 18

Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου: Αντιστοιχεί στην κατηγορία 38

10.1 Εκτίμηση Decision Tree Μοντέλου



Εικόνα 10.1: Γράφημα Decision Tree

Decision Tree

ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	ΠΟΣΟΣΤΑ
1	OXI	NAI	OXI	OXI	OXI	NAI	OXI	OXI	OXI	OXI	20,00%
2	OXI	OXI	OXI	OXI	OXI	NAI	NAI	OXI	OXI	OXI	20,00%
3	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
4	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
5	OXI	OXI	OXI	OXI	OXI	NAI	OXI	OXI	NAI	OXI	20,00%
8	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
12	OXI	OXI	OXI	OXI	OXI	OXI	NAI	OXI	OXI	OXI	10,00%
13	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
14	OXI	OXI	NAI	OXI	OXI	OXI	NAI	OXI	OXI	OXI	20,00%
17	NAI	NAI	NAI	OXI	NAI	OXI	NAI	NAI	NAI	NAI	80,00%
18	OXI	OXI	OXI	OXI	NAI	OXI	OXI	OXI	OXI	OXI	10,00%
38	NAI	OXI	NAI	NAI	NAI	OXI	NAI	OXI	NAI	NAI	70,00%

Εικόνα 10.2: Πίνακας Αποτελεσμάτων Decision Tree

Στο παραπάνω διάγραμμα βλέπουμε τα αποτελέσματα των δοκιμών που αφορούν τον κώδικα μοντελοποίησης Decision Tree, με τα μεγαλύτερα ποσοστά επιτυχημένης αναγνώρισης να είναι 80% στην πινακίδα ¨Απαγορεύεται η είσοδος σε όλα τα οχήματα¨ με αριθμό 17 και 70% στην πινακίδα ¨ Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου¨ με αριθμό 38.

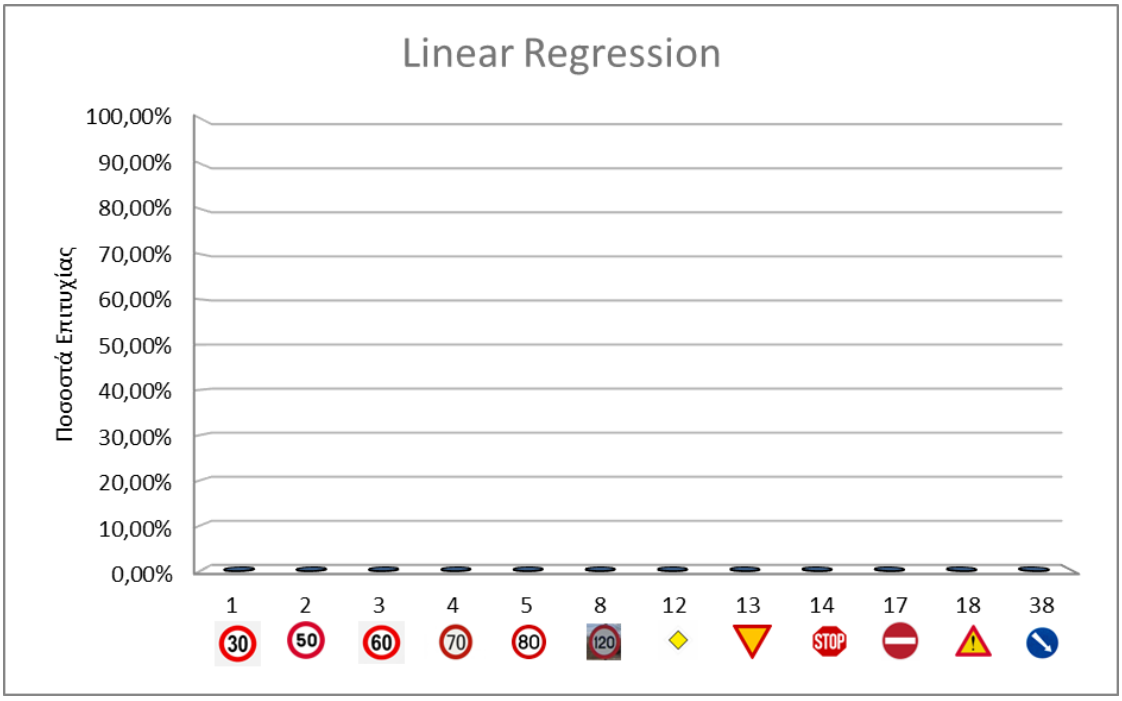
Δεν αναγνωρίστηκαν οι πινακίδες:

- 3, Μεγίστη ταχύτητα (60km/h)
- 4, Μεγίστη ταχύτητα (70km/h)
- 8, Μεγίστη ταχύτητα (120km/h)
- 13, Υποχρεωτική παραχώρηση προτεραιότητας

Ενώ οι πινακίδες με αριθμό 12 (Οδός προτεραιότητας) και 18 (Προσοχή άλλοι κίνδυνοι) έπιασαν το ποσοστό επιτυχίας 10%. Αμέσως μετά με 20% ποσοστό επιτυχής αναγνώρισης είναι οι πινακίδες:

- 1, Μεγίστη ταχύτητα (30km/h)
- 2, Μεγίστη ταχύτητα (50km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 14, Υποχρεωτική διακοπή πορείας STOP

10.2 Εκτίμηση Linear Regression Μοντέλου



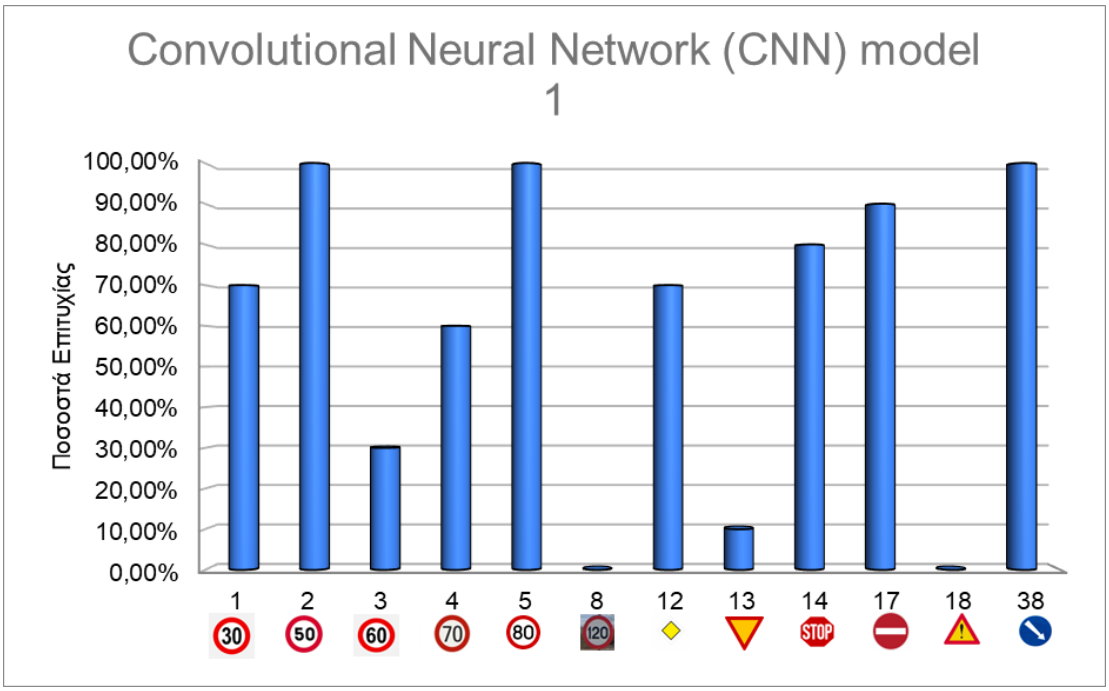
Εικόνα 10.3: Γράφημα Linear Regression

Linear ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
2	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
3	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
4	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
5	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
8	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
12	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
13	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
14	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
17	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
18	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
38	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%

Εικόνα 10.4: Πίνακας Αποτελεσμάτων Linear Regression

Με τον κώδικα μοντελοποίησης Linear Regression, τα αποτελέσματα των δοκιμών μας είναι όλα μηδενικά.

10.3 Εκτίμηση CNN Μοντέλου για ένα πέρασμα από τα δεδομένα εκπαίδευσης



Εικόνα 10.5: Γράφημα CNN 1

CNN1											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	70,00%
2	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	30,00%
4	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	60,00%
5	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
12	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	70,00%
13	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
14	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	80,00%
17	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%
18	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
38	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%

Εικόνα 10.6: Πίνακας Αποτελεσμάτων CNN1

Το παρών CNN μοντέλο έχει εκπαιδευτεί στην βάση δεδομένων μια φόρα. Τα αποτελέσματα είναι αρκετά ικανοποιητικά καθώς έχουμε τρεις κατηγορίες πινακίδων να έχουν ποσοστό επιτυχείς αναγνώρισης 100%. Οι πινακίδες αυτές είναι οι:

- 2, Μεγίστη ταχύτητα (50km/h)
- 5, Μεγίστη ταχύτητα (80km/h)

- 38, Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου

Σειρά στην φθίνουσα κατάταξη έχουν οι πινακίδες 17, Απαγορεύεται η είσοδος σε όλα τα οχήματα, με ποσοστό 90%, η 14, Υποχρεωτική διακοπή πορείας STOP, με 80% και οι πινακίδες 1, Μεγίστη ταχύτητα (30km/h) και 12, Οδός προτεραιότητας , με ποσοστά 70%.

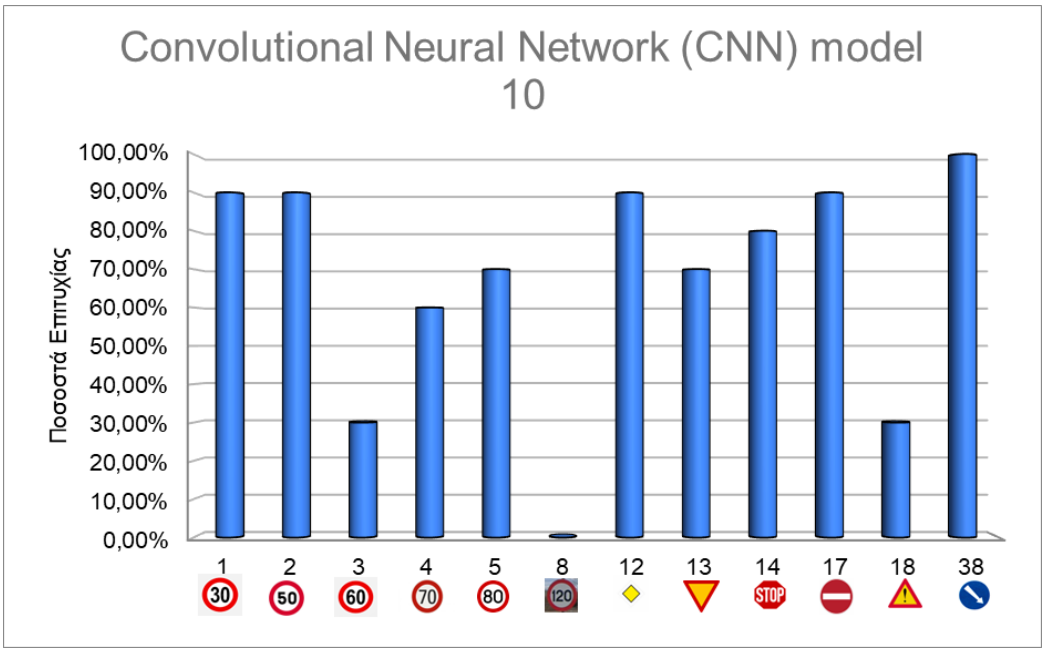
Ακολουθούν οι πινακίδες:

- 4, Μεγίστη ταχύτητα (70km/h)
- 3, Μεγίστη ταχύτητα (60km/h)
- 13, Υποχρεωτική παραχώρηση προτεραιότητας

Με ποσοστά επιτυχίας 60%, 30% και 10% αντίστοιχα.

Με 0% έχουμε δυο πινακίδες, την 8, Μεγίστη ταχύτητα (120km/h) και την 38, Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου.

10.4 Εκτίμηση CNN Μοντέλου για 10 περάσματα από τα δεδομένα εκπαίδευσης



Εικόνα 10.7: Γράφημα CNN 10

CNN10											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	90,00%
2	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	90,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	30,00%
4	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	60,00%
5	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	70,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
12	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	90,00%
13	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	70,00%
14	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	80,00%
17	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%
18	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	30,00%
38	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%

Εικόνα 10.8: Πίνακας Αποτελεσμάτων CNN10

Στην συνεχεία εκπαιδεύσαμε το CNN 10 φορές στην βάση δεδομένων μας και παρατηρήσαμε πως τα αποτελέσματα είναι πιο ομοιόμορφα αν και μειώθηκε το πλήθος της άριστης αναγνώρισης.

Με 100% ποσοστό αναγνώρισης έχουμε την πινακίδα Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου με αριθμό 38.

Με ποσοστό επιτυχίας 90% έχουμε τις πινακίδες:

- 1, Μεγίστη ταχύτητα (30km/h)
- 2, Μεγίστη ταχύτητα (50km/h)
- 12, Οδός προτεραιότητας
- 17, Απαγορεύεται η είσοδος σε όλα τα οχήματα

Στα 80% είναι η πινακίδα 14, Υποχρεωτική διακοπή πορείας STOP και ακολουθούν με 70% οι πινακίδες 5, Μεγίστη ταχύτητα (80km/h), και 13, Υποχρεωτική παραχώρηση προτεραιότητας.

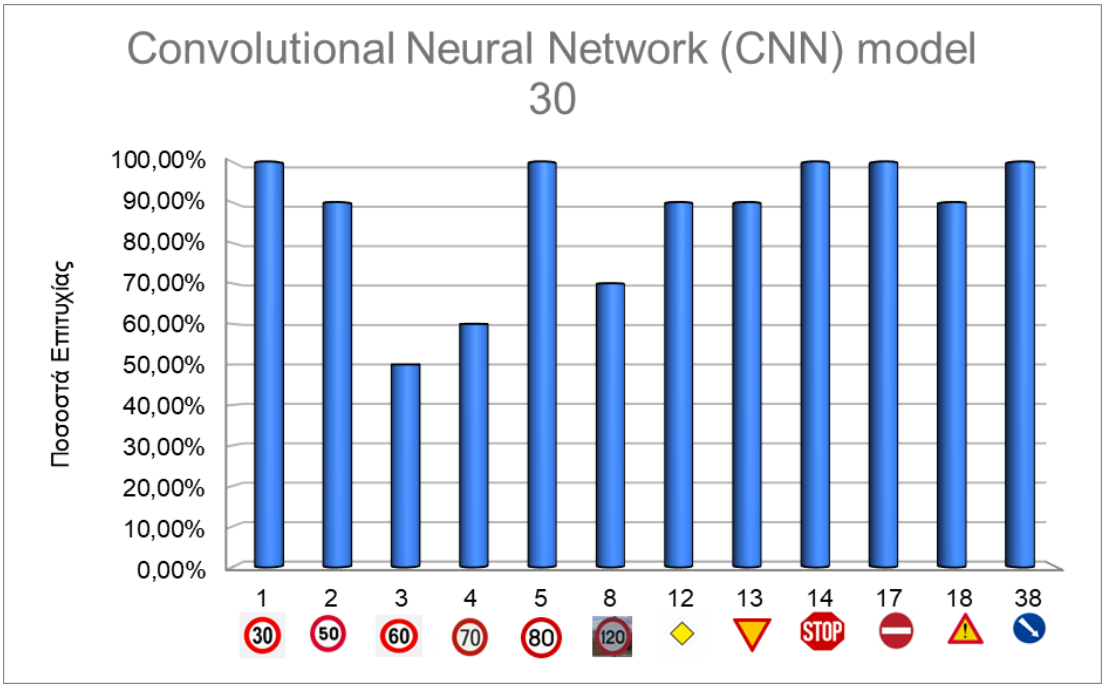
Με ποσοστό 60% είναι η πινακίδα Μεγίστη ταχύτητα (70km/h) με αριθμό 4.

Τα χαμηλότερα ποσοστά έχουν οι πινακίδες:

- 3, Μεγίστη ταχύτητα (60km/h)
- 18, Προσοχή άλλοι κίνδυνοι
- 8, Μεγίστη ταχύτητα (120km/h)

με την 3 και 18 να αγγίζουν το 30% και την 8 το 0%.

10.4 Εκτίμηση CNN Μοντέλου για 30 περάσματα από τα δεδομένα εκπαίδευσης



Εικόνα 10.9: Γράφημα CNN 30

CNN30											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
2	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	90,00%
3	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	50,00%
4	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	60,00%
5	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
8	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	70,00%
12	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%
13	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%
14	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
17	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
18	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	90,00%
38	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%

Εικόνα 10.10: Πίνακας Αποτελεσμάτων CNN30

Το μοντέλο που εκπαιδεύτηκε 30 φορές γύρω από τα δεδομένα μας φαίνεται να έχει τα πιο θετικά και καλά μοιρασμένα ποσοστά στις δοκιμασίες μας.

Με 100% επιτυχείς αναγνωρίσεις είναι οι πινακίδες:

- 1, Μεγίστη ταχύτητα (30km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 14, Υποχρεωτική διακοπή πορείας STOP

- 17, Απαγορεύεται η είσοδος σε όλα τα οχήματα
- 38, Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου

Στο 90% έχουμε τις:

- 2, Μεγίστη ταχύτητα (50km/h)
- 12, Οδός προτεραιότητας
- 13, Υποχρεωτική παραχώρηση προτεραιότητας
- 18, Προσοχή άλλοι κίνδυνοι

Τα χαμηλότερα ποσοστά έχουν οι πινακίδες:

- 8, Μεγίστη ταχύτητα (120km/h)
- 4, Μεγίστη ταχύτητα (70km/h)
- 3, Μεγίστη ταχύτητα (60km/h)

Φτάνοντας τα ποσοστά 70%, 60% και 50% αντίστοιχα.

10.5 Εκτίμηση Logistic Regression Μοντέλου για Max_iter = 10

Η παράμετρος max iter καθορίζει το πόσες φορές θα περάσει ο αλγόριθμος τα δεδομένα εκπαίδευσης. Λειτουργεί δηλαδή με την ίδια λογική που είδαμε παραπάνω στο μοντέλο CNN.

Εικόνα 10.11: Γράφημα Logistic Regression 10

Logistic1											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	10,00%
2	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
4	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
5	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
12	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%
13	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	30,00%
14	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
17	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	100,00%
18	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
38	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%

Εικόνα 10.12: Πίνακας Αποτελεσμάτων Logistic Regression 10

Το μοντέλο αυτό είναι εκπαιδευόμενο δέκα φόρες γύρω από τα δεδομένα μας και τα αποτελέσματα είναι αποθαρρυντικά. Έχουμε μια πινακίδα με 100% ποσοστό επιτυχίας με αριθμό 17 και δυο με 90% με τους αριθμούς 12 και 38.

Με 30% έχουμε την πινακίδα Υποχρεωτική παραχώρηση προτεραιότητας . Οι πινακίδες 1, Μεγίστη ταχύτητα (30km/h),και 14, Υποχρεωτική διακοπή πορείας STOP, έχουν ποσοστό επιτυχίας 10%.

Με 0% ποσοστό επιτυχείς αναγνώρισης είναι οι πινακίδες:

- 2, Μεγίστη ταχύτητα (50km/h)
- 3, Μεγίστη ταχύτητα (60km/h)
- 4, Μεγίστη ταχύτητα (70km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 8, Μεγίστη ταχύτητα (120km/h)
- 18, Προσοχή άλλοι κίνδυνοι

10.6 Εκτίμηση Logistic Regression Μοντέλου για Max_iter = 30

Εικόνα 10.14: Γράφημα Logistic Regression 30

Logistic3											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	20,00%
2	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	40,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
4	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	10,00%
5	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
12	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	40,00%
13	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	20,00%
14	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	50,00%
17	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	80,00%
18	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
38	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%

Εικόνα 10.13: Πίνακας αποτελεσμάτων Logistic Regression 30

Μετά από 30 εκπαιδεύσεις έχουμε την 38 με ποσοστό 90% και την 17 με 80%.

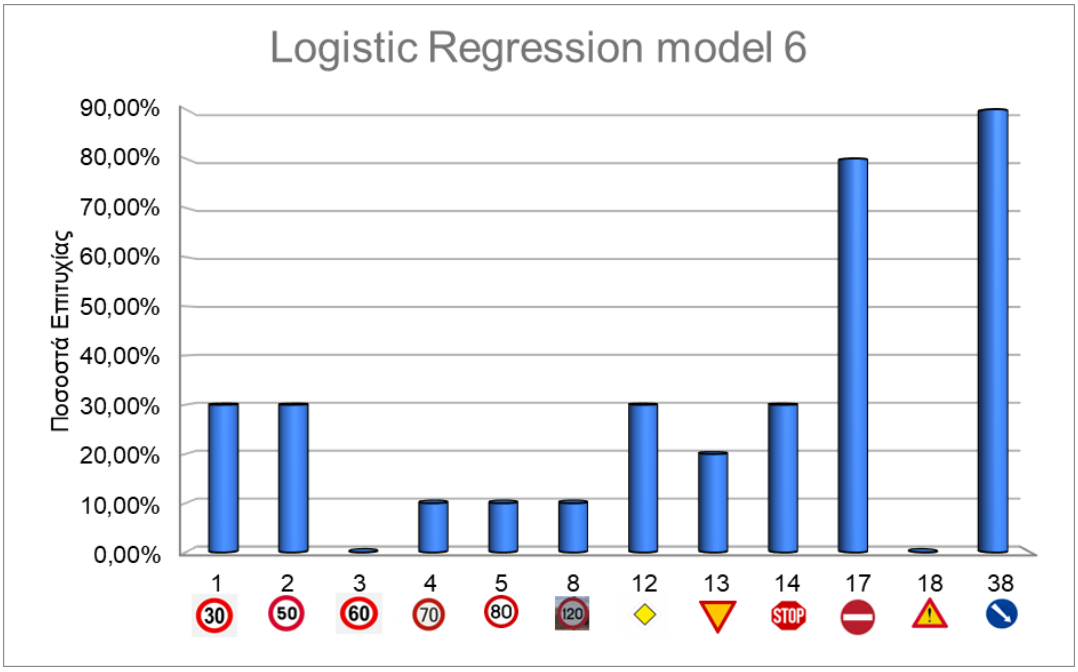
Στο 50% είναι η πινακίδα Υποχρεωτική διακοπή πορείας STOP με αριθμό 14 και ακολουθούν με 40% οι πινακίδες Οδός προτεραιότητας και Μεγίστη ταχύτητα (50km/h).

Η πινακίδα Μεγίστη ταχύτητα (70km/h) έχει ποσοστό 10% και οι πινακίδες:

- 3, Μεγίστη ταχύτητα (60km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 8, Μεγίστη ταχύτητα (120km/h)
- 18, Προσοχή άλλοι κίνδυνοι

Έχουν μηδενικό ποσοστό αναγνώρισης.

10.7 Εκτίμηση Logistic Regression Μοντέλου για Max_iter = 60



Εικόνα 10.15: Γράφημα Logistic Regression 60

Logistic6											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	30,00%
2	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	30,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
4	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	10,00%
5	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	10,00%
12	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	30,00%
13	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	20,00%
14	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	30,00%
17	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	80,00%
18	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
38	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	90,00%

Εικόνα 10.16: Πίνακας Αποτελεσμάτων Logistic Regression 60

Στις 60 εκπαιδεύσεις έχουμε δυο πινακίδες με μηδενικό ποσοστό αναγνώρισης, τις πινακίδες με αριθμό 3 και 18.

Στο 10% είναι οι:

- 4, Μεγίστη ταχύτητα (70km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 8, Μεγίστη ταχύτητα (120km/h)

Και με 20% επιτυχία είναι η πινακίδα Υποχρεωτική παραχώρηση προτεραιότητας.

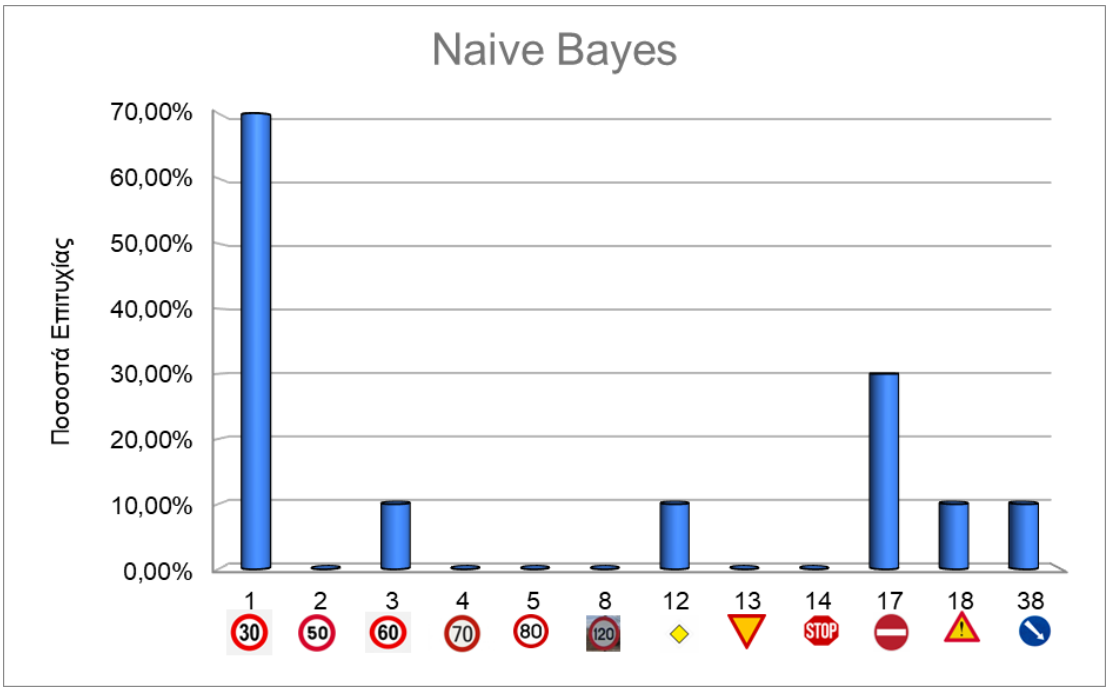
Τέσσερις πινακίδες αγγίζουν το 30%, οι:

- 1, Μεγίστη ταχύτητα (30km/h)
- 2, Μεγίστη ταχύτητα (50km/h)

- 12, Οδός προτεραιότητας
- 14, Υποχρεωτική διακοπή πορείας STOP

Στο 80% είναι η πινακίδα 17, Απαγορεύεται η είσοδος σε όλα τα οχήματα, και στο 90% η 38, Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου.

10.8 Εκτίμηση Naive Bayes Μοντέλου



Εικόνα 10.18: Γράφημα Naive Bayes

Naive Bayes											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	70,00%
2	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
3	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
4	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
5	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
8	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
12	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
13	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
14	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	0,00%
17	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	30,00%
18	ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%
38	ΝΑΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	10,00%

Εικόνα 10.17: Πίνακας Αποτελεσμάτων Naive Bayes

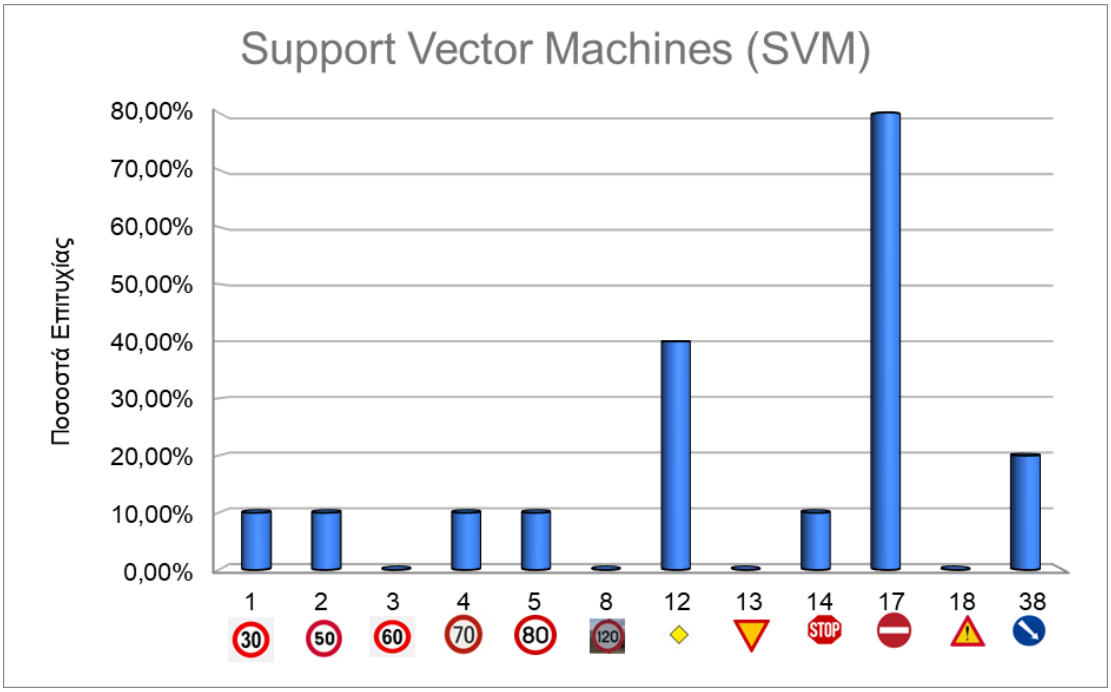
Στον κώδικα μοντελοποίησης Naive Bayes οι μισές κατηγορίες πινακίδων από τις δόκιμες μας έχουν 0% ποσοστό επιτυχίας. Οι κατηγορίες αυτές είναι:

- 2, Μεγίστη ταχύτητα (50km/h)

- 4, Μεγίστη ταχύτητα (70km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 8, Μεγίστη ταχύτητα (120km/h)
- 13, Υποχρεωτική παραχώρηση προτεραιότητας
- 14, Υποχρεωτική διακοπή πορείας STOP

Αμέσως μετά, με ποσοστό 10%, είναι οι κατηγορίες 3, 12, 18 και 38. Η πινακίδα Απαγορεύεται η είσοδος σε όλα τα οχήματα αγγίζει το 30% και το μεγαλύτερο ποσοστό των δοκιμασιών έχει η πινακίδα Μεγίστη ταχύτητα (30km/h) με αριθμό 1 και ποσοστό επιτυχίας 70%.

10.9 Εκτίμηση SVM Μοντέλου



Εικόνα 10 19: Γράφημα SVM

SVM											
ΠΙΝΑΚΙΔΕΣ	0η ΠΡΟΣΠΑΘΕΙΑ	1η ΠΡΟΣΠΑΘΕΙΑ	2η ΠΡΟΣΠΑΘΕΙΑ	3η ΠΡΟΣΠΑΘΕΙΑ	4η ΠΡΟΣΠΑΘΕΙΑ	5η ΠΡΟΣΠΑΘΕΙΑ	6η ΠΡΟΣΠΑΘΕΙΑ	7η ΠΡΟΣΠΑΘΕΙΑ	8η ΠΡΟΣΠΑΘΕΙΑ	9η ΠΡΟΣΠΑΘΕΙΑ	
1	OXI	OXI	NAI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	10,00%
2	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	NAI	10,00%
3	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
4	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	NAI	OXI	10,00%
5	OXI	OXI	OXI	OXI	NAI	OXI	OXI	OXI	OXI	OXI	10,00%
8	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
12	NAI	NAI	OXI	NAI	OXI	OXI	NAI	OXI	OXI	OXI	40,00%
13	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
14	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	NAI	OXI	10,00%
17	NAI	OXI	NAI	OXI	NAI	NAI	NAI	NAI	NAI	NAI	80,00%
18	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	OXI	0,00%
38	OXI	OXI	OXI	OXI	OXI	OXI	NAI	OXI	OXI	NAI	20,00%

Εικόνα 10.20: Πίνακας Αποτελεσμάτων SVM

Στο SVM έχουμε 4 πινακίδες με μηδενικό ποσοστό επιτυχείς αναγνώρισης. Οι πινακίδες αυτές είναι οι:

- 3, Μεγίστη ταχύτητα (60km/h)
- 8, Μεγίστη ταχύτητα (120km/h)
- 13, Υποχρεωτική παραχώρηση προτεραιότητας
- 18, Προσοχή άλλοι κίνδυνοι

Στην συνεχεία με 10% είναι οι:

- 1, Μεγίστη ταχύτητα (30km/h)
- 2, Μεγίστη ταχύτητα (50km/h)
- 4, Μεγίστη ταχύτητα (70km/h)
- 5, Μεγίστη ταχύτητα (80km/h)
- 14, Υποχρεωτική διακοπή πορείας STOP

Και με 20% ακολουθεί η Υποχρεωτική διέλευση μόνο από την δεξιά πλευρά της νησίδας ή του εμποδίου.

Οι δυο πινακίδες που υπολείπονται είναι αυτές που έχουν το μεγαλύτερα ποσοστά των δοκιμών. Η πινακίδα Οδός προτεραιότητας έχει 40% και η 17, Απαγορεύεται η είσοδος σε όλα τα οχήματα, φτάνει στα 70%.

Κεφάλαιο 11: Συμπεράσματα – Προτάσεις

11.1 Συμπεράσματα

Η αναγνώριση πινακίδων του κώδικα οδικής κυκλοφορίας αποτελεί ένα σημαντικό κομμάτι της οδικής ασφάλειας και προάγει την τήρηση των κανόνων κυκλοφορίας για την προστασία της ανθρώπινης ζωής και περιουσίας. Η εξέλιξη της μηχανικής μάθησης και επεξεργασίας δεδομένων ανοίγει νέους ορίζοντες στον τομέα αυτόν και διευκολύνει την ανάπτυξη αποτελεσματικών λύσεων.

Στην παρούσα πτυχιακή εργασία, πραγματοποιήσαμε μια προσέγγιση μελετώντας, αναπτύσσοντας και τροποποιώντας διάφορους αλγορίθμους μηχανικής μάθησης για την αναγνώριση πινακίδων κυκλοφορίας. Η κατάλληλη επιλογή των αλγορίθμων και η προσαρμογή τους στο συγκεκριμένο πρόβλημα ήταν κρίσιμης σημασίας για την επίτευξη προβλέψεων - αποτελεσμάτων.

Στη συνέχεια, προχωρήσαμε σε μια σειρά δοκιμών και συγκρίσεων των αλγορίθμων προκειμένου να αξιολογήσουμε την απόδοσή τους σε διάφορες συνθήκες και περιπτώσεις. Η σύγκριση αυτή μας βοήθησε να προσδιορίσουμε τον πλέον αποτελεσματικό αλγόριθμο για τον συγκεκριμένο σκοπό και να βελτιώσουμε την ακρίβεια της αναγνώρισης. Όπως καταλαβαίνουμε και από τα αποτελέσματα των παραπάνω δοκιμών ο πιο αποδοτικός αλγόριθμος είναι αυτός που κάνει χρήση της τεχνολογίας CNN. Αποδήχθηκε ο πιο αποτελεσματικός μεταξύ των άλλων 5 καθώς κατάφερε να αναγνωρίσει επιτυχώς 104 δείγματα από τα 120 που του δώθηκαν. Αυτό τον καθιστά τον αποτελεσματικότερο αλγόριθμο για χρήση σε τέτοιου είδους εφαρμογές. Στην $2^{η}$ θέση με σημαντικά μικρότερη όμως απόδοση έρχεται ο αλγόριθμος Logistic Regression. Κατάφερε να αναγνωρίσει 34 από τα 120 σήματα. Στην $3^{η,}$ $4^{η}$ και $5^{η}$ θέση έρχονται οι αλγόριθμοι Decision Tree, SVM και Naive Bayes ενώ στην τελευταία και $6^{η}$ θέση έρχεται ο αλγόριθμος Linear Regression με μηδενικά ποσοστά επιτυχίας, κάτι που τον καθιστά τελείως ανήκανο για αναγώριση σημάτων οδικής κυκλοφορίας.

11.2 Μελοντικές βελτιώσεις σε υπάρχουσες προκλήσεις

Ωστόσο, αναγνωρίζουμε ότι οι τεχνολογίες αυτές εξακολουθούν να αντιμετωπίζουν προκλήσεις, όπως η αντιμετώπιση ποικίλων περιβαλλοντικών συνθηκών, η ευκρίνεια των εικόνων προς επεξεργασία, οι παρεμβολές που υπάρχουν σε κάποιες σημάνσεις π.χ. κάποιο graffiti ή αυτοκόλλητο, οι συνθήκες φωτισμού κ.α. εμποδίζουν την ορθή αναγνώριση τους. Ενδεχομένως, στο μέλλον, η συνεχή έρευνα και ανάπτυξη θα μπορούσε να βελτιώσει τις λειτουργίες της εφαρμογής μας και να αυξήσει την προσαρμοστικότητα του συστήματος. Επίσης, θα ήταν ενδιαφέρον να επεκτείνουμε το σύστημα αναγνώρισης πινακίδων του αποδοτικότερου αλγορίθμου (CNN) σε πιο προηγμένες τεχνολογίες, όπως για παράδειγμα με τη χρήση κάποιας συσκευής εγγραφής δεδομένων σε κάποιο όχημα κατά την κίνηση του προκειμένου να έχουμε προβλέψεις πραγματικού χρόνου και να αντιμετωπίσουμε ακόμα πιο πολύπλοκες και απαιτητικές προκλήσεις.

Συνοψίζοντας, η παρούσα πτυχιακή εργασία μας επέτρεψε να εξερευνήσουμε και να αξιοποιήσουμε τις δυνατότητες της μηχανικής μάθησης στον τομέα της αναγνώρισης πινακίδων κυκλοφορίας. Η επίτευξη υψηλής ακρίβειας αποτελεσμάτων στην αναγνώριση αυτών των πινακίδων με χρήση κάποιων τεχνολογιών που δοκιμάσαμε αποτελεί σημαντικό βήμα προς την προαγωγή της οδικής ασφάλειας και τη βελτίωση της κυκλοφορίας. Με τη συνεχή έρευνα και ανάπτυξη, αναμένουμε ότι οι τεχνολογίες αυτές θα συνεχίσουν να εξελίσσονται και να συμβάλλουν στη δημιουργία ενός ασφαλέστερου και βιώσιμου οδικού περιβάλλοντος για όλους.

Κεφάλαιο 12: Βιβλιογραφία

1.Τσάγκα Ε, Λάτσινος Α, Πατρής Γ, Αλεξάκης Ι. Κώδικας Οδικής Κυκλοφορίας (Κ.Ο.Κ.).; 2007. Accessed August 24, 2023. https://www.ioas.gr/uploads/docs/2016/05/397.pdf

2. Murphy K. *Machine Learning a Probabilistic Perspective*.; 2012. Accessed June 2023. https://static.googleusercontent.com/media/research.google.com/el//pubs/archive/38136.pdf

3. Domingos P. A few useful things to know about machine learning. *Communications of the ACM*. 2012;55(10):78. doi:https://doi.org/10.1145/2347736.2347755

4. Sagar V, Nanjundeswaraswamy T. *ARTIFICIAL INTELLIGENCE in AUTONOMOUS VEHICLES -A LITERATURE REVIEW By*.; 2019.

5. Albawi S, Bayat O, Al-Azawi S, Ucan ON. Social Touch Gesture Recognition Using Convolutional Neural Network. *Computational Intelligence and Neuroscience*. 2018;2018:1-10. doi:https://doi.org/10.1155/2018/6973103

6. Deepanshi. Convolutional Neural Network with Implementation in Python. Analytics Vidhya. Published August 14, 2021. Accessed June 2023. https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python/

7. Sharma A. Convolutional Neural Networks in Python. www.datacamp.com. Published December 2017. Accessed May 2023. https://www.datacamp.com/tutorial/convolutional-neural-networks-python

8. matlab. What Is a Convolutional Neural Network? | 3 things you need to know. ch.mathworks.com. Published 2014. Accessed May 2023. https://ch.mathworks.com/discovery/convolutional-neural-network-matlab.html

9. GeeksforGeeks. Introduction to Convolution Neural Network. GeeksforGeeks. Published August 21, 2017. Accessed May 2023. https://www.geeksforgeeks.org/introduction-convolution-neural-network/

10. Keras. Keras documentation: About Keras. keras.io. Published 2021. Accessed May 2023. https://keras.io/about/

11. SimpleLearn. What is TensorFlow 2.0 [The Best Guide to Understand TensorFlow 2.0]. Simplilearn.com. Published April 2023. Accessed July 24, 2023. https://www.simplilearn.com/tutorials/deep-learning-tutorial/tensorflow-2#what_is_tensorflow_20

12. Sayak P. Essentials of Linear Regression in Python. Datacamp. Published October 2022. Accessed July 2023. https://www.datacamp.com/tutorial/essentials-linear-regression-python

13. w3schools. Python Machine Learning Linear Regression. www.w3schools.com. Accessed June 2023. https://www.w3schools.com/python/python_ml_linear_regression.asp

14. IBM. About Linear Regression | IBM. www.ibm.com. Published 2022. Accessed June 2023. https://www.ibm.com/topics/linear-regression

15. Navlani A. Python Decision Tree Classification Tutorial: Scikit-Learn DecisionTreeClassifier. www.datacamp.com. Published February 2023. Accessed July 2023. https://www.datacamp.com/tutorial/decision-tree-classification-python

16. scikit-learn. 1.10. Decision Trees — scikit-learn 0.22 documentation. Scikit-learn.org. Published 2009. Accessed July 2023. https://scikit-learn.org/stable/modules/tree.html

17. Turing. The Importance of Decision Trees in Machine Learning. www.turing.com. Accessed July 2023. https://www.turing.com/kb/importance-of-decision-trees-in-machine-learning

18. Navlani A. Python Logistic Regression Tutorial with Sklearn & Scikit. www.datacamp.com. Published December 2019. Accessed July 2023. https://www.datacamp.com/tutorial/understanding-logistic-regression-python

19. Alzen JL, Langdon LS, Otero VK. A logistic regression investigation of the relationship between the Learning Assistant model and failure rates in introductory STEM courses. *International Journal of STEM Education*. 2018;5(1). doi:https://doi.org/10.1186/s40594-018-0152-1

20. DataCamp. Naive Bayes Classifier Tutorial: with Python Scikit-learn. www.datacamp.com. Accessed July 2023. https://www.datacamp.com/tutorial/naive-bayes-scikit-learn

21. Zhang Z. Naive Bayes Explained. Medium. Published August 14, 2019. Accessed July 2023. https://towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0

22. Boswell D. *Introduction to Support Vector Machines*.; 2002. Accessed August 2023. https://home.work.caltech.edu/~boswell/IntroToSVM.pdf

23. Fletcher T. *Support Vector Machines Explained*.; 2008.

24. DataCamp. Scikit-learn SVM Tutorial with Python (Support Vector Machines). www.datacamp.com. https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python

25. McGregor M. SVM Machine Learning Tutorial – What is the Support Vector Machine Algorithm, Explained with Code Examples. freeCodeCamp.org. Published July 1, 2020. Accessed August 2023. https://www.freecodecamp.org/news/svm-machine-learning-tutorial-what-is-the-support-vector-machine-algorithm-explained-with-code-examples/

26. Rosebrock A. Traffic Sign Classification with Keras and Deep Learning. PyImageSearch. Published November 4, 2019. Accessed April 2023. https://pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning/

27. Hu X, Petrelli A, Matthias M, Houben S, De Souza A. German Traffic Sign Benchmarks. benchmark.ini.rub.de. Published February 2013. https://benchmark.ini.rub.de/