ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Μοντέλο για τον υπολογισμό δείκτη ασφαλούς οδηγικής συμπεριφοράς από δεδομένα οδήγησης

Πτυχιακή εργασία

Δημήτρης Κνέκνας

Αθήνα, 2024

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATION AND TELEMATICS

Model for calculating safe driving behavior index from driving data

Bachelor Thesis

Dimitris Kneknas

Athens, 2024

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

ΒΑΡΛΑΜΗΣ ΗΡΑΚΛΗΣ
Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

ΤΣΕΡΠΕΣ ΚΩΝΣΤΑΝΤΙΝΟΣ
Αναπληρωτής Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

ΔΗΜΗΤΡΑΚΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ
Αναπληρωτής Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο

Ο Δημήτρης Κνέκνας δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

3) Όπου υφίστανται δικαιώματα άλλων δημιουργών έχουν διασφαλιστεί όλες οι αναγκαίες άδειες χρήσης ενώ το αντίστοιχο υλικό είναι ευδιάκριτο στην υποβληθείσα εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου σε όλους όσους συνέβαλαν στην ολοκλήρωση αυτής της πτυχιακής εργασίας.

Πρώτα και κύρια, ευχαριστώ τον επιβλέποντα καθηγητή μου, κ. Ηρακλή Βαρλάμη, για την αδιάλειπτη υποστήριξη, την καθοδήγηση και τις συμβουλές του καθ' όλη τη διάρκεια της έρευνας. Η εμπειρία και η γνώση του υπήρξαν καθοριστικές για την ολοκλήρωση αυτής της εργασίας.

Ευχαριστώ επίσης τα μέλη της επιτροπής μου για το χρόνο και την προσοχή τους στην αξιολόγηση της πτυχιακής μου εργασίας, καθώς και για τα χρήσιμα σχόλια και τις παρατηρήσεις τους.

Επιπλέον, ευχαριστώ την οικογένειά μου και τους φίλους μου για την υποστήριξη και την κατανόησή τους κατά τη διάρκεια των σπουδών μου. Η ενθάρρυνσή τους μου έδωσε δύναμη να συνεχίσω ακόμα και στις πιο δύσκολες στιγμές.

Τέλος, ευχαριστώ όλους όσους συνέβαλαν με οποιονδήποτε τρόπο στην ολοκλήρωση αυτής της πτυχιακής εργασίας. Η συμβολή τους υπήρξε πολύτιμη και εκτιμώ βαθύτατα την υποστήριξή τους.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη στα Ελληνικά

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη ενός καινοτόμου μοντέλου για τον υπολογισμό του δείκτη ασφαλούς οδηγικής συμπεριφοράς, αξιοποιώντας δεδομένα που συλλέγονται κατά την οδήγηση σε πραγματικό χρόνο. Η ανάγκη για βελτιωμένα προγνωστικά μοντέλα προέκυψε από τις αυξημένες δυνατότητες συλλογής δεδομένων μέσω σύγχρονων τεχνολογιών, όπως οι συσκευές με GPS και τα έξυπνα οχήματα, καθώς και την ανάγκη βελτίωσης της οδικής ασφάλειας και της οδηγικής εμπειρίας υπό διαφορετικές περιβαλλοντικές συνθήκες. Παρά τη διαθεσιμότητα μεγάλων συνόλων δεδομένων, η ενσωμάτωση και ανάλυσή τους, σε συνδυασμό με περιβαλλοντικούς παράγοντες, όπως οι καιρικές συνθήκες, παραμένει πρόκληση.

Ο κύριος στόχος αυτής της εργασίας είναι να γεφυρώσει αυτό το κενό μέσω της ανάπτυξης ενός ολοκληρωμένου αγωγού επεξεργασίας δεδομένων που συνδυάζει δεδομένα οδήγησης και καιρικές συνθήκες για την αξιολόγηση της ασφαλούς οδηγικής συμπεριφοράς. Η μεθοδολογία που ακολουθήθηκε περιλαμβάνει την ενοποίηση δεδομένων από πολλαπλές πηγές, τη στατιστική ανάλυση δεδομένων και την εφαρμογή προηγμένων τεχνικών μηχανικής μάθησης, όπως Gradient Boosting και Νευρωνικά Δίκτυα. Ειδικότερα, έγινε χρήση μεθόδων όπως το Optuna για τη βελτιστοποίηση των υπερπαραμέτρων των μοντέλων, ενώ εφαρμόστηκαν τεχνικές συνόλου, όπως Voting Regressor, Stacking Regressor και Bagging Regressor, για την περαιτέρω βελτίωση της ακρίβειας των προβλέψεων.

Τα αποτελέσματα των πειραμάτων δείχνουν ότι τα μοντέλα που αναπτύχθηκαν μπορούν να προβλέψουν με υψηλή ακρίβεια τις βαθμολογίες ασφαλείας οδήγησης, λαμβάνοντας υπόψη τις ατομικές διαφορές στην οδηγική συμπεριφορά και τις διαφορετικές περιβαλλοντικές συνθήκες. Το Gradient Boosting αποδείχθηκε ότι υπερέχει στην ακρίβεια πρόβλεψης έναντι των νευρωνικών δικτύων για το συγκεκριμένο σύνολο δεδομένων, ενώ οι συνδυαστικές μέθοδοι συνόλου επέτρεψαν τη βελτίωση της σταθερότητας και της ευρωστίας των τελικών προβλέψεων.

Η εργασία αυτή συμβάλλει σημαντικά στην κατανόηση της δυναμικής της οδηγικής συμπεριφοράς υπό διάφορες συνθήκες και προτείνει ένα ισχυρό εργαλείο για τη βελτίωση της οδικής ασφάλειας. Επιπλέον, προτείνει μελλοντικές κατευθύνσεις για την περαιτέρω βελτίωση των μοντέλων πρόβλεψης, όπως η ενσωμάτωση επιπλέον παραγόντων που επηρεάζουν την οδήγηση και η ενσωμάτωση μεγαλύτερων συνόλων δεδομένων που θα μπορούσαν να αυξήσουν την ακρίβεια και τη γενικευσιμότητα των αποτελεσμάτων.

Λέξεις κλειδιά: ασφαλής οδηγική συμπεριφορά, μηχανική μάθηση, Gradient Boosting, Νευρωνικά Δίκτυα, καιρικές συνθήκες, ανάλυση δεδομένων.

Abstract ή Περίληψη στα Αγγλικά

This thesis focuses on the development of an innovative model to calculate the safe driving behavior index, utilizing data collected while driving in real time. The need for improved predictive models arose from the increased data collection capabilities through modern technologies such as GPS devices and smart vehicles, as well as the need to improve road safety and driving experience under different environmental conditions. Despite the availability of large data sets, their integration and analysis, combined with environmental factors such as weather conditions, remains a challenge.

The main objective of this work is to bridge this gap by developing an integrated data processing pipeline that combines driving and weather data to assess safe driving behavior. The methodology followed includes the integration of data from multiple sources, statistical data analysis and the application of advanced machine learning techniques such as Gradient Boosting and Neural Networks. In particular, methods such as Optuna were used to optimize the hyperparameters of the models, while ensemble techniques such as Voting Regressor, Stacking Regressor and Bagging Regressor were applied to further improve the accuracy of predictions.

Experiment results show that the developed models can predict driving safety scores with high accuracy, taking into account individual differences in driving behavior and different environmental conditions. Gradient Boosting was shown to outperform neural networks in prediction accuracy for this particular data set, while combinatorial ensemble methods allowed to improve the stability and robustness of the final predictions.

This work makes a significant contribution to understanding the dynamics of driving behavior under various conditions and suggests a powerful tool for improving road safety. In addition, it suggests future directions to further improve the prediction models, such as incorporating additional factors affecting driving and incorporating larger data sets that could increase the accuracy and generalizability of the results.

Keywords: safe driving behavior, machine learning, Gradient Boosting, Neural Networks, weather conditions, data analysis.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ/ΑΚΡΩΝΥΜΙΑ

XGBoost	Extreme Gradient Boosting
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
SVR	Support Vector Regressor

ΕΙΣΑΓΩΓΗ

1. Περιγραφή του Ευρύτερου Πεδίου και του Προβλήματος

Τα τελευταία χρόνια, η σύγκλιση της επιστήμης δεδομένων με τον τομέα των μεταφορών έχει οδηγήσει σε αξιοσημείωτες εξελίξεις, ιδιαίτερα στην ανάλυση της οδηγικής συμπεριφοράς. Η ευρεία διάδοση των συσκευών με δυνατότητες GPS και η ανάπτυξη των έξυπνων οχημάτων έχουν δημιουργήσει έναν τεράστιο όγκο δεδομένων που αφορά την οδήγηση. Αυτά τα δεδομένα, όταν συνδυάζονται με εξωτερικούς παράγοντες όπως οι καιρικές συνθήκες, προσφέρουν μοναδικές ευκαιρίες για τη βελτίωση της ασφάλειας στην οδήγηση, την αύξηση της απόδοσης, καθώς και τη βελτίωση της συνολικής οδηγικής εμπειρίας.

Ωστόσο, παρά την αυξημένη διαθεσιμότητα εκτεταμένων συνόλων δεδομένων οδήγησης, η ενσωμάτωσή τους και η ανάλυσή τους σε συνδυασμό με περιβαλλοντικούς παράγοντες, όπως οι καιρικές συνθήκες, εξακολουθεί να αποτελεί πρόκληση. Οι παραδοσιακές μέθοδοι ανάλυσης συχνά αποτυγχάνουν να αποτυπώσουν τις σύνθετες αλληλεπιδράσεις μεταξύ της οδηγικής συμπεριφοράς και των περιβαλλοντικών συνθηκών. Αυτό το κενό στην κατανόηση όχι μόνο περιορίζει την ανάπτυξη προγνωστικών μοντέλων για την ενίσχυση της οδική ασφάλεια, αλλά και παρεμποδίζει την εξέλιξη προσαρμοστικών συστημάτων υποβοήθησης οδήγησης, τα οποία θα μπορούσαν να ανταποκρίνονται δυναμικά στις μεταβαλλόμενες συνθήκες του περιβάλλοντος.

2. Συμβολή της Πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία αποσκοπεί στην αντιμετώπιση των αναφερόμενων προκλήσεων μέσω της ανάπτυξης ενός αγωγού επεξεργασίας δεδομένων (data processing pipeline) που αναλύει λεπτομερώς τα δεδομένα οδήγησης σε συνδυασμό με τις καιρικές συνθήκες. Οι κύριες συνεισφορές της εργασίας περιλαμβάνουν τα εξής:

- **Ενοποίηση Δεδομένων (Data Integration):** Η εργασία αναπτύσσει ένα ισχυρό πλαίσιο για την ενοποίηση διαφορετικών συνόλων δεδομένων οδήγησης με δεδομένα καιρού που συλλέγονται από το OpenWeatherMap API. Αυτή η ενοποίηση επιτρέπει μια λεπτομερή ανάλυση της επίδρασης διαφόρων καιρικών παραμέτρων (όπως θερμοκρασία, υγρασία και ταχύτητα ανέμου) στην οδηγική συμπεριφορά.
- **Στατιστική Ανάλυση Δεδομένων:** Χρησιμοποιούνται στατιστικές μέθοδοι για τον υπολογισμό βασικών μετρήσεων (Mean, Median, Mode, Standard Deviation, Variance, Min and Max) τόσο για τα δεδομένα οδήγησης όσο και για τις καιρικές συνθήκες. Η διπλή αυτή ανάλυση παρέχει πλούσια δεδομένα, αναδεικνύοντας την πολύπλοκη δυναμική της οδήγησης υπό διάφορες περιβαλλοντικές συνθήκες.
- **Προεπεξεργασία και Καθαρισμός Δεδομένων:** Αυτά τα βήματα είναι ζωτικής σημασίας για τη βελτίωση της ποιότητας και της αξιοπιστίας των δεδομένων πριν

από την εφαρμογή αλγορίθμων μηχανικής μάθησης. Οι διαδικασίες περιλαμβάνουν τεχνικές κανονικοποίησης και κλιμάκωσης για την προσαρμογή των τιμών των χαρακτηριστικών σε συγκεκριμένα εύρη και τη μετατροπή κατηγορηματικών μεταβλητών σε δυαδικά χαρακτηριστικά, διευκολύνοντας την επεξεργασία τους από τους αλγόριθμους. Επιπλέον, πραγματοποιήθηκε ανίχνευση και αντιμετώπιση ελλιπών δεδομένων, καθώς και αφαίρεση συσχετιζόμενων χαρακτηριστικών, κρατώντας μόνο τα χρήσιμα για την ανάλυση. Αυτές οι διαδικασίες συμβάλλουν σε πιο ακριβείς και αποδοτικές προβλέψεις από τα μοντέλα μηχανικής μάθησης.

- **Ανάπτυξη Μοντέλου Μηχανικής Μάθησης:** Η εργασία εξερευνά τη χρήση προηγμένων τεχνικών μηχανικής μάθησης, όπως το Gradient Boosting και τα Νευρωνικά Δίκτυα, για την πρόβλεψη των βαθμολογιών ασφαλείας οδήγησης (driving safety scores). Η διαδικασία περιλαμβάνει βελτιστοποίηση υπερπαραμέτρων μέσω του εργαλείου Optuna, προκειμένου να επιτευχθεί η καλύτερη δυνατή απόδοση των μοντέλων.
- **Εφαρμογή Εξατομικευμένων Προσεγγίσεων:** Αναπτύσσονται εξατομικευμένα μοντέλα για κάθε οδηγό, χρησιμοποιώντας τεχνικές συνόλου, με σκοπό τη βελτίωση της ακρίβειας πρόβλεψης. Αυτή η προσέγγιση λαμβάνει υπόψη τις ατομικές διαφορές στην οδηγική συμπεριφορά, βελτιώνοντας την ικανότητα των μοντέλων να προβλέπουν με ακρίβεια.

Η πτυχιακή εργασία παρέχει ένα πολυδιάστατο εργαλείο ανάλυσης που ενσωματώνει δεδομένα οδήγησης και καιρικών συνθηκών, προάγοντας την οδική ασφάλεια μέσω ακριβέστερων μοντέλων πρόβλεψης και εξατομικευμένων προσεγγίσεων. Με αυτόν τον τρόπο, συμβάλλει σημαντικά στη βελτίωση της κατανόησης της οδηγικής συμπεριφοράς και της δυναμικής της, κάτω από διαφορετικές περιβαλλοντικές συνθήκες.

3. Δομή της Πτυχιακής εργασίας

Η πτυχιακή εργασία είναι οργανωμένη ως εξής:

- **Κεφάλαιο 1: Σχετικές Εργασίες**
  Το κεφάλαιο αυτό περιλαμβάνει την επισκόπηση της σχετικής βιβλιογραφίας και των προηγούμενων ερευνών που έχουν διεξαχθεί στον τομέα της ανάλυσης της οδηγικής συμπεριφοράς και της επίδρασης των εξωτερικών συνθηκών, όπως οι καιρικές συνθήκες, στην οδικής ασφάλεια.
- **Κεφάλαιο 2: Προτεινόμενη Προσέγγιση για την Ανάπτυξη του Μοντέλου**
  Στο κεφάλαιο αυτό περιγράφεται η μεθοδολογία που ακολουθήθηκε για την ανάπτυξη του μοντέλου, συμπεριλαμβανομένων των τεχνικών μηχανικής μάθησης που χρησιμοποιήθηκαν, της διαδικασίας βελτιστοποίησης των υπερπαραμέτρων και των μεθόδων συνόλου που εφαρμόστηκαν για την ενίσχυση της απόδοσης του μοντέλου.

- **Κεφάλαιο 3: Πειραματική Αξιολόγηση**
  Στο κεφάλαιο αυτό παρουσιάζονται τα πειραματικά αποτελέσματα από την εφαρμογή του μοντέλου σε πραγματικά δεδομένα οδήγησης. Γίνεται ανάλυση της απόδοσης του μοντέλου και συγκρίνονται τα αποτελέσματα με άλλες μεθόδους πρόβλεψης.
- **Κεφάλαιο 4: Συμπεράσματα και Μελλοντικές Εργασίες**
  Το τελευταίο κεφάλαιο περιλαμβάνει τα βασικά συμπεράσματα της εργασίας και προτείνει κατευθύνσεις για μελλοντική έρευνα και βελτίωση του μοντέλου.

ΚΕΦ.1: Σχετικές Εργασίες

1.1 Επισκόπηση Σχετικών Εργασιών

Η έρευνα στην ανάλυση της οδηγικής συμπεριφοράς, ειδικά σε σχέση με τις καιρικές συνθήκες, έχει σημειώσει σημαντικά βήματα προόδου τα τελευταία χρόνια. Πολυάριθμες μελέτες έχουν επικεντρωθεί στην κατανόηση του τρόπου με τον οποίο οι καιρικές συνθήκες επηρεάζουν τα πρότυπα οδήγησης, χρησιμοποιώντας δεδομένα από πολλαπλές πηγές και αναλυτικές τεχνικές.

1.2 Συσχέτιση Οδήγησης και Καιρικών Συνθηκών

Μία από τις πιο πρόσφατες και σημαντικές μελέτες σε αυτό το πεδίο είναι αυτή των Elyoussoufi, Walker, Black και DeGirolamo (2023), οι οποίοι διερεύνησαν τη σχέση μεταξύ των δυσμενών καιρικών συνθηκών, της κινητικότητας στους δρόμους και την συμπεριφορά των οδηγών. Χρησιμοποιώντας δεδομένα καιρού και κυκλοφορίας από το Colorado, οι ερευνητές ανακάλυψαν ότι τις ημέρες με χιόνι υπήρχε σημαντική μείωση στον όγκο της κυκλοφορίας σε σύγκριση με τις καθαρές και βροχερές ημέρες. Επίσης, οι οδηγοί κινούνταν με χαμηλότερες ταχύτητες τις χιονισμένες ημέρες, κάτι που αύξησε τη διάρκεια των ταξιδιών. Αυτά τα ευρήματα υπογραμμίζουν την ανάγκη προσαρμογής των στρατηγικών διαχείρισης της κυκλοφορίας υπό δυσμενείς καιρικές συνθήκες.

Μια προηγούμενη μελέτη των Pisano, Goodwin, και Rossetti (2008) εξέτασε την επίδραση των καιρικών συνθηκών στην ασφάλεια της οδήγησης στις Ηνωμένες Πολιτείες. Τα αποτελέσματα τους έδειξαν ότι περίπου το 24% όλων των ατυχημάτων σχετίζονται με καιρικές συνθήκες, με το 47% αυτών να σημειώνεται κατά τη διάρκεια βροχοπτώσεων και το 75% σε βρεγμένο οδόστρωμα. Η μελέτη αυτή υπογράμμισε ότι η βροχή αυξάνει σημαντικά τον κίνδυνο ατυχημάτων, κυρίως λόγω της μειωμένης τριβής του οδοστρώματος, γεγονός που επηρεάζει αρνητικά την πρόσφυση, τη σταθερότητα και την ικανότητα ελιγμών του οχήματος. Συνολικά, οι δυσμενείς καιρικές συνθήκες επιδεινώνουν την ασφάλεια της οδήγησης, αυξάνοντας τον κίνδυνο ατυχημάτων.

1.3 Εργασίες με Παρόμοια Δεδομένα

1.3.1 Driver Behavior Profiling using Smartphones: A Low-Cost Platform for Driver Monitoring

Μια σημαντική συνεισφορά στην ανάλυση της οδηγικής συμπεριφοράς προέρχεται από τη μελέτη των Castignani, Derrmann, Frank, και Engel (2015), η οποία εξετάζει τη χρήση smartphones για την παρακολούθηση και το προφίλ της οδηγικής συμπεριφοράς. Η πλατφόρμα SenseFleet, που προτείνεται στη μελέτη, αποτελεί μια οικονομική λύση για την παρακολούθηση της οδηγικής συμπεριφοράς μέσω δεδομένων από αισθητήρες κινητών συσκευών, όπως το GPS και το επιταχυνσιόμετρο (accelerometer).

Η πλατφόρμα αυτή χρησιμοποιεί δεδομένα από τους αισθητήρες των smartphones για την ανίχνευση επικίνδυνων οδηγικών συμβάντων, όπως επιταχύνσεις, φρεναρίσματα, και απότομους ελιγμούς. Ενσωματώνει επίσης στοιχεία καιρού και τοπογραφίας, παρέχοντας ένα λεπτομερές προφίλ του οδηγού. Η καινοτομία της μεθόδου έγκειται στη χρήση συστήματος fuzzy logic, το οποίο προσαρμόζει τα όρια ανίχνευσης συμβάντων, επιτρέποντας την ακριβή ανίχνευση επικίνδυνων συμβάντων. Τα αποτελέσματα έδειξαν ότι η πλατφόρμα είναι ικανή να εντοπίζει σημαντικές διαφορές μεταξύ ασφαλούς και επιθετικής οδήγησης, προσφέροντας έτσι ένα πολύτιμο εργαλείο για την ασφάλιση αυτοκινήτων, τη διαχείριση στόλων, και την προώθηση πιο ασφαλών οδηγικών συνηθειών.

1.3.2 Profiling drivers based on driver dependent vehicle driving features

Μια άλλη σημαντική μελέτη στον τομέα της ανάλυσης της οδηγικής συμπεριφοράς είναι αυτή των Halim, Kalsoom, και Baig (2016), η οποία επικεντρώνεται στη δημιουργία προφίλ οδηγών με βάση χαρακτηριστικά οδήγησης που εξαρτώνται από τον οδηγό. Χρησιμοποιώντας ένα προσαρμοσμένο σύστημα υλικού και λογισμικού, οι ερευνητές καταχώρησαν δεδομένα από πολλούς οδηγούς και τα ταξινόμησαν σε προφίλ μέσω τεχνικών συσταδοποίησης όπως το k-means. Η μελέτη κατέδειξε ότι τα προφίλ των οδηγών μπορούν να κατηγοριοποιηθούν με ακρίβεια και να χρησιμοποιηθούν για τη βελτίωση της ασφάλειας οδήγησης μέσω της παρακολούθησης σε πραγματικό χρόνο.

ΚΕΦ.2: Προτεινόμενη Προσέγγιση για την Ανάπτυξη του Μοντέλου

2.1 Επισκόπηση

Αυτή η πτυχιακή εργασία προτείνει μια ολοκληρωμένη προσέγγιση για την ανάπτυξη ενός μοντέλου πρόβλεψης των driving safety scores, αντιμετωπίζοντας τις προκλήσεις που σχετίζονται με την ακρίβεια και την ανθεκτικότητα. Η προτεινόμενη μεθοδολογία περιλαμβάνει τα εξής κύρια βήματα:

- **Εκπαίδευση και σύγκριση μοντέλων:** Πραγματοποιείται εκπαίδευση Νευρωνικών Δικτύων και μοντέλων Gradient Boosting, ώστε να εξεταστεί ποιο μοντέλο επιδεικνύει την καλύτερη απόδοση στην πρόβλεψη των driving safety scores.
- **Δημιουργία εξατομικευμένων μοντέλων παλινδρόμησης:** Για κάθε οδηγό (driver), αναπτύσσονται μεμονωμένα μοντέλα παλινδρόμησης. Οι υπερπαράμετροι αυτών των μοντέλων βελτιστοποιούνται μέσω του Optuna, ενός ισχυρού εργαλείου για την αναζήτηση και την προσαρμογή υπερπαραμέτρων.
- **Συνδυασμός μοντέλων μέσω μεθόδων συνόλου (ensemble):** Τα βελτιστοποιημένα μοντέλα συνδυάζονται χρησιμοποιώντας μεθόδους συνόλου, όπως Voting Regressor, Stacking Regressor και Bagging Regressor, για τη βελτίωση της ακρίβειας και της σταθερότητας των τελικών προβλέψεων.

Ο στόχος αυτής της προσέγγισης είναι η ανάπτυξη ενός ακριβούς και ανθεκτικού μοντέλου πρόβλεψης, ικανού να λειτουργεί αξιόπιστα υπό διαφορετικές περιβαλλοντικές συνθήκες και προκλήσεις. Η χρήση μεθόδων συνόλου επιτρέπει την αξιοποίηση των δυνατών σημείων κάθε μοντέλου, οδηγώντας σε μια πιο ισχυρή και ευέλικτη λύση.

2.2 Εκπαίδευση Νευρωνικών Δικτύων και Gradient Boosting Μοντέλων για την Αξιολόγηση της Απόδοσης

2.2.1 Gradient Boosting

Το Gradient Boosting είναι ένας ισχυρός αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται ευρέως για την κατασκευή ακριβών μοντέλων πρόβλεψης. Βασίζεται στη διαδοχική βελτίωση της ακρίβειας των προβλέψεων μέσω της συνδυαστικής χρήσης απλών μοντέλων, όπως τα δέντρα αποφάσεων, με στόχο τη διόρθωση των σφαλμάτων που έκαναν τα προηγούμενα μοντέλα.

2.2.1.1 Βασικές Έννοιες

- **Ensemble Learning (Εκμάθηση Συνόλου):** Το Gradient Boosting είναι μέρος των τεχνικών εκμάθησης συνόλου, όπου πολλαπλά μοντέλα συνδυάζονται για να δημιουργήσουν ένα πιο ισχυρό και ακριβές μοντέλο. Εκμεταλλεύεται τη διαφοροποίηση των αδυναμιών των επιμέρους μοντέλων, παρέχοντας έτσι βελτιωμένες προβλέψεις.
- **Boosting (Ενίσχυση):** Το Boosting είναι μια διαδοχική μέθοδος εκμάθησης συνόλου, όπου κάθε νέο μοντέλο εκπαιδεύεται για να διορθώσει τα σφάλματα που έκαναν τα προηγούμενα μοντέλα. Σε αντίθεση με το Bagging (όπως τα Random Forests), όπου τα μοντέλα εκπαιδεύονται ανεξάρτητα, το Boosting επικεντρώνεται σε διαδοχική εκπαίδευση, με κάθε νέο μοντέλο να εστιάζει σε παραδείγματα όπου τα προηγούμενα μοντέλα απέτυχαν.

2.2.1.2 Πώς Λειτουργεί το Gradient Boosting

- **Αρχικό Μοντέλο:** Η διαδικασία ξεκινά με την κατασκευή ενός αρχικού μοντέλου, το οποίο είναι συχνά απλό. Για παράδειγμα, στην παλινδρόμηση, το αρχικό μοντέλο μπορεί να προβλέπει τη μέση τιμή των στόχων.
- **Υπολογισμός Σφαλμάτων (Residuals):** Τα υπολείμματα, δηλαδή οι διαφορές μεταξύ των πραγματικών τιμών και των προβλέψεων του αρχικού μοντέλου, υπολογίζονται και χρησιμοποιούνται ως η βάση για την εκπαίδευση του επόμενου μοντέλου.

- **Εκπαίδευση Νέου Μοντέλου στα Υπολείμματα:** Ένα νέο μοντέλο εκπαιδεύεται για να προβλέψει τα υπολείμματα. Αυτό το μοντέλο επικεντρώνεται στις περιοχές όπου το αρχικό μοντέλο απέτυχε, προσπαθώντας να διορθώσει τις ανακρίβειες.

- **Ενημέρωση του Μοντέλου:** Οι προβλέψεις του νέου μοντέλου προστίθενται στις προβλέψεις του προηγούμενου μοντέλου. Ένας συντελεστής κλιμάκωσης, γνωστός ως "ρυθμός εκμάθησης" (learning rate), εφαρμόζεται για να ελέγξει το πόσο κάθε νέο μοντέλο συνεισφέρει στη συνολική πρόβλεψη.

- **Επανάληψη:** Τα παραπάνω βήματα επαναλαμβάνονται για έναν προκαθορισμένο αριθμό επαναλήψεων ή μέχρι να επιτευχθεί ικανοποιητική μείωση των υπολειμμάτων.

2.2.1.3 Βασικά Συστατικά

- **Συνάρτηση Απώλειας (Loss Function):** Η επιλογή της συνάρτησης απώλειας εξαρτάται από το είδος του προβλήματος. Στην παλινδρόμηση, συχνά χρησιμοποιείται το Μέσο Τετραγωνικό Σφάλμα (MSE). Το Gradient Boosting προσπαθεί να ελαχιστοποιήσει αυτή τη συνάρτηση μέσω της εκπαίδευσης των διαδοχικών μοντέλων.

- **Βασικά Μοντέλα (Base Learners):** Τα δέντρα αποφάσεων είναι συνήθως τα βασικά μοντέλα στο Gradient Boosting. Αν και τα δέντρα αποφάσεων μπορούν να συλλάβουν περίπλοκες αλληλεπιδράσεις στα δεδομένα, συχνά χρησιμοποιούνται "ρηχά" δέντρα (stumps) για να μειωθεί ο κίνδυνος υπερπροσαρμογής (overfitting).

- **Ρυθμός Εκμάθησης (Learning Rate):** Ο ρυθμός εκμάθησης καθορίζει την επιρροή κάθε νέου δέντρου στη συνολική πρόβλεψη. Χαμηλότερες τιμές του ρυθμού εκμάθησης τείνουν να οδηγούν σε καλύτερη απόδοση, αν και απαιτούν μεγαλύτερο αριθμό δέντρων.

- **Αριθμός Δέντρων (Number of Trees):** Ο αριθμός των δέντρων είναι μια κρίσιμη υπερπαράμετρος. Λίγα δέντρα μπορεί να οδηγήσουν σε κακή προσαρμογή (underfitting), ενώ ένας υπερβολικά μεγάλος αριθμός δέντρων μπορεί να προκαλέσει υπερπροσαρμογή.

2.2.1.4 Πλεονεκτήματα και Μειονεκτήματα

Οι συγκεκριμένες προσεγγίσεις του Gradient Boosting παρουσιάζουν μια σειρά από πλεονεκτήματα που συνοψίζονται στα ακόλουθα:

- **Ικανότητα Χειρισμού Ποικιλίας Δεδομένων:** Το Gradient Boosting μπορεί να διαχειριστεί τόσο κατηγορικά όσο και αριθμητικά δεδομένα με υψηλή αποτελεσματικότητα.
- **Ανθεκτικότητα στην Υπερπροσαρμογή:** Με τη χρήση περιορισμένων σε μέγεθος δέντρων και ρυθμού εκμάθησης, το Gradient Boosting μπορεί να ελέγξει την υπερπροσαρμογή και να διατηρήσει καλή γενικευσιμότητα.
- **Υψηλή Προγνωστική Ακρίβεια:** Το Gradient Boosting μπορεί να εντοπίσει και να συλλάβει πολύπλοκα μοτίβα στα δεδομένα, προσφέροντας υψηλή ακρίβεια στις προβλέψεις.

Παρόλα αυτά, η χρήση του Gradient Boosting συνοδεύεται και από ορισμένα μειονεκτήματα που πρέπει να ληφθούν υπόψη:

- **Υπολογιστική Απαίτηση:** Η διαδοχική φύση της εκπαίδευσης καθιστά το Gradient Boosting υπολογιστικά απαιτητικό, ιδιαίτερα για μεγάλα σύνολα δεδομένων ή πολύπλοκα μοντέλα.
- **Ευαισθησία στις Ρυθμίσεις Υπερπαραμέτρων:** Ο σωστός συντονισμός των υπερπαραμέτρων όπως ο ρυθμός εκμάθησης και ο αριθμός δέντρων είναι κρίσιμος. Κακή ρύθμιση μπορεί να οδηγήσει είτε σε υποπροσαρμογή (underfitting) είτε σε υπερπροσαρμογή (overfitting).
- **Περιορισμένη Ερμηνευσιμότητα:** Σε σύγκριση με απλούστερα μοντέλα, όπως η γραμμική παλινδρόμηση, το Gradient Boosting μπορεί να είναι πιο δύσκολο στην ερμηνεία.

2.2.1.5 Συμπέρασμα

Το Gradient Boosting είναι μια εξαιρετικά ευέλικτη και ισχυρή μέθοδος για την κατασκευή ακριβών μοντέλων πρόβλεψης μέσω της διαδοχικής διόρθωσης σφαλμάτων. Ωστόσο, η επιτυχής εφαρμογή του απαιτεί προσεκτικό συντονισμό των υπερπαραμέτρων για τη μέγιστη δυνατή απόδοση

2.2.2 Νευρωνικά δίκτυα

Τα νευρωνικά δίκτυα είναι υπολογιστικά μοντέλα εμπνευσμένα από τη δομή και τη λειτουργία του ανθρώπινου εγκεφάλου. Χρησιμοποιούνται στη μηχανική μάθηση για την επίλυση σύνθετων προβλημάτων όπως η ταξινόμηση, η παλινδρόμηση, η αναγνώριση προτύπων, η αναγνώριση εικόνας, η επεξεργασία φυσικής γλώσσας και η ανάπτυξη συστημάτων τεχνητής νοημοσύνης σε παιχνίδια.

2.2.2.1 Δομή ενός νευρωνικού δικτύου

- **Νευρώνες (κόμβοι):** Κάθε νευρώνας είναι μια μονάδα επεξεργασίας που λαμβάνει είσοδο, εφαρμόζει μια συνάρτηση και παράγει μια έξοδο. Με μαθηματικούς όρους, ένας νευρώνας παίρνει το σταθμισμένο άθροισμα των εισόδων και το περνάει μέσα από μια συνάρτηση ενεργοποίησης για να καθορίσει την έξοδό του.

- **Στρώματα (Layers):**

    - **Επίπεδο Εισόδου (Input Layer):** Το πρώτο επίπεδο σε ένα νευρωνικό δίκτυο, όπου τα δεδομένα τροφοδοτούνται στο σύστημα. Κάθε νευρώνας σε αυτό το επίπεδο αντιστοιχεί σε ένα χαρακτηριστικό (feature) από τα δεδομένα εισόδου.

    - **Κρυφά Επίπεδα (Hidden Layers):** Αυτά είναι τα ενδιάμεσα επίπεδα μεταξύ του επιπέδου εισόδου και του επιπέδου εξόδου. Εκεί πραγματοποιούνται οι υπολογισμοί που επιτρέπουν στο δίκτυο να μάθει από τα δεδομένα. Ο αριθμός των κρυφών επιπέδων και των νευρώνων τους μπορεί να επηρεάσει σημαντικά την ικανότητα του δικτύου να καταγράφει πολύπλοκα μοτίβα.

- **Επίπεδο Εξόδου (Output Layer):** Το τελικό επίπεδο που παράγει την έξοδο του δικτύου. Ο αριθμός των νευρώνων σε αυτό το επίπεδο αντιστοιχεί στον αριθμό των κατηγοριών ή των τιμών εξόδου.

- **Weights and Biases:**
  - **Weights (Βάρη):** Κάθε σύνδεση μεταξύ νευρώνων έχει ένα βάρος, το οποίο ρυθμίζει την επιρροή της εισόδου στην έξοδο. Κατά την εκπαίδευση, το δίκτυο προσαρμόζει αυτά τα βάρη για να ελαχιστοποιήσει το σφάλμα στις προβλέψεις.
  - **Biases:** Προστίθενται στο σταθμισμένο άθροισμα των εισόδων πριν από την εφαρμογή της συνάρτησης ενεργοποίησης, επιτρέποντας στο δίκτυο να προσαρμόζεται καλύτερα στα δεδομένα.

- **Συναρτήσεις Ενεργοποίησης (Activation Functions):** Οι συναρτήσεις ενεργοποίησης εισάγουν τη μη γραμμικότητα στο μοντέλο, επιτρέποντας στο δίκτυο να μάθει πολύπλοκα μοτίβα. Οι πιο κοινές είναι:
  - **Sigmoid:** Παράγει έξοδο μεταξύ 0 και 1, που χρησιμοποιείται συχνά στη δυαδική ταξινόμηση.
  - **ReLU (Rectified Linear Unit):** Εξάγει την είσοδο εάν είναι θετική, διαφορετικά μηδέν. Χρησιμοποιείται ευρέως για να μειώσει το πρόβλημα της εξαφάνισης της κλίσης.
  - **Tanh:** Παρόμοιο με το Sigmoid αλλά βγάζει μεταξύ -1 και 1, που χρησιμοποιείται συχνά σε κρυφά επίπεδα

2.2.2.2 Πώς μαθαίνουν τα νευρωνικά δίκτυα: Τα νευρωνικά δίκτυα μαθαίνουν μέσω μιας διαδικασίας που ονομάζεται εκπαίδευση (training), η οποία συνήθως περιλαμβάνει τα ακόλουθα βήματα:

- **Forward Propagation (Μπροστινή διάδοση):** Τα δεδομένα περνούν μέσα από το δίκτυο από το επίπεδο εισόδου στο επίπεδο εξόδου, παράγοντας μια πρόβλεψη.

- **Loss Calculation (Υπολογισμός απώλειας):** Η πρόβλεψη συγκρίνεται με την πραγματική τιμή χρησιμοποιώντας μια συνάρτηση απώλειας, η οποία μετρά τη διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Στην παλινδρόμηση, μια κοινή συνάρτηση είναι το Μέσο Τετραγωνικό Σφάλμα (MSE).

- **Backpropagation (Πίσω διάδοση):** Το δίκτυο υπολογίζει την παράγωγο της συνάρτησης απώλειας ως προς κάθε βάρος, χρησιμοποιώντας τον κανόνα της αλυσίδας (chain rule). Αυτό επιτρέπει στο σφάλμα να διαδίδεται προς τα πίσω στο δίκτυο, ξεκινώντας από το επίπεδο εξόδου.

- **Weight Update (Ενημέρωση βάρους):** Τα βάρη ενημερώνονται χρησιμοποιώντας έναν αλγόριθμο βελτιστοποίησης με σκοπό την ελαχιστοποίηση της απώλειας. Αυτό επαναλαμβάνεται μέχρι το δίκτυο να συγκλίνει, δηλαδή η απώλεια να ελαχιστοποιηθεί.

- **Iteration (Επανάληψη):** Τα βήματα επαναλαμβάνονται για πολλές εποχές (epochs) έως ότου το δίκτυο επιτύχει την επιθυμητή απόδοση.

2.2.2.3 Νευρωνικά δίκτυα με διαφορετικούς βελτιστοποιητές

- **Βελτιστοποιητής Adam:**
    - **Περιγραφή:** Ο Adam (Adaptive Moment Estimation) είναι ένας δημοφιλής αλγόριθμος βελτιστοποίησης που συνδυάζει τα πλεονεκτήματα των AdaGrad και RMSProp. Είναι γνωστός για την αποδοτικότητα και την καταλληλότητά του για μεγάλα σύνολα δεδομένων και μοντέλα με πολλαπλές παραμέτρους.
    - **Πλεονεκτήματα:**
        - Προσαρμοστικοί ρυθμοί μάθησης για κάθε παράμετρο.
        - Συνδυάζει ορμή και σταθερότητα στην εκπαίδευση.
        - Αποτελεσματικός σε προβλήματα μεγάλης κλίμακας.

- **Βελτιστοποιητής AdamW:**
    - **Περιγραφή:** Ο AdamW είναι μια βελτιωμένη έκδοση του Adam που διορθώνει τη διαχείριση της αποσύνθεσης βάρους (weight decay). Εισήχθη για να διασφαλίσει ότι η αποσύνθεση βάρους λειτουργεί ως τακτοποιητής (regularization) χωρίς να επηρεάζει τον ρυθμό εκμάθησης.
    - **Πλεονεκτήματα:**
        - Σωστή εφαρμογή της αποσύνθεσης βάρους.
        - Καλύτερη γενίκευση και απόδοση σε σύγκριση με τον Adam.
        - Ευκολία χρήσης με λιγότερο συντονισμό υπερπαραμέτρων.

- **Βελτιστοποιητής Nadam:**
    - **Περιγραφή:** Το Nadam (Nesterov-accelerated Adaptive Moment Estimation) είναι μια παραλλαγή του Adam που ενσωματώνει την ορμή Nesterov, βελτιώνοντας την ταχύτητα σύγκλισης και την απόδοση στην εκπαίδευση μοντέλων βαθιάς μάθησης.
    - **Πλεονεκτήματα:**
        - Ταχύτερη σύγκλιση λόγω της ορμής Nesterov.
        - Καλύτερη απόδοση σε πολύπλοκα σενάρια.
        - Ισχυρός σε αρχιτεκτονικές νευρωνικών δικτύων και σύνολα δεδομένων διαφορετικών τύπων.

2.2.2.4 Συμπέρασμα:

Τα νευρωνικά δίκτυα είναι ισχυρά εργαλεία για την ανάλυση και την πρόβλεψη σύνθετων δεδομένων. Η επιτυχής εκπαίδευσή τους εξαρτάται από την κατάλληλη επιλογή αρχιτεκτονικής, τη σωστή ρύθμιση των υπερπαραμέτρων και την επιλογή του κατάλληλου βελτιστοποιητή. Οι βελτιστοποιητές όπως οι Adam, AdamW και Nadam προσφέρουν διαφορετικά πλεονεκτήματα που μπορούν να ενισχύσουν την απόδοση του μοντέλου σε διάφορα σενάρια.

2.2.2.5 Χρήση Dropout και Early Stopping για Βελτίωση της Γενίκευσης και Αποφυγή του Overfitting

Για τη δημιουργία νευρωνικών δικτύων που είναι πιο ανθεκτικά στο overfitting και έχουν καλύτερη ικανότητα γενίκευσης σε νέα, άγνωστα δεδομένα, εφαρμόζουμε τις τεχνικές Dropout και Early Stopping.

2.2.2.5.1 Dropout:

- **Πρόληψη Overfitting:** Το Dropout αποτρέπει το νευρωνικό δίκτυο από το να εξαρτάται υπερβολικά από συγκεκριμένους νευρώνες ή χαρακτηριστικά (features) κατά την εκπαίδευση. Κατά τη διάρκεια της εκπαίδευσης, ένα ποσοστό των νευρώνων απενεργοποιείται τυχαία σε κάθε βήμα, αναγκάζοντας το δίκτυο να μάθει πιο γενικευμένα χαρακτηριστικά και να μην βασίζεται σε μεμονωμένα μοτίβα.
- **Ενίσχυση Γενίκευσης:** Με την τυχαία απενεργοποίηση των νευρώνων σε κάθε βήμα, το δίκτυο μαθαίνει να λειτουργεί χωρίς να εξαρτάται από συγκεκριμένους νευρώνες. Αυτό ενισχύει την ικανότητά του να προσαρμόζεται σε νέα, άγνωστα δεδομένα, καθώς το μοντέλο δεν βασίζεται αποκλειστικά σε συγκεκριμένα χαρακτηριστικά που μπορεί να μην είναι διαθέσιμα σε διαφορετικά σύνολα δεδομένων.

2.2.2.5.2 Early Stopping:

- **Αποφυγή Overfitting:** Το Early Stopping παρακολουθεί την απόδοση του μοντέλου σε ένα σύνολο επικύρωσης κατά τη διάρκεια της εκπαίδευσης. Όταν η απόδοση αρχίζει να επιδεινώνεται, δηλαδή όταν το μοντέλο αρχίζει να μαθαίνει το “θόρυβο” των δεδομένων εκπαίδευσης, η εκπαίδευση διακόπτεται. Αυτό αποτρέπει το μοντέλο από το να γίνει υπερβολικά περίπλοκο και να αποτυγχάνει σε νέα δεδομένα.

- **Εξοικονόμηση Χρόνου και Πόρων:** Το Early Stopping σταματά την εκπαίδευση όταν δεν υπάρχει πλέον βελτίωση, αποτρέποντας τη σπατάλη χρόνου και υπολογιστικών πόρων. Αυτό καθιστά την εκπαίδευση πιο αποδοτική, καθώς αποφεύγεται η περιττή συνέχιση της εκπαίδευσης πέρα από το σημείο βέλτιστης απόδοσης.

- **Διατήρηση Ισορροπίας:** Με την έγκαιρη διακοπή της εκπαίδευσης, το Early Stopping βοηθά στη διατήρηση της ισορροπίας μεταξύ της πολυπλοκότητας και της γενικευσιμότητας του μοντέλου. Έτσι, το μοντέλο είναι αρκετά ισχυρό για να καταγράφει σημαντικά μοτίβα, αλλά και αρκετά απλό ώστε να αποφεύγει το overfitting.

2.2.2.5.3 Συμπέρασμα:

Η συνδυαστική χρήση των τεχνικών Dropout και Early Stopping συμβάλλει στη δημιουργία νευρωνικών δικτύων που είναι πιο ανθεκτικά στο overfitting, πιο αποδοτικά στη γενίκευση και ικανά να αποδώσουν καλύτερα σε νέα δεδομένα.

2.2.3 Αποτελέσματα της Εξέτασης των Μοντέλων

Μετά από την αξιολόγηση των μοντέλων, διαπιστώθηκε ότι το Gradient Boosting υπερείχε σταθερά έναντι των νευρωνικών δικτύων τόσο στην ακρίβεια πρόβλεψης όσο και στην ευρωστία των αποτελεσμάτων για το συγκεκριμένο σύνολο δεδομένων. Αυτή η υπεροχή του Gradient Boosting οδήγησε στη μετατόπιση της εστίασης προς τη χρήση των μοντέλων Gradient Boosting στις επόμενες μεθόδους συνόλου που εξετάστηκαν στην εργασία.

2.3 Βελτιστοποίηση των Υπερπαράμετρων μέσω του Optuna

Το Optuna είναι ένα αυτόματο πλαίσιο βελτιστοποίησης υπερπαραμέτρων, σχεδιασμένο να βρίσκει αποτελεσματικά τις βέλτιστες υπερπαραμέτρους για μοντέλα μηχανικής μάθησης. Χρησιμοποιεί προηγμένες τεχνικές όπως το Bayesian optimization και το Tree-structured Parzen Estimator (TPE) για την έξυπνη εξερεύνηση του χώρου των υπερπαραμέτρων.

- **Αναζήτηση Υπερπαραμέτρων:** Το Optuna εκτελεί μια καθοδηγούμενη αναζήτηση για τη βελτιστοποίηση κρίσιμων υπερπαραμέτρων, όπως ο ρυθμός εκμάθησης, το μέγιστο βάθος των δέντρων και ο αριθμός των εκτιμητών. Η διαδικασία αυτή επιτρέπει την εύρεση του συνδυασμού υπερπαραμέτρων που μεγιστοποιεί την απόδοση του μοντέλου.

- **Διαχείριση Δοκιμών:** Το Optuna διαχειρίζεται διαφορετικές δοκιμές υπερπαραμέτρων και αξιολογεί την απόδοσή τους με βάση μια προκαθορισμένη αντικειμενική συνάρτηση, όπως η ελαχιστοποίηση του σφάλματος επικύρωσης. Αυτό επιτρέπει την αποτελεσματική σύγκριση και επιλογή του βέλτιστου συνδυασμού υπερπαραμέτρων.

- **Κλάδεμα Δοκιμών (Pruning):** Ένας από τους μηχανισμούς του Optuna είναι το “pruning”, το οποίο τερματίζει πρόωρα τις δοκιμές που δεν δείχνουν υποσχόμενα αποτελέσματα. Αυτό μειώνει τον χρόνο και τους υπολογιστικούς πόρους που απαιτούνται για τη διαδικασία βελτιστοποίησης, επιταχύνοντας την εύρεση της καλύτερης λύσης.

2.4 Μέθοδοι συνόλου

Μετά τη βελτιστοποίηση μεμονωμένων μοντέλων Gradient Boosting για κάθε οδηγό (driver), οι μέθοδοι συνόλου χρησιμοποιούνται για τον συνδυασμό αυτών των μοντέλων με σκοπό τη βελτίωση της προγνωστικής απόδοσης. Οι μέθοδοι συνόλου εκμεταλλεύονται τα πλεονεκτήματα πολλαπλών μοντέλων για να παράγουν πιο ισχυρές και ακριβείς προβλέψεις.

2.4.1 Voting Regressor

Η μέθοδος Voting Regressor συνδυάζει τις προβλέψεις πολλαπλών μοντέλων υπολογίζοντας τον μέσο όρο των εξόδων τους, βελτιώνοντας έτσι την ακρίβεια και την αξιοπιστία της τελικής πρόβλεψης. Στην παλινδρόμηση, δεν υπάρχουν Hard Voting (Σκληρή Ψηφοφορία) και Soft Voting (Μαλακή Ψηφοφορία) όπως στην ταξινόμηση, αλλά υπάρχουν δύο κύριοι τρόποι συνδυασμού των προβλέψεων:

- **Απλή Μέση Τιμή (Simple Averaging)**: Υπολογίζεται ο μέσος όρος των προβλέψεων από όλα τα μοντέλα.
- **Σταθμισμένη Μέση Τιμή (Weighted Averaging)**: Κάθε μοντέλο έχει ένα βάρος και οι προβλέψεις συνδυάζονται υπολογίζοντας έναν σταθμισμένο μέσο όρο, με τα μοντέλα που θεωρούνται πιο αξιόπιστα να λαμβάνουν μεγαλύτερη βαρύτητα.

Η χρήση του Voting Regressor ενισχύει την απόδοση του συνόλου, εκμεταλλευόμενη τις δυνάμεις διαφορετικών μοντέλων και μετριάζοντας τις αδυναμίες τους.

2.4.2 Stacking Regressor

Το Stacking Regressor είναι μια πιο εξελιγμένη τεχνική συνόλου που συνδυάζει πολλαπλά βασικά μοντέλα (base models) και ένα μετα-μοντέλο (meta-model). Τα βασικά μοντέλα κάνουν τις αρχικές προβλέψεις, και αυτές οι προβλέψεις χρησιμοποιούνται ως χαρακτηριστικά εισόδου για το μετα-μοντέλο, το οποίο στη συνέχεια κάνει την τελική πρόβλεψη.

- **Base Models**: Τα μεμονωμένα μοντέλα Gradient Boosting, τα οποία έχουν βελτιστοποιηθεί για κάθε driver.
- **Meta-Model**: Ένα μοντέλο υψηλότερου επιπέδου, το οποίο μαθαίνει να συνδυάζει τις εξόδους των βασικών μοντέλων για τη βελτίωση της ακρίβειας της πρόβλεψης.

Η μέθοδος Stacking Regressor είναι ιδιαίτερα ισχυρή όταν τα βασικά μοντέλα είναι αρκετά διαφορετικά μεταξύ τους, καθώς το μετα-μοντέλο μπορεί να μάθει τις βέλτιστες συνδυαστικές στρατηγικές.

2.4.3 Bagging Regressor

Το Bagging (Bootstrap Aggregating) είναι μια μέθοδος συνόλου που βελτιώνει τη σταθερότητα και την ακρίβεια των αλγορίθμων μηχανικής μάθησης εκπαιδεύοντας πολλαπλά μοντέλα σε διαφορετικά υποσύνολα των δεδομένων εκπαίδευσης και υπολογίζοντας τον μέσο όρο των προβλέψεών τους.

- **Bootstrap Samples**: Δημιουργούνται πολλαπλά σύνολα δεδομένων με δειγματοληψία με αντικατάσταση από τα αρχικά δεδομένα εκπαίδευσης.
- **Συνάθροιση (Aggregation)**: Οι προβλέψεις από κάθε μοντέλο υπολογίζονται κατά μέσο όρο για την παραγωγή της τελικής πρόβλεψης, μειώνοντας τη διακύμανση και βελτιώνοντας την ευρωστία του μοντέλου.
- **Base Estimator**: Ορίζει το είδος του μοντέλου που χρησιμοποιείται για κάθε μία από τις επαναλήψεις του bagging.
- **N_estimators**: Καθορίζει τον αριθμό των μοντέλων που θα δημιουργηθούν και θα συνδυαστούν για την τελική πρόβλεψη.

Η μέθοδος Bagging Regressor είναι αποτελεσματική στην αντιμετώπιση της υπερπροσαρμογής και στη βελτίωση της συνολικής απόδοσης του μοντέλου.

2.5 Αποτέλεσμα

Η συνδυαστική αυτή προσέγγιση αξιοποιεί τα πλεονεκτήματα του Gradient Boosting και την ευελιξία του Optuna για τη βελτιστοποίηση των υπερπαραμέτρων, καταλήγοντας σε ένα ισχυρό μοντέλο συνόλου που μπορεί να προβλέψει αποτελεσματικά τα driving safety scores κάτω από διαφορετικές περιβαλλοντικές συνθήκες.

ΚΕΦ.3: Πειραματική αξιολόγηση

3.1 Περιγραφή του Dataset

Η συλλογή των δεδομένων πραγματοποιήθηκε μέσω επτά οδηγών που συμμετείχαν σε πολυάριθμα ταξίδια, παρέχοντας ένα πλούσιο και πολυδιάστατο σύνολο πληροφοριών σχετικά με την οδηγική συμπεριφορά. Τα δεδομένα οργανώνονται σε δύο κύρια σύνολα αρχείων, τα οποία επιτρέπουν την εκτενή ανάλυση της συμπεριφοράς των οδηγών και της αλληλεπίδρασής τους με το περιβάλλον.

Πρώτο Αρχείο: Καταγραφή των Ταξιδιών

Το πρώτο σύνολο δεδομένων περιλαμβάνει αναλυτικές πληροφορίες για κάθε ταξίδι που πραγματοποιήθηκε από τους οδηγούς. Κάθε εγγραφή περιέχει βασικά μεταδεδομένα όπως:

- **Μοναδικό Αναγνωριστικό Εταιρείας (companyId)**: Ταυτοποιεί την εταιρεία στην οποία ανήκει το όχημα.
- **Μοναδικό Αναγνωριστικό Χρήστη (userId)**: Ταυτοποιεί τον οδηγό.
- **Μοναδικό Αναγνωριστικό Εφαρμογής (applicationId)**: Συνδέει το συγκεκριμένο ταξίδι με την εφαρμογή που χρησιμοποιήθηκε για την καταγραφή του.

Επιπλέον, καταγράφονται χρονικές σφραγίδες (timestamps) για την έναρξη και τη λήξη του ταξιδιού, καθώς και γεωγραφικές συντεταγμένες για τα σημεία έναρξης και τερματισμού. Επίσης, παρέχονται δεδομένα όπως η συνολική απόσταση που διανύθηκε (σε χιλιόμετρα), η διάρκεια του ταξιδιού, καθώς και καταγραφές σημαντικών συμβάντων όπως έντονες επιταχύνσεις και απότομα φρεναρίσματα. Τέλος, παρέχει εκτιμήσεις για την ασφάλεια του ταξιδιού, τη φιλικότητα προς το περιβάλλον και την οικονομία καυσίμου.

Δεύτερο Αρχείο: Λεπτομερής Παρακολούθηση Οδηγικής Συμπεριφοράς

Το δεύτερο αρχείο εστιάζει στη λεπτομερή παρακολούθηση της οδηγικής συμπεριφοράς κατά τη διάρκεια των ταξιδιών. Καταγράφονται δεδομένα όπως ταχύτητα, επιταχύνσεις, γυροσκοπικά δεδομένα και γεωγραφικές συντεταγμένες. Αυτό το αρχείο επιπρόσθετα αναλύει και καταγράφει αναλυτικά τα συμβάντα που επηρεάζουν τη συμπεριφορά του

οχήματος, παρέχοντας πληροφορίες για τα αποτελέσματα σε κάθε φάση της οδήγησης. Συμπληρωματικά, δίνονται λεπτομέρειες για τις συσκευές που χρησιμοποιούνται για τη συλλογή των δεδομένων, καθώς και για τις στιγμές έναρξης και τερματισμού των ταξιδιών.

3.2 Σύνδεσμος προς το GitHub

Το notebook και το σύνολο δεδομένων που χρησιμοποιήθηκαν για την παρούσα πτυχιακή εργασία είναι διαθέσιμα στο GitHub. Μπορείτε να τα βρείτε στο ακόλουθο repository:

https://github.com/DimitrisKneknas/BachelorThesis

3.3 Περιγραφή του Κώδικα

3.3.1 Βιβλιοθήκες Python

```
from google.colab import drive
import os
import json
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
import copy
import statistics
import requests
import datetime
import ast
import optuna
from tabulate import tabulate

from xgboost import XGBRegressor, plot_tree
from matplotlib.pylab import rcParams
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, mean_absolute_error,
mean_absolute_percentage_error

import tensorflow as tf
from tensorflow.keras import layers
```

```
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.ensemble import VotingRegressor, StackingRegressor,
RandomForestRegressor, BaggingRegressor
from sklearn.linear_model import Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
```

Ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση του μοντέλου περιλαμβάνει τη χρήση αρκετών σημαντικών βιβλιοθηκών της Python, οι οποίες είναι ιδιαίτερα χρήσιμες για την ανάλυση δεδομένων, την οπτικοποίηση, τη μηχανική μάθηση, και τη βελτιστοποίηση. Παρακάτω παρουσιάζεται μια αναλυτική περιγραφή των κυριότερων βιβλιοθηκών και του ρόλου τους στη διαδικασία.

- **Pandas: Χειρισμός και Ανάλυση Δεδομένων**

  Το Pandas είναι μια εξαιρετικά ισχυρή βιβλιοθήκη για τον χειρισμό και την ανάλυση δεδομένων. Παρέχει δομές δεδομένων όπως τα DataFrames και οι Series, που επιτρέπουν την αποδοτική διαχείριση, συγχώνευση (merging), αναδιάρθρωση (reshaping) και καθαρισμό δεδομένων (cleaning data), διευκολύνοντας τη διαχείριση μεγάλων και περίπλοκων συνόλων δεδομένων.

- **Matplotlib και Seaborn: Οπτικοποίηση Δεδομένων**

  Το Matplotlib είναι μια ολοκληρωμένη βιβλιοθήκη για τη δημιουργία διαγραμμάτων και γραφημάτων. Σε συνδυασμό με το Seaborn, το οποίο προσφέρει υψηλού επιπέδου στατιστικά γραφικά, μπορούμε να δημιουργήσουμε ελκυστικές και κατανοητές οπτικοποιήσεις, οι οποίες διευκολύνουν την ερμηνεία των δεδομένων και των αποτελεσμάτων.

- **NumPy: Αριθμητικοί Υπολογισμοί**

  Η βιβλιοθήκη NumPy αποτελεί τη βάση για επιστημονικούς υπολογισμούς στην Python, προσφέροντας πολυδιάστατα arrays και εργαλεία για γρήγορες και αποδοτικές αριθμητικές πράξεις, συμπεριλαμβανομένων μαθηματικών και λογικών λειτουργιών, γραμμικής άλγεβρας, και τυχαίας προσομοίωσης.

- **Scikit-learn: Μηχανική Μάθηση και Προεπεξεργασία**

  Η βιβλιοθήκη Scikit-learn είναι ένα σημαντικό εργαλείο για την υλοποίηση αλγορίθμων μηχανικής μάθησης. Προσφέρει ευρύ φάσμα εργαλείων για την ταξινόμηση (classification), παλινδρόμηση (regression), ομαδοποίηση (clustering) και μείωση διαστάσεων (dimensionality reduction), καθώς και εργαλεία για την προεπεξεργασία δεδομένων και την αξιολόγηση μοντέλων. Επίσης, περιλαμβάνει μια ποικιλία από ensemble methods, οι οποίοι συνδυάζουν πολλαπλά μοντέλα για τη βελτίωση της ακρίβειας και της γενικευσιμότητας του τελικού μοντέλου.

- **Statistics: Στατιστικές Λειτουργίες**

  Η βιβλιοθήκη Statistics παρέχει βασικές στατιστικές συναρτήσεις, όπως τον υπολογισμό μέτρων κεντρικής τάσης (mean, median, mode) και μέτρων διασποράς (variance, standard deviation), διευκολύνοντας την ανάλυση των στατιστικών ιδιοτήτων των δεδομένων.

- **Requests: Διαχείριση HTTP Αιτημάτων**

  Η βιβλιοθήκη Requests επιτρέπει την αποστολή HTTP αιτημάτων με ευκολία, κάνοντας πιο απλή τη διαδικασία απόκτησης δεδομένων από διαδικτυακές πηγές μέσω των μεθόδων GET, POST, PUT και DELETE.

- **Optuna: Βελτιστοποίηση Υπερπαραμέτρων**

  Το Optuna είναι ένα εργαλείο βελτιστοποίησης υπερπαραμέτρων που αυτοματοποιεί την αναζήτηση των καλύτερων παραμέτρων για μοντέλα μηχανικής μάθησης, επιτρέποντας την αποδοτική βελτίωση της απόδοσης των μοντέλων.

- **XGBoost: Gradient Boosting**

  Το XGBoost είναι μια ισχυρή βιβλιοθήκη για την εφαρμογή του αλγορίθμου gradient boosting, η οποία υπερέχει σε αποδοτικότητα και επιδόσεις, ιδίως σε μεγάλα και πολύπλοκα σύνολα δεδομένων. Προσφέρει προηγμένες λειτουργίες όπως τακτοποίηση (regularization), cross-validation και πρόωρη διακοπή για την αποφυγή υπερεκπαίδευσης.

- **TensorFlow: Βαθιά Μάθηση**

  Το TensorFlow είναι ένα ισχυρό πλαίσιο για τη δημιουργία και εκπαίδευση μοντέλων βαθιάς μάθησης. Υποστηρίζει την ανάπτυξη σύνθετων αρχιτεκτονικών νευρωνικών δικτύων και μπορεί να εφαρμοστεί σε ένα ευρύ φάσμα εργασιών, από την ανάλυση εικόνας μέχρι την επεξεργασία φυσικής γλώσσας.

3.3.2 Ρυθμίσεις Περιβάλλοντος

```
pd.set_option('display.max_columns', None)
drive.mount('/content/drive')
```

Στο πλαίσιο της προετοιμασίας του περιβάλλοντος εργασίας, πραγματοποιήθηκαν ορισμένες ρυθμίσεις για τη διαχείριση και προβολή των δεδομένων καθώς και για την ενσωμάτωση εξωτερικών αρχείων από το Google Drive. Οι βασικές εντολές που χρησιμοποιήθηκαν είναι οι εξής:

- **pd.set_option('display.max_columns', None)**: Αυτή η εντολή εξασφαλίζει ότι όλες οι στήλες ενός DataFrame θα εμφανίζονται κατά την εμφάνισή του.
- **drive.mount('/content/drive')**: Συνδέει το Google Drive με το περιβάλλον Google Colab, επιτρέποντας την πρόσβαση στα αρχεία που είναι αποθηκευμένα στο Drive.

Αυτές οι ρυθμίσεις είναι απαραίτητες για την αποτελεσματική διαχείριση και επεξεργασία δεδομένων, καθώς και για την ενσωμάτωση εξωτερικών αρχείων που είναι αποθηκευμένα στο Google Drive.

3.3.3 Συνάρτηση calculate_statistics

```
def calculate_statistics(argument):
    if len(argument) > 1:
        mean_value = statistics.mean(argument)
        median_value = statistics.median(argument)
        mode_value = statistics.mode(argument)
        stdev_value = statistics.stdev(argument)
        variance_value = statistics.variance(argument)
        min_value = min(argument)
        max_value = max(argument)
    elif len(argument) == 1:
        mean_value = argument[0]
        median_value = argument[0]
```

```
        mode_value = argument[0]
        stdev_value = 0
        variance_value = 0
        min_value = argument[0]
        max_value = argument[0]
    else:
        mean_value = np.nan
        median_value = np.nan
        mode_value = np.nan
        stdev_value = np.nan
        variance_value = np.nan
        min_value = np.nan
        max_value = np.nan

    return mean_value, median_value, mode_value, stdev_value, variance_value,
min_value, max_value
```

Η συνάρτηση calculate_statistics παίρνει ως είσοδο μια λίστα αριθμητικών τιμών και υπολογίζει τα παρακάτω στατιστικά μέτρα:

- Μέσος Όρος (Mean): Ο αριθμητικός μέσος των τιμών στη λίστα.
- Διάμεσος (Median): Η μεσαία τιμή της ταξινομημένης λίστας.
- Επικρατούσα Τιμή (Mode): Η τιμή που εμφανίζεται πιο συχνά στη λίστα.
- Τυπική Απόκλιση (Standard Deviation): Ένα μέτρο της διασποράς των τιμών γύρω από τον μέσο όρο.
- Διακύμανση (Variance): Το τετράγωνο της τυπικής απόκλισης, εκφράζοντας τη μεταβλητότητα των δεδομένων.
- Ελάχιστη Τιμή (Min) και Μέγιστη Τιμή (Max): Οι ακραίες τιμές της λίστας.

Σε περίπτωση που η λίστα περιέχει μόνο ένα στοιχείο, όλα τα στατιστικά μέτρα ισούνται με αυτό το στοιχείο, με εξαίρεση την τυπική απόκλιση και τη διακύμανση που είναι μηδέν. Αν η λίστα είναι κενή, επιστρέφεται NaN για όλα τα μέτρα.

3.3.4 Συνάρτηση get_weather_by_coordinates_and_timestamp

```
def get_weather_by_coordinates_and_timestamp(lat, lon, timestamp):
    API_KEY = "0ee81ff49fcfecfa184feb152a7c8baa"
    BASE_URL = 'http://api.openweathermap.org/data/2.5/weather'
```

```
    params = {
        'lat': lat,
        'lon': lon,
        'appid': API_KEY,
        'units': 'metric',
        'dt': timestamp
    }

    response = requests.get(BASE_URL, params=params)
    print("response.status_code:", response.status_code)

    if response.status_code == 200:
        weather_data = response.json()
        return weather_data
    else:
        return None
```

Η συνάρτηση get_weather_by_coordinates_and_timestamp χρησιμοποιεί το OpenWeatherMap API για να ανακτήσει καιρικά δεδομένα με βάση συγκεκριμένες γεωγραφικές συντεταγμένες και χρονική στιγμή (timestamp). Τα βήματα περιλαμβάνουν:

- Προετοιμασία των παραμέτρων του αιτήματος (latitude, longitude, timestamp).
- Αποστολή του αιτήματος μέσω της βιβλιοθήκης requests.
- Επιστροφή των δεδομένων καιρού σε μορφή JSON αν το αίτημα είναι επιτυχές, διαφορετικά επιστρέφεται None.

3.3.5 Συνάρτηση date_to_timestamp

```
def date_to_timestamp(date):
    parsed_date = datetime.datetime.fromisoformat(date)
    utc_date = parsed_date.astimezone(datetime.timezone.utc)
    return int(utc_date.timestamp())
```

Η συνάρτηση date_to_timestamp παίρνει ως είσοδο μια ημερομηνία σε μορφή ISO (YYYY-MM-DDTHH:MM) και την μετατρέπει σε timestamp. Η διαδικασία περιλαμβάνει:

- Ανάλυση της ημερομηνίας σε αντικείμενο datetime.
- Μετατροπή της ημερομηνίας σε UTC.
- Δημιουργία και επιστροφή του timestamp που αντιστοιχεί στην ημερομηνία.

3.3.6 Συνάρτηση calculate_weather_statistics

```
def calculate_weather_statistics(trip_events_weather,
trip_events_weather_statistics_list, name_print = "trip_events"):
    weather_main_temp_list = [weather_data['main']['temp'] for weather_data in
trip_events_weather]
    mean_value1, median_value1, mode_value1, stdev_value1, variance_value1,
min_value1, max_value1 = calculate_statistics(weather_main_temp_list)

    weather_main_feels_like_list = [weather_data['main']['feels_like'] for
weather_data in trip_events_weather]
    mean_value2, median_value2, mode_value2, stdev_value2, variance_value2,
min_value2, max_value2 = calculate_statistics(weather_main_feels_like_list)

    weather_main_temp_min_list = [weather_data['main']['temp_min'] for
weather_data in trip_events_weather]
    mean_value3, median_value3, mode_value3, stdev_value3, variance_value3,
min_value3, max_value3 = calculate_statistics(weather_main_temp_min_list)


    weather_main_temp_max_list = [weather_data['main']['temp_max'] for
weather_data in trip_events_weather]
    mean_value4, median_value4, mode_value4, stdev_value4, variance_value4,
min_value4, max_value4 = calculate_statistics(weather_main_temp_max_list)


    weather_main_pressure_list = [weather_data['main']['pressure'] for
weather_data in trip_events_weather]
    mean_value5, median_value5, mode_value5, stdev_value5, variance_value5,
min_value5, max_value5 = calculate_statistics(weather_main_pressure_list)


    weather_main_humidity_list = [weather_data['main']['humidity'] for
weather_data in trip_events_weather]
    mean_value6, median_value6, mode_value6, stdev_value6, variance_value6,
min_value6, max_value6 = calculate_statistics(weather_main_humidity_list)


    weather_visibility_list = [weather_data['visibility'] for weather_data in
trip_events_weather]
    mean_value7, median_value7, mode_value7, stdev_value7, variance_value7,
min_value7, max_value7 = calculate_statistics(weather_visibility_list)


    weather_wind_speed_list = [weather_data['wind']['speed'] for weather_data
in trip_events_weather]
    mean_value8, median_value8, mode_value8, stdev_value8, variance_value8,
min_value8, max_value8 = calculate_statistics(weather_wind_speed_list)
```

```
    weather_wind_deg_list = [weather_data['wind']['deg'] for weather_data in
trip_events_weather]
    mean_value9, median_value9, mode_value9, stdev_value9, variance_value9,
min_value9, max_value9 = calculate_statistics(weather_wind_deg_list)


    weather_clouds_all_list = [weather_data['clouds']['all'] for weather_data
in trip_events_weather]
    mean_value10, median_value10, mode_value10, stdev_value10,
variance_value10, min_value10, max_value10 =
calculate_statistics(weather_clouds_all_list)


    weather_sys_sunrise_list = [weather_data['sys']['sunrise'] for weather_data
in trip_events_weather]
    mean_value11, median_value11, mode_value11, stdev_value11,
variance_value11, min_value11, max_value11 =
calculate_statistics(weather_sys_sunrise_list)


    weather_sys_sunset_list = [weather_data['sys']['sunset'] for weather_data
in trip_events_weather]
    mean_value12, median_value12, mode_value12, stdev_value12,
variance_value12, min_value12, max_value12 =
calculate_statistics(weather_sys_sunset_list)

    weather_main_list = [weather_data['weather'][0]['main'] for weather_data in
trip_events_weather]
    weather_main_counts = {}
    for weather_main in weather_main_list:
        if weather_main is not None:
            if weather_main in weather_main_counts:
                weather_main_counts[weather_main] += 1
            else:
                weather_main_counts[weather_main] = 1


    trip_events_weather_statistics_list.append({
        f'Mean_temp_{name_print}_weather': mean_value1,
        f'Median_temp_{name_print}_weather': median_value1,
        f'Mode_temp_{name_print}_weather': mode_value1,
        f'Standard_Deviation_temp_{name_print}_weather': stdev_value1,
        f'Variance_temp_{name_print}_weather': variance_value1,
        f'Min_temp_{name_print}_weather': min_value1,
        f'Max_temp_{name_print}_weather': max_value1,


        f'Mean_feels_like_{name_print}_weather': mean_value2,
        f'Median_feels_like_{name_print}_weather': median_value2,
        f'Mode_feels_like_{name_print}_weather': mode_value2,
```

```
        f'Standard_Deviation_feels_like_{name_print}_weather': stdev_value2,
        f'Variance_feels_like_{name_print}_weather': variance_value2,
        f'Min_feels_like_{name_print}_weather': min_value2,
        f'Max_feels_like_{name_print}_weather': max_value2,


        f'Mean_Minimum_temp_{name_print}_weather': mean_value3,
        f'Median_Minimum_temp_{name_print}_weather': median_value3,
        f'Mode_Minimum_temp_{name_print}_weather': mode_value3,
        f'Standard_Deviation_Minimum_temp_{name_print}_weather': stdev_value3,
        f'Variance_Minimum_temp_{name_print}_weather': variance_value3,
        f'Min_Minimum_temp_{name_print}_weather': min_value3,
        f'Max_Minimum_temp_{name_print}_weather': max_value3,


        f'Mean_Maximum_temp_{name_print}_weather': mean_value4,
        f'Median_Maximum_temp_{name_print}_weather': median_value4,
        f'Mode_Maximum_temp_{name_print}_weather': mode_value4,
        f'Standard_Deviation_Maximum_temp_{name_print}_weather': stdev_value4,
        f'Variance_Maximum_temp_{name_print}_weather': variance_value4,
        f'Min_Maximum_temp_{name_print}_weather': min_value4,
        f'Max_Maximum_temp_{name_print}_weather': max_value4,


        f'Mean_Pressure_{name_print}_weather': mean_value5,
        f'Median_Pressure_{name_print}_weather': median_value5,
        f'Mode_Pressure_{name_print}_weather': mode_value5,
        f'Standard_Deviation_Pressure_{name_print}_weather': stdev_value5,
        f'Variance_Pressure_{name_print}_weather': variance_value5,
        f'Min_Pressure_{name_print}_weather': min_value5,
        f'Max_Pressure_{name_print}_weather': max_value5,


        f'Mean_Humidity_{name_print}_weather': mean_value6,
        f'Median_Humidity_{name_print}_weather': median_value6,
        f'Mode_Humidity_{name_print}_weather': mode_value6,
        f'Standard_Deviation_Humidity_{name_print}_weather': stdev_value6,
        f'Variance_Humidity_{name_print}_weather': variance_value6,
        f'Min_Humidity_{name_print}_weather': min_value6,
        f'Max_Humidity_{name_print}_weather': max_value6,


        f'Mean_Visibility_{name_print}_weather': mean_value7,
        f'Median_Visibility_{name_print}_weather': median_value7,
        f'Mode_Visibility_{name_print}_weather': mode_value7,
        f'Standard_Deviation_Visibility_{name_print}_weather': stdev_value7,
        f'Variance_Visibility_{name_print}_weather': variance_value7,
        f'Min_Visibility_{name_print}_weather': min_value7,
        f'Max_Visibility_{name_print}_weather': max_value7,
```

```
        f'Mean_Wind_Speed_{name_print}_weather': mean_value8,
        f'Median_Wind_Speed_{name_print}_weather': median_value8,
        f'Mode_Wind_Speed_{name_print}_weather': mode_value8,
        f'Standard_Deviation_Wind_Speed_{name_print}_weather': stdev_value8,
        f'Variance_Wind_Speed_{name_print}_weather': variance_value8,
        f'Min_Wind_Speed_{name_print}_weather': min_value8,
        f'Max_Wind_Speed_{name_print}_weather': max_value8,


        f'Mean_Wind_Direction_{name_print}_weather': mean_value9,
        f'Median_Wind_Direction_{name_print}_weather': median_value9,
        f'Mode_Wind_Direction_{name_print}_weather': mode_value9,
        f'Standard_Deviation_Wind_Direction_{name_print}_weather':
stdev_value9,
        f'Variance_Wind_Direction_{name_print}_weather': variance_value9,
        f'Min_Wind_Direction_{name_print}_weather': min_value9,
        f'Max_Wind_Direction_{name_print}_weather': max_value9,


        f'Mean_Cloudiness_Percentage_{name_print}_weather': mean_value10,
        f'Median_Cloudiness_Percentage_{name_print}_weather': median_value10,
        f'Mode_Cloudiness_Percentage_{name_print}_weather': mode_value10,
        f'Standard_Deviation_Cloudiness_Percentage_{name_print}_weather':
stdev_value10,
        f'Variance_Cloudiness_Percentage_{name_print}_weather':
variance_value10,
        f'Min_Cloudiness_Percentage_{name_print}_weather': min_value10,
        f'Max_Cloudiness_Percentage_{name_print}_weather': max_value10,


        f'Mean_Sunrise_Time_{name_print}_weather': mean_value11,
        f'Median_Sunrise_Time_{name_print}_weather': median_value11,
        f'Mode_Sunrise_Time_{name_print}_weather': mode_value11,
        f'Standard_Deviation_Sunrise_Time_{name_print}_weather': stdev_value11,
        f'Variance_Sunrise_Time_{name_print}_weather': variance_value11,
        f'Min_Sunrise_Time_{name_print}_weather': min_value11,
        f'Max_Sunrise_Time_{name_print}_weather': max_value11,


        f'Mean_Sunset_Time_{name_print}_weather': mean_value12,
        f'Median_Sunset_Time_{name_print}_weather': median_value12,
        f'Mode_Sunset_Time_{name_print}_weather': mode_value12,
        f'Standard_Deviation_Sunset_Time_{name_print}_weather': stdev_value12,
        f'Variance_Sunset_Time_{name_print}_weather': variance_value12,
        f'Min_Sunset_Time_{name_print}_weather': min_value12,
        f'Max_Sunset_Time_{name_print}_weather': max_value12,

        f'Weather_Main_{name_print}_weather': weather_main_counts
    })
```

```
    return trip_events_weather_statistics_list
```

Η συνάρτηση calculate_weather_statistics υπολογίζει διάφορα στατιστικά μέτρα για τα καιρικά δεδομένα που σχετίζονται με τα γεγονότα ταξιδιού (trip events). Τα καιρικά δεδομένα περιλαμβάνουν πληροφορίες όπως θερμοκρασία (temperature), πίεση (pressure), υγρασία (humidity), ταχύτητα ανέμου (wind speed), και άλλα. Τα κύρια βήματα είναι:

- **Δημιουργία Λιστών Δεδομένων**: Για κάθε κατηγορία καιρικών δεδομένων, δημιουργούνται λίστες που περιέχουν τις αντίστοιχες τιμές από τα δεδομένα των γεγονότων ταξιδιού.
- **Υπολογισμός Στατιστικών**: Για κάθε λίστα δεδομένων, καλείται η συνάρτηση calculate_statistics για να υπολογίσει τα βασικά στατιστικά μέτρα (Mean, Median, Mode, Standard Deviation, Variance, Min, Max).
- **Αποθήκευση Στατιστικών**: Τα υπολογισμένα στατιστικά μέτρα αποθηκεύονται σε ένα λεξικό (dictionary), το οποίο στη συνέχεια προστίθεται σε μια λίστα που περιέχει όλα τα στατιστικά δεδομένα του καιρού για τα γεγονότα ταξιδιού.

3.3.7 Συνάρτηση calculate_raw_events_statistics

```
def calculate_raw_events_statistics(raw_events, raw_data_list, raw_startReason,
raw_stopReason):

    if raw_data_list is None:
        raw_data_list = []

    acceleration_direct_values = [event["AccelerationDirect"] for event in
raw_events if event["AccelerationDirect"] is not None]
    mean_value1, median_value1, mode_value1, stdev_value1, variance_value1,
min_value1, max_value1 = calculate_statistics(acceleration_direct_values)


    acceleration_direct_end_values = [event["AccelerationDirectEnd"] for event
in raw_events if event["AccelerationDirectEnd"] is not None]
    mean_value2, median_value2, mode_value2, stdev_value2, variance_value2,
min_value2, max_value2 = calculate_statistics(acceleration_direct_end_values)


    acceleration_lateral_values = [event["AccelerationLateral"] for event in
```

```
raw_events if event["AccelerationLateral"] is not None]
    mean_value3, median_value3, mode_value3, stdev_value3, variance_value3,
min_value3, max_value3 = calculate_statistics(acceleration_lateral_values)


    acceleration_lateral_end_values = [event["AccelerationLateralEnd"] for
event in raw_events if event["AccelerationLateralEnd"] is not None]
    mean_value4, median_value4, mode_value4, stdev_value4, variance_value4,
min_value4, max_value4 = calculate_statistics(acceleration_lateral_end_values)


    acceleration_vertical_values = [event["AccelerationVertical"] for event in
raw_events if event["AccelerationVertical"] is not None]
    mean_value5, median_value5, mode_value5, stdev_value5, variance_value5,
min_value5, max_value5 = calculate_statistics(acceleration_vertical_values)


    acceleration_vertical_end_values = [event["AccelerationVerticalEnd"] for
event in raw_events if event["AccelerationVerticalEnd"] is not None]
    mean_value6, median_value6, mode_value6, stdev_value6, variance_value6,
min_value6, max_value6 = calculate_statistics(acceleration_vertical_end_values)


    duration_values = [event["Duration"] for event in raw_events if
event["Duration"] is not None]
    mean_value7, median_value7, mode_value7, stdev_value7, variance_value7,
min_value7, max_value7 = calculate_statistics(duration_values)


    prev_event_speed_values = [event["PrevEventSpeed"] for event in raw_events
if event["PrevEventSpeed"] is not None]
    mean_value8, median_value8, mode_value8, stdev_value8, variance_value8,
min_value8, max_value8 = calculate_statistics(prev_event_speed_values)


    pure_duration_values = [event["PureDuration"] for event in raw_events if
event["PureDuration"] is not None]
    mean_value9, median_value9, mode_value9, stdev_value9, variance_value9,
min_value9, max_value9 = calculate_statistics(pure_duration_values)


    reliability_values = [event["Reliability"] for event in raw_events if
event["Reliability"] is not None and event["Reliability"] >= 0]
    mean_value10, median_value10, mode_value10, stdev_value10,
variance_value10, min_value10, max_value10 =
calculate_statistics(reliability_values)


    speed_median_values = [event["SpeedMedian"] for event in raw_events if
event["SpeedMedian"] is not None]
```

```
    mean_value11, median_value11, mode_value11, stdev_value11,
variance_value11, min_value11, max_value11 =
calculate_statistics(speed_median_values)


    speed_start_values = [event["SpeedStart"] for event in raw_events if
event["SpeedStart"] is not None]
    mean_value12, median_value12, mode_value12, stdev_value12,
variance_value12, min_value12, max_value12 =
calculate_statistics(speed_start_values)


    speed_stop_values = [event["SpeedStop"] for event in raw_events if
event["SpeedStop"] is not None]
    mean_value13, median_value13, mode_value13, stdev_value13,
variance_value13, min_value13, max_value13 =
calculate_statistics(speed_stop_values)


    accuracy_max_values = [event['Accuracy']['max'] for event in raw_events if
event['Accuracy']['max'] is not None]
    mean_value14, median_value14, mode_value14, stdev_value14,
variance_value14, min_value14, max_value14 =
calculate_statistics(accuracy_max_values)


    accuracy_median_values = [event['Accuracy']['median'] for event in
raw_events if event['Accuracy']['median'] is not None]
    mean_value15, median_value15, mode_value15, stdev_value15,
variance_value15, min_value15, max_value15 =
calculate_statistics(accuracy_median_values)


    accuracy_min_values = [event['Accuracy']['min'] for event in raw_events if
event['Accuracy']['min'] is not None]
    mean_value16, median_value16, mode_value16, stdev_value16,
variance_value16, min_value16, max_value16 =
calculate_statistics(accuracy_min_values)


    accuracy_quantile_05_values = [event['Accuracy']['quantile_05'] for event
in raw_events if event['Accuracy']['quantile_05'] is not None]
    mean_value17, median_value17, mode_value17, stdev_value17,
variance_value17, min_value17, max_value17 =
calculate_statistics(accuracy_quantile_05_values)


    accuracy_quantile_95_values = [event['Accuracy']['quantile_95'] for event
in raw_events if event['Accuracy']['quantile_95'] is not None]
    mean_value18, median_value18, mode_value18, stdev_value18,
```

```
variance_value18, min_value18, max_value18 =
calculate_statistics(accuracy_quantile_95_values)


    range_direct_max_values = [event['RangeDirect']['max'] for event in
raw_events if event['RangeDirect']['max'] is not None]
    mean_value19, median_value19, mode_value19, stdev_value19,
variance_value19, min_value19, max_value19 =
calculate_statistics(range_direct_max_values)


    range_direct_median_values = [event['RangeDirect']['median'] for event in
raw_events if event['RangeDirect']['median'] is not None]
    mean_value20, median_value20, mode_value20, stdev_value20,
variance_value20, min_value20, max_value20 =
calculate_statistics(range_direct_median_values)


    range_direct_min_values = [event['RangeDirect']['min'] for event in
raw_events if event['RangeDirect']['min'] is not None]
    mean_value21, median_value21, mode_value21, stdev_value21,
variance_value21, min_value21, max_value21 =
calculate_statistics(range_direct_min_values)


    range_direct_quantile_05_values = [event['RangeDirect']['quantile_05'] for
event in raw_events if event['RangeDirect']['quantile_05'] is not None]
    mean_value22, median_value22, mode_value22, stdev_value22,
variance_value22, min_value22, max_value22 =
calculate_statistics(range_direct_quantile_05_values)


    range_direct_quantile_95_values = [event['RangeDirect']['quantile_95'] for
event in raw_events if event['RangeDirect']['quantile_95'] is not None]
    mean_value23, median_value23, mode_value23, stdev_value23,
variance_value23, min_value23, max_value23 =
calculate_statistics(range_direct_quantile_95_values)


    range_lateral_max_values = [event['RangeLateral']['max'] for event in
raw_events if event['RangeLateral']['max'] is not None]
    mean_value24, median_value24, mode_value24, stdev_value24,
variance_value24, min_value24, max_value24 =
calculate_statistics(range_lateral_max_values)


    range_lateral_median_values = [event['RangeLateral']['median'] for event in
raw_events if event['RangeLateral']['median'] is not None]
    mean_value25, median_value25, mode_value25, stdev_value25,
variance_value25, min_value25, max_value25 =
```

```
calculate_statistics(range_lateral_median_values)


    range_lateral_min_values = [event['RangeLateral']['min'] for event in
raw_events if event['RangeLateral']['min'] is not None]
    mean_value26, median_value26, mode_value26, stdev_value26,
variance_value26, min_value26, max_value26 =
calculate_statistics(range_lateral_min_values)


    range_lateral_quantile_05_values = [event['RangeLateral']['quantile_05']
for event in raw_events if event['RangeLateral']['quantile_05'] is not None]
    mean_value27, median_value27, mode_value27, stdev_value27,
variance_value27, min_value27, max_value27 =
calculate_statistics(range_lateral_quantile_05_values)


    range_lateral_quantile_95_values = [event['RangeLateral']['quantile_95']
for event in raw_events if event['RangeLateral']['quantile_95'] is not None]
    mean_value28, median_value28, mode_value28, stdev_value28,
variance_value28, min_value28, max_value28 =
calculate_statistics(range_lateral_quantile_95_values)


    speed_max_values = [event['Speed']['max'] for event in raw_events if
event['Speed']['max'] is not None]
    mean_value29, median_value29, mode_value29, stdev_value29,
variance_value29, min_value29, max_value29 =
calculate_statistics(speed_max_values)


    speed_median_values = [event['Speed']['median'] for event in raw_events if
event['Speed']['median'] is not None]
    mean_value30, median_value30, mode_value30, stdev_value30,
variance_value30, min_value30, max_value30 =
calculate_statistics(speed_median_values)


    speed_min_values = [event['Speed']['min'] for event in raw_events if
event['Speed']['min'] is not None]
    mean_value31, median_value31, mode_value31, stdev_value31,
variance_value31, min_value31, max_value31 =
calculate_statistics(speed_min_values)


    speed_quantile_05_values = [event['Speed']['quantile_05'] for event in
raw_events if event['Speed']['quantile_05'] is not None]
    mean_value32, median_value32, mode_value32, stdev_value32,
variance_value32, min_value32, max_value32 =
calculate_statistics(speed_quantile_05_values)
```

```
    speed_quantile_95_values = [event['Speed']['quantile_95'] for event in
raw_events if event['Speed']['quantile_95'] is not None]
    mean_value33, median_value33, mode_value33, stdev_value33,
variance_value33, min_value33, max_value33 =
calculate_statistics(speed_quantile_95_values)


    range_vertical_max_values = [event['RangeVertical']['max'] for event in
raw_events if event['RangeVertical']['max'] is not None]
    mean_value34, median_value34, mode_value34, stdev_value34,
variance_value34, min_value34, max_value34 =
calculate_statistics(range_vertical_max_values)


    range_vertical_median_values = [event['RangeVertical']['median'] for event
in raw_events if event['RangeVertical']['median'] is not None]
    mean_value35, median_value35, mode_value35, stdev_value35,
variance_value35, min_value35, max_value35 =
calculate_statistics(range_vertical_median_values)


    range_vertical_min_values = [event['RangeVertical']['min'] for event in
raw_events if event['RangeVertical']['min'] is not None]
    mean_value36, median_value36, mode_value36, stdev_value36,
variance_value36, min_value36, max_value36 =
calculate_statistics(range_vertical_min_values)


    range_vertical_quantile_05_values = [event['RangeVertical']['quantile_05']
for event in raw_events if event['RangeVertical']['quantile_05'] is not None]
    mean_value37, median_value37, mode_value37, stdev_value37,
variance_value37, min_value37, max_value37 =
calculate_statistics(range_vertical_quantile_05_values)


    range_vertical_quantile_95_values = [event['RangeVertical']['quantile_95']
for event in raw_events if event['RangeVertical']['quantile_95'] is not None]
    mean_value38, median_value38, mode_value38, stdev_value38,
variance_value38, min_value38, max_value38 =
calculate_statistics(range_vertical_quantile_95_values)



    raw_data_list.append({
        'Raw_startReason': raw_startReason,
        'Raw_stopReason': raw_stopReason,
```

```
        'Mean_acceleration_direct_values': mean_value1,
        'Median_acceleration_direct_values': median_value1,
        'Mode_acceleration_direct_values': mode_value1,
        'Standard_Deviation_acceleration_direct_values': stdev_value1,
        'Variance_acceleration_direct_values': variance_value1,
        'Min_acceleration_direct_values': min_value1,
        'Max_acceleration_direct_values': max_value1,


        'Mean_acceleration_direct_end_values': mean_value2,
        'Median_acceleration_direct_end_values': median_value2,
        'Mode_acceleration_direct_end_values': mode_value2,
        'Standard_Deviation_acceleration_direct_end_values': stdev_value2,
        'Variance_acceleration_direct_end_values': variance_value2,
        'Min_acceleration_direct_end_values': min_value2,
        'Max_acceleration_direct_end_values': max_value2,


        'Mean_acceleration_lateral_values': mean_value3,
        'Median_acceleration_lateral_values': median_value3,
        'Mode_acceleration_lateral_values': mode_value3,
        'Standard_Deviation_acceleration_lateral_values': stdev_value3,
        'Variance_acceleration_lateral_values': variance_value3,
        'Min_acceleration_lateral_values': min_value3,
        'Max_acceleration_lateral_values': max_value3,


        'Mean_acceleration_lateral_end_values': mean_value4,
        'Median_acceleration_lateral_end_values': median_value4,
        'Mode_acceleration_lateral_end_values': mode_value4,
        'Standard_Deviation_acceleration_lateral_end_values': stdev_value4,
        'Variance_acceleration_lateral_end_values': variance_value4,
        'Min_acceleration_lateral_end_values': min_value4,
        'Max_acceleration_lateral_end_values': max_value4,


        'Mean_acceleration_vertical_values': mean_value5,
        'Median_acceleration_vertical_values': median_value5,
        'Mode_acceleration_vertical_values': mode_value5,
        'Standard_Deviation_acceleration_vertical_values': stdev_value5,
        'Variance_acceleration_vertical_values': variance_value5,
        'Min_acceleration_vertical_values': min_value5,
        'Max_acceleration_vertical_values': max_value5,


        'Mean_acceleration_vertical_end_values': mean_value6,
        'Median_acceleration_vertical_end_values': median_value6,
        'Mode_acceleration_vertical_end_values': mode_value6,
        'Standard_Deviation_acceleration_vertical_end_values': stdev_value6,
        'Variance_acceleration_vertical_end_values': variance_value6,
```

```
        'Min_acceleration_vertical_end_values': min_value6,
        'Max_acceleration_vertical_end_values': max_value6,


        'Mean_Duration_values': mean_value7,
        'Median_Duration_values': median_value7,
        'Mode_Duration_values': mode_value7,
        'Standard_Deviation_Duration_values': stdev_value7,
        'Variance_Duration_values': variance_value7,
        'Min_Duration_values': min_value7,
        'Max_Duration_values': max_value7,


        'Mean_PrevEventSpeed_values': mean_value8,
        'Median_PrevEventSpeed_values': median_value8,
        'Mode_PrevEventSpeed_values': mode_value8,
        'Standard_Deviation_PrevEventSpeed_values': stdev_value8,
        'Variance_PrevEventSpeed_values': variance_value8,
        'Min_PrevEventSpeed_values': min_value8,
        'Max_PrevEventSpeed_values': max_value8,


        'Mean_PureDuration_values': mean_value9,
        'Median_PureDuration_values': median_value9,
        'Mode_PureDuration_values': mode_value9,
        'Standard_Deviation_PureDuration_values': stdev_value9,
        'Variance_PureDuration_values': variance_value9,
        'Min_PureDuration_values': min_value9,
        'Max_PureDuration_values': max_value9,


        'Mean_Reliability_values': mean_value10,
        'Median_Reliability_values': median_value10,
        'Mode_Reliability_values': mode_value10,
        'Standard_Deviation_Reliability_values': stdev_value10,
        'Variance_Reliability_values': variance_value10,
        'Min_Reliability_values': min_value10,
        'Max_Reliability_values': max_value10,


        'Mean_SpeedMedian_values': mean_value11,
        'Median_SpeedMedian_values': median_value11,
        'Mode_SpeedMedian_values': mode_value11,
        'Standard_Deviation_SpeedMedian_values': stdev_value11,
        'Variance_SpeedMedian_values': variance_value11,
        'Min_SpeedMedian_values': min_value11,
        'Max_SpeedMedian_values': max_value11,


        'Mean_SpeedStart_values': mean_value12,
```

```
        'Median_SpeedStart_values': median_value12,
        'Mode_SpeedStart_values': mode_value12,
        'Standard_Deviation_SpeedStart_values': stdev_value12,
        'Variance_SpeedStart_values': variance_value12,
        'Min_SpeedStart_values': min_value12,
        'Max_SpeedStart_values': max_value12,


        'Mean_SpeedStop_values': mean_value13,
        'Median_SpeedStop_values': median_value13,
        'Mode_SpeedStop_values': mode_value13,
        'Standard_Deviation_SpeedStop_values': stdev_value13,
        'Variance_SpeedStop_values': variance_value13,
        'Min_SpeedStop_values': min_value13,
        'Max_SpeedStop_values': max_value13,


        'Mean_Accuracy_max_values': mean_value14,
        'Median_Accuracy_max_values': median_value14,
        'Mode_Accuracy_max_values': mode_value14,
        'Standard_Deviation_Accuracy_max_values': stdev_value14,
        'Variance_Accuracy_max_values': variance_value14,
        'Min_Accuracy_max_values': min_value14,
        'Max_Accuracy_max_values': max_value14,


        'Mean_Accuracy_median_values': mean_value15,
        'Median_Accuracy_median_values': median_value15,
        'Mode_Accuracy_median_values': mode_value15,
        'Standard_Deviation_Accuracy_median_values': stdev_value15,
        'Variance_Accuracy_median_values': variance_value15,
        'Min_Accuracy_median_values': min_value15,
        'Max_Accuracy_median_values': max_value15,


        'Mean_Accuracy_min_values': mean_value16,
        'Median_Accuracy_min_values': median_value16,
        'Mode_Accuracy_min_values': mode_value16,
        'Standard_Deviation_Accuracy_min_values': stdev_value16,
        'Variance_Accuracy_min_values': variance_value16,
        'Min_Accuracy_min_values': min_value16,
        'Max_Accuracy_min_values': max_value16,


        'Mean_Accuracy_quantile_05_values': mean_value17,
        'Median_Accuracy_quantile_05_values': median_value17,
        'Mode_Accuracy_quantile_05_values': mode_value17,
        'Standard_Deviation_Accuracy_quantile_05_values': stdev_value17,
        'Variance_Accuracy_quantile_05_values': variance_value17,
        'Min_Accuracy_quantile_05_values': min_value17,
```

```
        'Max_Accuracy_quantile_05_values': max_value17,


        'Mean_Accuracy_quantile_95_values': mean_value18,
        'Median_Accuracy_quantile_95_values': median_value18,
        'Mode_Accuracy_quantile_95_values': mode_value18,
        'Standard_Deviation_Accuracy_quantile_95_values': stdev_value18,
        'Variance_Accuracy_quantile_95_values': variance_value18,
        'Min_Accuracy_quantile_95_values': min_value18,
        'Max_Accuracy_quantile_95_values': max_value18,


        'Mean_RangeDirect_max_values': mean_value19,
        'Median_RangeDirect_max_values': median_value19,
        'Mode_RangeDirect_max_values': mode_value19,
        'Standard_Deviation_RangeDirect_max_values': stdev_value19,
        'Variance_RangeDirect_max_values': variance_value19,
        'Min_RangeDirect_max_values': min_value19,
        'Max_RangeDirect_max_values': max_value19,


        'Mean_RangeDirect_median_values': mean_value20,
        'Median_RangeDirect_median_values': median_value20,
        'Mode_RangeDirect_median_values': mode_value20,
        'Standard_Deviation_RangeDirect_median_values': stdev_value20,
        'Variance_RangeDirect_median_values': variance_value20,
        'Min_RangeDirect_median_values': min_value20,
        'Max_RangeDirect_median_values': max_value20,


        'Mean_RangeDirect_min_values': mean_value21,
        'Median_RangeDirect_min_values': median_value21,
        'Mode_RangeDirect_min_values': mode_value21,
        'Standard_Deviation_RangeDirect_min_values': stdev_value21,
        'Variance_RangeDirect_min_values': variance_value21,
        'Min_RangeDirect_min_values': min_value21,
        'Max_RangeDirect_min_values': max_value21,


        'Mean_RangeDirect_quantile_05_values': mean_value22,
        'Median_RangeDirect_quantile_05_values': median_value22,
        'Mode_RangeDirect_quantile_05_values': mode_value22,
        'Standard_Deviation_RangeDirect_quantile_05_values': stdev_value22,
        'Variance_RangeDirect_quantile_05_values': variance_value22,
        'Min_RangeDirect_quantile_05_values': min_value22,
        'Max_RangeDirect_quantile_05_values': max_value22,


        'Mean_RangeDirect_quantile_95_values': mean_value23,
        'Median_RangeDirect_quantile_95_values': median_value23,
```

```
        'Mode_RangeDirect_quantile_95_values': mode_value23,
        'Standard_Deviation_RangeDirect_quantile_95_values': stdev_value23,
        'Variance_RangeDirect_quantile_95_values': variance_value23,
        'Min_RangeDirect_quantile_95_values': min_value23,
        'Max_RangeDirect_quantile_95_values': max_value23,


        'Mean_RangeLateral_max_values': mean_value24,
        'Median_RangeLateral_max_values': median_value24,
        'Mode_RangeLateral_max_values': mode_value24,
        'Standard_Deviation_RangeLateral_max_values': stdev_value24,
        'Variance_RangeLateral_max_values': variance_value24,
        'Min_RangeLateral_max_values': min_value24,
        'Max_RangeLateral_max_values': max_value24,


        'Mean_RangeLateral_median_values': mean_value25,
        'Median_RangeLateral_median_values': median_value25,
        'Mode_RangeLateral_median_values': mode_value25,
        'Standard_Deviation_RangeLateral_median_values': stdev_value25,
        'Variance_RangeLateral_median_values': variance_value25,
        'Min_RangeLateral_median_values': min_value25,
        'Max_RangeLateral_median_values': max_value25,


        'Mean_RangeLateral_min_values': mean_value26,
        'Median_RangeLateral_min_values': median_value26,
        'Mode_RangeLateral_min_values': mode_value26,
        'Standard_Deviation_RangeLateral_min_values': stdev_value26,
        'Variance_RangeLateral_min_values': variance_value26,
        'Min_RangeLateral_min_values': min_value26,
        'Max_RangeLateral_min_values': max_value26,


        'Mean_RangeLateral_quantile_05_values': mean_value27,
        'Median_RangeLateral_quantile_05_values': median_value27,
        'Mode_RangeLateral_quantile_05_values': mode_value27,
        'Standard_Deviation_RangeLateral_quantile_05_values': stdev_value27,
        'Variance_RangeLateral_quantile_05_values': variance_value27,
        'Min_RangeLateral_quantile_05_values': min_value27,
        'Max_RangeLateral_quantile_05_values': max_value27,


        'Mean_RangeLateral_quantile_95_values': mean_value28,
        'Median_RangeLateral_quantile_95_values': median_value28,
        'Mode_RangeLateral_quantile_95_values': mode_value28,
        'Standard_Deviation_RangeLateral_quantile_95_values': stdev_value28,
        'Variance_RangeLateral_quantile_95_values': variance_value28,
        'Min_RangeLateral_quantile_95_values': min_value28,
        'Max_RangeLateral_quantile_95_values': max_value28,
```

```
    'Mean_Speed_max_values': mean_value29,
    'Median_Speed_max_values': median_value29,
    'Mode_Speed_max_values': mode_value29,
    'Standard_Deviation_Speed_max_values': stdev_value29,
    'Variance_Speed_max_values': variance_value29,
    'Min_Speed_max_values': min_value29,
    'Max_Speed_max_values': max_value29,


    'Mean_Speed_median_values': mean_value30,
    'Median_Speed_median_values': median_value30,
    'Mode_Speed_median_values': mode_value30,
    'Standard_Deviation_Speed_median_values': stdev_value30,
    'Variance_Speed_median_values': variance_value30,
    'Min_Speed_median_values': min_value30,
    'Max_Speed_median_values': max_value30,


    'Mean_Speed_min_values': mean_value31,
    'Median_Speed_min_values': median_value31,
    'Mode_Speed_min_values': mode_value31,
    'Standard_Deviation_Speed_min_values': stdev_value31,
    'Variance_Speed_min_values': variance_value31,
    'Min_Speed_min_values': min_value31,
    'Max_Speed_min_values': max_value31,


    'Mean_Speed_quantile_05_values': mean_value32,
    'Median_Speed_quantile_05_values': median_value32,
    'Mode_Speed_quantile_05_values': mode_value32,
    'Standard_Deviation_Speed_quantile_05_values': stdev_value32,
    'Variance_Speed_quantile_05_values': variance_value32,
    'Min_Speed_quantile_05_values': min_value32,
    'Max_Speed_quantile_05_values': max_value32,


    'Mean_Speed_quantile_95_values': mean_value33,
    'Median_Speed_quantile_95_values': median_value33,
    'Mode_Speed_quantile_95_values': mode_value33,
    'Standard_Deviation_Speed_quantile_95_values': stdev_value33,
    'Variance_Speed_quantile_95_values': variance_value33,
    'Min_Speed_quantile_95_values': min_value33,
    'Max_Speed_quantile_95_values': max_value33,


    'Mean_RangeVertical_max_values': mean_value34,
    'Median_RangeVertical_max_values': median_value34,
    'Mode_RangeVertical_max_values': mode_value34,
```

```
        'Standard_Deviation_RangeVertical_max_values': stdev_value34,
        'Variance_RangeVertical_max_values': variance_value34,
        'Min_RangeVertical_max_values': min_value34,
        'Max_RangeVertical_max_values': max_value34,


        'Mean_RangeVertical_median_values': mean_value35,
        'Median_RangeVertical_median_values': median_value35,
        'Mode_RangeVertical_median_values': mode_value35,
        'Standard_Deviation_RangeVertical_median_values': stdev_value35,
        'Variance_RangeVertical_median_values': variance_value35,
        'Min_RangeVertical_median_values': min_value35,
        'Max_RangeVertical_median_values': max_value35,


        'Mean_RangeVertical_min_values': mean_value36,
        'Median_RangeVertical_min_values': median_value36,
        'Mode_RangeVertical_min_values': mode_value36,
        'Standard_Deviation_RangeVertical_min_values': stdev_value36,
        'Variance_RangeVertical_min_values': variance_value36,
        'Min_RangeVertical_min_values': min_value36,
        'Max_RangeVertical_min_values': max_value36,


        'Mean_RangeVertical_quantile_05_values': mean_value37,
        'Median_RangeVertical_quantile_05_values': median_value37,
        'Mode_RangeVertical_quantile_05_values': mode_value37,
        'Standard_Deviation_RangeVertical_quantile_05_values': stdev_value37,
        'Variance_RangeVertical_quantile_05_values': variance_value37,
        'Min_RangeVertical_quantile_05_values': min_value37,
        'Max_RangeVertical_quantile_05_values': max_value37,


        'Mean_RangeVertical_quantile_95_values': mean_value38,
        'Median_RangeVertical_quantile_95_values': median_value38,
        'Mode_RangeVertical_quantile_95_values': mode_value38,
        'Standard_Deviation_RangeVertical_quantile_95_values': stdev_value38,
        'Variance_RangeVertical_quantile_95_values': variance_value38,
        'Min_RangeVertical_quantile_95_values': min_value38,
        'Max_RangeVertical_quantile_95_values': max_value38
    })

    return raw_data_list
```

Η συνάρτηση calculate_raw_events_statistics υπολογίζει τα στατιστικά μέτρα για τα ακατέργαστα δεδομένα συμβάντων, τα οποία περιλαμβάνουν διάφορες μετρήσεις όπως επιταχύνσεις (direct, lateral, vertical), ταχύτητες, διάρκεια και άλλες παραμέτρους. Τα δεδομένα αυτά προέρχονται από αισθητήρες και περιγράφουν φυσικά χαρακτηριστικά των γεγονότων που συμβαίνουν κατά τη διάρκεια ενός ταξιδιού.

Τα Βασικά Βήματα της Συνάρτησης είναι:

- **Δημιουργία Λιστών Δεδομένων:** Για κάθε τύπο δεδομένων στα συμβάντα, δημιουργούνται λίστες που περιέχουν τις τιμές από τα ακατέργαστα δεδομένα, αγνοώντας τις None τιμές.
- **Υπολογισμός Στατιστικών:** Με τη χρήση της συνάρτησης calculate_statistics, υπολογίζονται βασικά στατιστικά μέτρα (Mean, Median, Mode, Standard Deviation, Variance, Min και Max) για κάθε λίστα δεδομένων.
- **Αποθήκευση Στατιστικών:** Τα υπολογισμένα στατιστικά αποθηκεύονται σε ένα λεξικό (dictionary), το οποίο περιλαμβάνει επίσης τις αιτίες εκκίνησης (startReason) και διακοπής (stopReason) του γεγονότος. Το λεξικό αυτό προστίθεται σε μια λίστα (raw_data_list) με όλα τα στατιστικά των ακατέργαστων δεδομένων

Η συνάρτηση αυτή παρέχει σημαντικές πληροφορίες για τη συμπεριφορά των δεδομένων των συμβάντων, διευκολύνοντας την ανάλυση και κατανόηση των φυσικών φαινομένων που καταγράφονται κατά τη διάρκεια των ταξιδιών.

3.3.8 Χρήση του Αρχείου "EnrichedTracks_20.11.2023.csv" για τη Συνένωση Δύο Διαφορετικών Αρχείων JSON

```
df = pd.read_csv(folder+'/EnrichedTracks_20.11.2023.csv')
df = df.loc[:, ['RawTrackToken', 'EnrichedTrackToken']]
df.sample(5)
```

	RawTrackToken	EnrichedTrackToken
438	2e09c2f5-2f87-4bae-8f6e-e431ff034fcd	2338caca-0008-4e41-b874-d14baf920dc0
910	c47ea399-f9a3-4b2b-8a3f-416858a9a374	8d7eeb14-844c-4dff-9de3-b4f04986727b
1345	19ef9b2f-0c6c-4031-a966-1d7580d2afd0	ce77dadb-a721-49bb-95b3-1e0ccfa50553
258	a9ee0d58-372d-4bf0-a757-d0864e5ecaa4	c197fcbe-0397-487f-af59-81ab5c9cc35d
1457	a8dbed51-1168-4ed8-92bc-b7cf16b4e6fe	bfaa2a73-ed05-47f2-a553-3955eea418b3

Εικ.1: Παράδειγμα αποτελεσμάτων από τυχαίο δείγμα δεδομένων που εξήχθησαν από το αρχείο EnrichedTracks_20.11.2023.csv

Το αρχείο EnrichedTracks_20.11.2023.csv περιέχει δύο βασικά αναγνωριστικά (tokens) τα οποία είναι απαραίτητα για τη συνένωση των αρχείων:

- **EnrichedTrackToken:** Το αναγνωριστικό που αντιστοιχεί στο πρώτο αρχείο, το οποίο περιέχει την καταγραφή των Ταξιδιών
- **RawTrackToken:** Το αναγνωριστικό που αντιστοιχεί στο δεύτερο αρχείο, το οποίο περιέχει λεπτομερή δεδομένα για την παρακολούθηση της οδηγικής συμπεριφοράς.

Αυτά τα δύο tokens χρησιμοποιούνται για να συνενώσουμε τα σωστά αρχεία JSON που περιέχουν τις σχετικές πληροφορίες για τα ταξίδια. Τα ονόματα των αρχείων είναι δομημένα ως εξής:

- **ProcessedTrackData_<EnrichedTrackToken>.json**
- **RawTrackData_<RawTrackToken>.json**

Για να αξιοποιήσουμε αυτές τις πληροφορίες:

- **Φόρτωτουμε το Αρχείου CSV:** Αρχικά, το αρχείο EnrichedTracks_20.11.2023.csv φορτώνεται σε ένα DataFrame χρησιμοποιώντας τη βιβλιοθήκη pandas.
- **Φιλτράρισμα των Απαραίτητων Στηλών:** Στη συνέχεια, φιλτράρονται μόνο οι στήλες που περιέχουν τα tokens, δηλαδή τις στήλες RawTrackToken και EnrichedTrackToken, ώστε να εξασφαλιστεί ότι διαθέτουμε μόνο τις απαραίτητες πληροφορίες για την επεξεργασία.

- **Δειγματοληψία για Επαλήθευση:** Επιπρόσθετα, εκτελούμε μια δειγματοληψία από το DataFrame για να επαληθεύσουμε ότι τα δεδομένα φορτώθηκαν σωστά.

3.3.9 Διαδικασία Συνένωσης Δύο Διαφορετικών Αρχείων JSON

```
folder = '/content/drive/My Drive/DrivingData/Damoov Driving Data'
driver_folder = os.path.join(folder, 'driver3')
files = os.listdir(driver_folder)

filtered_files = [file_name for file_name in files if
file_name.startswith("ProcessedTrackData")]

for file_name in filtered_files:
    file_path = os.path.join(driver_folder, file_name)
    print("file_path:", file_path)

    with open(file_path, "r", encoding="utf-8-sig") as file:
        merged_data = json.load(file)

    token = file_name.split('_')[-1].split('.')[0]
    print("file_name:", file_name)
    print("token:", token)

    rawTrackToken = df[df['EnrichedTrackToken'] ==
token]['RawTrackToken'].values[0]
    print("RawTrackToken:", rawTrackToken)

    filename = f'RawTrackData_{rawTrackToken}.json'
    path = os.path.join(driver_folder, filename)
    print("path:", path)

    with open(path, "r", encoding="utf-8-sig") as file:
        data = json.load(file)
        merged_data.update(data)
        print(merged_data.keys())

    print()
    print()
    print()
```

```
file_path: /content/drive/My Drive/DrivingData/Damoov Driving Data/driver3/ProcessedTrackData_977b8fc8-306d-4513-9350-dfef2c8f4538.json
file_name: ProcessedTrackData_977b8fc8-306d-4513-9350-dfef2c8f4538.json
token: 977b8fc8-306d-4513-9350-dfef2c8f4538
RawTrackToken: 9eaaf1e7-6ec9-4b62-ada2-6fc558336c7f
path: /content/drive/My Drive/DrivingData/Damoov Driving Data/driver3/RawTrackData_9eaaf1e7-6ec9-4b62-ada2-6fc558336c7f.json
dict_keys(['trip', 'startDate', 'endDate', 'points', 'lastKnownPoints', 'events', 'startReason', 'stopReason', 'deviceToken', 'companyId', 'trackToken'])



file_path: /content/drive/My Drive/DrivingData/Damoov Driving Data/driver3/ProcessedTrackData_eccd17d1-29c1-42a2-ba9f-be417f356c8e.json
file_name: ProcessedTrackData_eccd17d1-29c1-42a2-ba9f-be417f356c8e.json
token: eccd17d1-29c1-42a2-ba9f-be417f356c8e
RawTrackToken: dcbfdab6-c6ae-43f0-9d9e-b5d2554f5f54
path: /content/drive/My Drive/DrivingData/Damoov Driving Data/driver3/RawTrackData_dcbfdab6-c6ae-43f0-9d9e-b5d2554f5f54.json
dict_keys(['trip', 'startDate', 'endDate', 'points', 'lastKnownPoints', 'events', 'startReason', 'stopReason', 'deviceToken', 'companyId', 'trackToken'])
```

Εικ.2: Παράδειγμα αποτελεσμάτων από τη συνένωση δύο διαφορετικών αρχείων JSON

Ο παραπάνω κώδικας έχει σχεδιαστεί για να συνδυάζει δεδομένα από δύο διαφορετικά αρχεία JSON. Η διαδικασία αυτή βασίζεται στην αντιστοίχιση των EnrichedTrackToken και RawTrackToken, τα οποία συνδέονται μέσω ενός πίνακα δεδομένων που έχει φορτωθεί σε ένα DataFrame από το αρχείο EnrichedTracks_20.11.2023.csv.

Τα Βασικά Βήματα είναι:

- **Αρχικοποίηση του Φακέλου και Φιλτράρισμα των Αρχείων:**
    - **folder:** Η μεταβλητή αυτή ορίζει τη διαδρομή του κύριου φακέλου όπου αποθηκεύονται τα δεδομένα.
    - **driver_folder:** Η μεταβλητή αυτή προσδιορίζει τον υποφάκελο που περιέχει τα δεδομένα ενός συγκεκριμένου οδηγού (στη συγκεκριμένη περίπτωση, τα δεδομένα του οδηγού driver3, ο οποίος έχει τις λιγότερες διαδρομές, ώστε να ελεγχθεί αρχικά η σωστή λειτουργία του κώδικα).
    - **files:** Αποθηκεύει τη λίστα με όλα τα αρχεία που βρίσκονται στο φάκελο driver_folder.
    - **filtered_files:** Φιλτράρονται και αποθηκεύονται όλα τα αρχεία που ξεκινούν με το όνομα “ProcessedTrackData” για περαιτέρω επεξεργασία.
- **Επεξεργασία των Αρχείων ProcessedTrackData:**
    - Για κάθε αρχείο που βρίσκεται στη λίστα filtered_files, ο κώδικας ανοίγει το αρχείο JSON και φορτώνει τα δεδομένα του στο αντικείμενο merged_data.

- **token**: Το EnrichedTrackToken εξάγεται από το όνομα του αρχείου, το οποίο είναι το τμήμα του ονόματος που ακολουθεί μετά το “ProcessedTrackData_” και πριν από την κατάληξη “.json”.
- Χρησιμοποιείται το token για να εντοπιστεί το αντίστοιχο RawTrackToken από το DataFrame df.

- **Εντοπισμός και Επεξεργασία του Αρχείου RawTrackData:**
  - Με βάση το RawTrackToken, το οποίο ανακτήθηκε από το προηγούμενο βήμα, εντοπίζεται το αντίστοιχο αρχείο RawTrackData στον φάκελο driver_folder.
  - Το αρχείο RawTrackData ανοίγεται και τα δεδομένα του συνδυάζονται (merge) με τα δεδομένα του αρχείου ProcessedTrackData στο αντικείμενο merged_data.
- **Εμφάνιση Πληροφοριών:**
  - Ο κώδικας εκτυπώνει διάφορες χρήσιμες πληροφορίες για την επιβεβαίωση της σωστής εκτέλεσης της διαδικασίας:
    - **file_path**: Η πλήρης διαδρομή του κάθε αρχείου ProcessedTrackData.
    - **token**: Το EnrichedTrackToken που εξάγεται από το όνομα του αρχείου.
    - **RawTrackToken**: Το αντίστοιχο RawTrackToken που εντοπίστηκε από το DataFrame df.
    - **keys**: Τα κλειδιά του συνδυασμένου λεξικού merged_data, τα οποία δείχνουν τα διαφορετικά είδη δεδομένων που περιέχονται στο συνδυασμένο σύνολο δεδομένων.

3.3.10 Συνάρτηση create_dataframe

```
def create_dataframe(folder_path):
    files = os.listdir(folder_path)
    filtered_files = [file_name for file_name in files if
file_name.startswith("ProcessedTrackData")]

    trip_scores_list = []
```

```
    date_updated_list = []
    trip_duration_list = []
    transport_type_list = []
    statistics_list = []
    trip_events_statistics_list = []
    raw_data_list = []
    raw_events_weather_statistics_list = []
    raw_type_counts_list = []
    raw_accident_trigger_counts_list = []
    raw_lastKnownPoints_list = []

    for file_name in filtered_files:
        file_path = os.path.join(driver_folder, file_name)

        with open(file_path, "r", encoding="utf-8-sig") as file:
            merged_data = json.load(file)

        token = file_name.split('_')[-1].split('.')[0]
        rawTrackToken = df[df['EnrichedTrackToken'] ==
token]['RawTrackToken'].values[0]
        filename = f'RawTrackData_{rawTrackToken}.json'
        path = os.path.join(driver_folder, filename)

        with open(path, "r", encoding="utf-8-sig") as file:
            data = json.load(file)
            merged_data.update(data)

            trip_scores_list.append(merged_data['trip']['scores'])
            date_updated_list.append(merged_data['trip']['dateUpdated'])

trip_duration_list.append(merged_data['trip']['data']['endDateUnixMilliseconds'
] - merged_data['trip']['data']['startDateUnixMilliseconds'])

transport_type_list.append(merged_data['trip']['data']['transportType'])
            statistics_list.append(merged_data['trip']['statistics'])

            trip_events = merged_data['trip']['events']

            raw_startReason = merged_data['startReason']
            raw_stopReason = merged_data['stopReason']

            if merged_data['events'] is None:
                raw_events = []
            else:
                raw_events = merged_data['events']


            if merged_data['lastKnownPoints'] is None:
                raw_lastKnownPoints = []
            else:
```

```
                raw_lastKnownPoints = merged_data['lastKnownPoints']


            trip_events_value = []
            trip_events_type = []
            trip_events_confirmNeeded = []
            trip_events_asI15 = []
            trip_events_weather = []

            for trip_event in trip_events:
                value = trip_event['value']
                trip_events_value.append(value)

                event_type = trip_event['type']
                trip_events_type.append(event_type)

                event_confirmNeeded = trip_event['confirmNeeded']
                trip_events_confirmNeeded.append(event_confirmNeeded)

                event_asI15 = trip_event['asI15']
                trip_events_asI15.append(event_asI15)

                latitude = trip_event['lat']
                longitude = trip_event['long']
                timestamp = date_to_timestamp(trip_event['date'])

                weather_data =
get_weather_by_coordinates_and_timestamp(latitude, longitude, timestamp)
                trip_events_weather.append(weather_data)


            raw_lastKnownPoints_accuracy = []
            for lastKnownPoint in raw_lastKnownPoints:
                accuracy = lastKnownPoint['accuracy']
                raw_lastKnownPoints_accuracy.append(accuracy)


            mean_value, median_value, mode_value, stdev_value, variance_value,
min_value, max_value = calculate_statistics(raw_lastKnownPoints_accuracy)
            raw_lastKnownPoints_list.append({
                'Mean_raw_lastKnownPoints_accuracy_values': mean_value,
                'Median_raw_lastKnownPoints_accuracy_values': median_value,
                'Mode_raw_lastKnownPoints_accuracy_values': mode_value,
                'Standard_Deviation_raw_lastKnownPoints_accuracy_values':
stdev_value,
                'Variance_raw_lastKnownPoints_accuracy_values': variance_value,
                'Min_raw_lastKnownPoints_accuracy_values': min_value,
                'Max_raw_lastKnownPoints_accuracy_values': max_value
            })
```

```
            raw_events_weather = []
            raw_events_type = []
            accident_trigger_counts = {}
            for raw_event in raw_events:
                event_type = raw_event['Type']
                raw_events_type.append(event_type)

                accident_trigger = raw_event['AccidentTrigger']
                if accident_trigger != 'NOT_DEFINED':
                    if accident_trigger in accident_trigger_counts:
                        accident_trigger_counts[accident_trigger] += 1
                    else:
                        accident_trigger_counts[accident_trigger] = 1


                latitude = raw_event['PointStart']['Latitude']
                longitude = raw_event['PointStart']['Longitude']
                timestamp = raw_event['TimeStart']

                weather_data =
get_weather_by_coordinates_and_timestamp(latitude, longitude, timestamp)
                raw_events_weather.append(weather_data)


            raw_accident_trigger_counts_list.append(accident_trigger_counts)


            raw_type_counts = {}
            for events_type in raw_events_type:
                if events_type in raw_type_counts:
                    raw_type_counts[events_type] += 1
                else:
                    raw_type_counts[events_type] = 1

            raw_type_counts_list.append(raw_type_counts)

            type_counts = {}
            for events_type in trip_events_type:
                if events_type in type_counts:
                    type_counts[events_type] += 1
                else:
                    type_counts[events_type] = 1


            confirmNeeded_counts = {}
            for events_confirmNeeded in trip_events_confirmNeeded:
                if events_confirmNeeded is not None:
                    if events_confirmNeeded in confirmNeeded_counts:
                        confirmNeeded_counts[events_confirmNeeded] += 1
```

```
                    else:
                        confirmNeeded_counts[events_confirmNeeded] = 1


            asI15_counts = {}
            for events_asI15 in trip_events_asI15:
                if events_asI15 is not None:
                    if events_asI15 in asI15_counts:
                        asI15_counts[events_asI15] += 1
                    else:
                        asI15_counts[events_asI15] = 1


            mean_value, median_value, mode_value, stdev_value, variance_value,
min_value, max_value = calculate_statistics(trip_events_value)

            trip_events_statistics_list.append({
                'trip_events_type': type_counts,
                'trip_events_confirmNeeded': confirmNeeded_counts,
                'trip_events_asI15': asI15_counts,
                'Mean_trip_events_values': mean_value,
                'Median_trip_events_values': median_value,
                'Mode_trip_events_values': mode_value,
                'Standard_Deviation_trip_events_values': stdev_value,
                'Variance_trip_events_values': variance_value,
                'Min_trip_events_values': min_value,
                'Max_trip_events_values': max_value
            })

            raw_data_list = calculate_raw_events_statistics(raw_events,
raw_data_list, raw_startReason, raw_stopReason)

            raw_events_weather_statistics_list =
calculate_weather_statistics(raw_events_weather,
raw_events_weather_statistics_list, name_print="raw_events")


    df1 = pd.DataFrame(trip_scores_list)
    df2 = pd.DataFrame({'Date_Updated': date_updated_list, 'Trip_Duration':
trip_duration_list})
    df3 = pd.DataFrame(transport_type_list)
    df4 = pd.DataFrame(statistics_list)
    df5 = pd.DataFrame(trip_events_statistics_list)
    df6 = pd.DataFrame(raw_data_list)
    df7 = pd.DataFrame(raw_events_weather_statistics_list)
    df8 = pd.DataFrame({'Raw_type_counts': raw_type_counts_list,
'Raw_accident_trigger_counts': raw_accident_trigger_counts_list})
    df9 = pd.DataFrame(raw_lastKnownPoints_list)
    merged_df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9],
axis=1)
```

```
    return merged_df
```

Η συνάρτηση create_dataframe δημιουργεί ένα ενοποιημένο DataFrame που συνδυάζει δομένα από αρχεία JSON για κάθε οδηγό, επιτρέποντας την ανάλυση πληροφοριών ταξιδιού, συμβάντων και καιρικών συνθηκών.

Τα Βασικά Βήματα της Συνάρτησης είναι:

- **Αρχικοποίηση και Φιλτράρισμα Αρχείων:**
    - **folder_path**: Η διαδρομή του φακέλου που περιέχει τα δεδομένα.
    - Φιλτράρονται τα αρχεία στο φάκελο για να εντοπιστούν εκείνα που ξεκινούν με “ProcessedTrackData”, τα οποία αποθηκεύονται σε μια λίστα (filtered_files).
- **Συνδυασμός Δεδομένων από Εμπλουτισμένα και Ακατέργαστα Αρχεία JSON**:
    - Για κάθε αρχείο στη λίστα filtered_files, ανοίγεται και φορτώνεται το περιεχόμενό του σε ένα λεξικό merged_data.
    - Από το όνομα του αρχείου εξάγεται το EnrichedTrackToken, το οποίο χρησιμοποιείται για να εντοπιστεί το αντίστοιχο RawTrackToken από το DataFrame df.
    - Στη συνέχεια, ανοίγεται το αντίστοιχο αρχείο RawTrackData, και τα δεδομένα του συνδυάζονται με τα δεδομένα του ProcessedTrackData στο merged_data.
- **Εξαγωγή και Υπολογισμός Στατιστικών**:
    - Από τα συνδυασμένα δεδομένα εξάγονται διάφορες πληροφορίες, όπως:
        - **Βαθμολογίες Ταξιδιού** (**trip_scores_list**)
        - **Ημερομηνία Ενημέρωσης** (**date_updated_list**)
        - **Διάρκεια Ταξιδιού** (**trip_duration_list**)
        - **Τύπος Μεταφοράς** (**transport_type_list**)
        - **Στατιστικά Δεδομένα** (**statistics_list**)
        - **Στατιστικά Συμβάντων** (**trip_events_statistics_list**)

- **Στατιστικά Ακατέργαστων Δεδομένων** (**raw_data_list**)
- **Στατιστικά Καιρικών Συνθηκών για Ακατέργαστα Συμβάντα** (**raw_events_weather_statistics_list**)
- **Καταμέτρηση Τύπων Ακατέργαστων Συμβάντων** (**raw_type_counts_list**)
- **Καταμέτρηση Ατυχηματικών Αιτίων** (**raw_accident_trigger_counts_list**)
- **Στατιστικά για τα Τελευταία Γνωστά Σημεία** (**raw_lastKnownPoints_list**)

- **Υπολογισμός Στατιστικών για Συμβάντα και Καιρικές Συνθήκες**:
  - Υπολογίζονται στατιστικά μέτρα με τη χρήση της συνάρτησης calculate_statistics.
- **Δημιουργία και Συνένωση DataFrames**:
  - Οι πληροφορίες που έχουν συλλεχθεί και επεξεργαστεί οργανώνονται σε ξεχωριστά DataFrames.
  - Τα DataFrames συνενώνονται σε ένα ενιαίο DataFrame (merged_df) χρησιμοποιώντας τη συνάρτηση pd.concat, η οποία συνδυάζει τα δεδομένα σε μια ενιαία μορφή με όλες τις απαραίτητες πληροφορίες.
- **Επιστροφή του Ενοποιημένου DataFrame:**
  - Η συνάρτηση ολοκληρώνεται με την επιστροφή του merged_df, το οποίο περιέχει όλα τα δεδομένα από τα αρχεία JSON για τον συγκεκριμένο οδηγό και είναι έτοιμο για περαιτέρω ανάλυση και μελέτη.

Η συνάρτηση create_dataframe είναι ουσιώδης για την αυτοματοποίηση της διαδικασίας συλλογής, επεξεργασίας και οργάνωσης δεδομένων από πολλαπλά αρχεία JSON. Με την ακριβή αντιστοίχιση των ProcessedTrackToken και RawTrackToken, διασφαλίζεται ότι τα δεδομένα ταξιδιού συνδυάζονται σωστά, επιτρέποντας μια ολοκληρωμένη ανάλυση των συμβάντων και των περιβαλλοντικών συνθηκών κατά τη διάρκεια των ταξιδιών.

3.3.11 Εφαρμογή της Συνάρτησης create_dataframe για Κάθε Οδηγό

```
folder = '/content/drive/My Drive/DrivingData/Damoov Driving Data'
driver_folder = os.path.join(folder, 'driver1')
df_driver1 = create_dataframe(driver_folder)
df_driver1.to_csv('df_driver1.csv', index=None)

driver_folder = os.path.join(folder, 'driver2')
df_driver2 = create_dataframe(driver_folder)
df_driver2.to_csv('df_driver2.csv', index=None)

driver_folder = os.path.join(folder, 'driver3')
df_driver3 = create_dataframe(driver_folder)
df_driver3.to_csv('df_driver3.csv', index=None)

driver_folder = os.path.join(folder, 'driver4')
df_driver4 = create_dataframe(driver_folder)
df_driver4.to_csv('df_driver4.csv', index=None)

driver_folder = os.path.join(folder, 'driver5')
df_driver5 = create_dataframe(driver_folder)
df_driver5.to_csv('df_driver5.csv', index=None)

driver_folder = os.path.join(folder, 'driver6')
df_driver6 = create_dataframe(driver_folder)
df_driver6.to_csv('df_driver6.csv', index=None)

driver_folder = os.path.join(folder, 'driver7')
df_driver7 = create_dataframe(driver_folder)
df_driver7.to_csv('df_driver7.csv', index=None)
```

Στη συνέχεια, γίνεται χρήση της συνάρτησης create_dataframe για τη δημιουργία ενός ενοποιημένου DataFrame για κάθε οδηγό. Κάθε DataFrame αποθηκεύεται σε ένα ξεχωριστό αρχείο CSV για μελλοντική χρήση.

Διαδικασία για Κάθε Οδηγό:

- **Ορισμός Διαδρομής Φακέλου για τον Οδηγό**: Η διαδρομή του φακέλου που περιέχει τα δεδομένα καθορίζεται από τη μεταβλητή driver_folder, η οποία συνδυάζει την κύρια διαδρομή (folder) με το όνομα του υποφακέλου που αντιστοιχεί στον οδηγό.
- **Δημιουργία του DataFrame**: Η συνάρτηση create_dataframe καλείται για κάθε οδηγό, με την αντίστοιχη διαδρομή φακέλου να περνά ως παράμετρος. Το

αποτέλεσμα είναι ένα DataFrame που περιέχει τα συνδυασμένα δεδομένα από τα εμπλουτισμένα και ακατέργαστα αρχεία JSON για τον συγκεκριμένο οδηγό.

- **Αποθήκευση του DataFrame σε Αρχείο CSV**: Το DataFrame κάθε οδηγού αποθηκεύεται σε ένα ξεχωριστό αρχείο CSV με το όνομα df_driverX.csv, όπου X αντιστοιχεί στον αριθμό του οδηγού (π.χ., df_driver1.csv, df_driver2.csv, κ.λπ.). Η παράμετρος index=None χρησιμοποιείται για να αποτραπεί η αποθήκευση των δεικτών (index) του DataFrame στο αρχείο CSV.

3.3.12 Συνένωση των DataFrames για Όλους τους Οδηγούς

```
folder = '/content/drive/My Drive/DrivingData'
df_driver1 = pd.read_csv(folder+'/df_driver1.csv')
df_driver2 = pd.read_csv(folder+'/df_driver2.csv')
df_driver3 = pd.read_csv(folder+'/df_driver3.csv')
df_driver4 = pd.read_csv(folder+'/df_driver4.csv')
df_driver5 = pd.read_csv(folder+'/df_driver5.csv')
df_driver6 = pd.read_csv(folder+'/df_driver6.csv')
df_driver7 = pd.read_csv(folder+'/df_driver7.csv')
df = pd.concat([df_driver1, df_driver2, df_driver3, df_driver4, df_driver5,
df_driver6, df_driver7], axis=0)
df.reset_index(drop=True, inplace=True)
```

Μετά τη δημιουργία και αποθήκευση των επιμέρους DataFrames για κάθε οδηγό, το επόμενο βήμα είναι να συνενωθούν όλα αυτά τα DataFrames σε ένα ενιαίο DataFrame που περιέχει τα δεδομένα όλων των οδηγών.

Δομή του Κώδικα:

- **Φόρτωση των Αρχείων CSV για Κάθε Οδηγό:** Φορτώνονται τα αρχεία CSV που δημιουργήθηκαν προηγουμένως για κάθε οδηγό σε ξεχωριστά DataFrames
- **Συνένωση των DataFrames:** Τα DataFrames για όλους τους οδηγούς συνενώνονται σε ένα ενιαίο DataFrame (df) χρησιμοποιώντας τη συνάρτηση pd.concat(). Η συνένωση γίνεται κατά μήκος του άξονα των γραμμών (axis=0), προσθέτοντας τα δεδομένα του ενός οδηγού κάτω από τα δεδομένα του άλλου.

- **Αναδιάταξη των Δεικτών:** Μετά τη συνένωση των DataFrames, οι δείκτες (index) στο νέο DataFrame df επαναρυθμίζονται χρησιμοποιώντας τη μέθοδο reset_index(), με το επιχείρημα drop=True για να αποτραπεί η εισαγωγή των παλιών δεικτών ως νέα στήλη στο DataFrame.

3.3.13 Προσδιορισμός της Ώρας της Ημέρας και Κατηγοριοποίηση Λεπτών

```
def time_of_day(df):
    morning_start = 6
    morning_end = 12
    afternoon_start = 12
    afternoon_end = 18
    evening_start = 18

    df['Morning'] = ((df['Hour'] >= morning_start) & (df['Hour'] <
morning_end)).astype(int)
    df['Afternoon'] = ((df['Hour'] >= afternoon_start) & (df['Hour'] <
afternoon_end)).astype(int)
    df['Evening'] = (((df['Hour'] >= evening_start) | (df['Hour'] <
morning_start))).astype(int)

    return df


def categorize_minutes(df):
    minute_intervals = [(0, 14), (15, 29), (30, 44), (45, 59)]

    for i, (start, end) in enumerate(minute_intervals):
        interval_name = f'Minute_{start}-{end}'
        df[interval_name] = ((df['Minute'] >= start) & (df['Minute'] <=
end)).astype(int)

    return df
```

Ο κώδικας επεκτείνει την ανάλυση του DataFrame df με δύο βασικές λειτουργίες:

- τον προσδιορισμό της ώρας της ημέρας
- την κατηγοριοποίηση των λεπτών της ώρας.

Αυτές οι λειτουργίες προσθέτουν πρόσθετες δυνατότητες στο DataFrame, επιτρέποντας πιο λεπτομερή ανάλυση της χρονικής κατανομής των δεδομένων οδήγησης.

Συναρτήσεις:

- **Συνάρτηση time_of_day: Προσδιορισμός της Ώρας της Ημέρας**
    - Η συνάρτηση time_of_day κατηγοριοποιεί τις εγγραφές του DataFrame σε τρεις χρονικές ζώνες της ημέρας:
        - **Morning**: Από τις 06:00 έως τις 11:59.
        - **Afternoon**: Από τις 12:00 έως τις 17:59.
        - **Evening**: Από τις 18:00 έως τις 05:59.
    - Η συνάρτηση δημιουργεί τρεις νέες στήλες στο DataFrame (Morning, Afternoon, Evening), κάθε μία από τις οποίες περιέχει δυαδικές τιμές (0 ή 1) που δείχνουν αν η συγκεκριμένη εγγραφή ανήκει στην αντίστοιχη χρονική ζώνη.
- **Συνάρτηση categorize_minutes: Κατηγοριοποίηση Λεπτών της Ώρας**
    - Η συνάρτηση categorize_minutes κατηγοριοποιεί τις εγγραφές του DataFrame σε τέσσερα χρονικά διαστήματα εντός της ώρας:
        - 0-14 Λεπτά
        - 15-29 Λεπτά
        - 30-44 Λεπτά
        - 45-59 Λεπτά
    - Προσθέτει τέσσερις νέες στήλες στο DataFrame (Minute_0-14, Minute_15-29, Minute_30-44, Minute_45-59), κάθε μία από τις οποίες περιέχει δυαδικές τιμές (0 ή 1) που δείχνουν αν η εγγραφή ανήκει στο αντίστοιχο χρονικό διάστημα.

3.3.14 Συνάρτηση preprocess_dataframe

```
def preprocess_dataframe(df):
    df['Date_Updated'] = pd.to_datetime(df['Date_Updated'])
    df['Season'] = df['Date_Updated'].dt.month.apply(lambda x: 'Spring' if 3 <=
```

```
x <= 5 else ('Summer' if 6 <= x <= 8 else ('Autumn' if 9 <= x <= 11 else
'Winter')))
    df['Day_of_Week'] = df['Date_Updated'].dt.dayofweek
    df['Weekend'] = df['Date_Updated'].dt.dayofweek.isin([5, 6]).astype(int)
    df['Hour'] = df['Date_Updated'].dt.hour
    df['Minute'] = df['Date_Updated'].dt.minute

    df = time_of_day(df)
    df = categorize_minutes(df)

    return df

df = preprocess_dataframe(df)
```

Η συνάρτηση preprocess_dataframe αναλαμβάνει την προεπεξεργασία του DataFrame df, προσθέτοντας νέες στήλες που παρέχουν πρόσθετες πληροφορίες βασισμένες στην ημερομηνία και την ώρα. Αυτή η διαδικασία εμπλουτίζει το αρχικό DataFrame με χρήσιμες πληροφορίες για την ανάλυση των ταξιδιών.

Αναλυτική Επεξήγηση:

- **Μετατροπή της Στήλης Date_Updated σε Αντικείμενο datetime:**
    - Η στήλη Date_Updated, η οποία περιέχει την ημερομηνία και ώρα ενημέρωσης των δεδομένων, μετατρέπεται σε αντικείμενο datetime χρησιμοποιώντας τη μέθοδο pd.to_datetime(). Αυτό επιτρέπει την εύκολη εξαγωγή χρονικών πληροφοριών, όπως μήνας, ημέρα της εβδομάδας, ώρα και λεπτά.
- **Προσδιορισμός της Εποχής (Season):**
    - Βασισμένη στον μήνα που προκύπτει από την ημερομηνία, η συνάρτηση καθορίζει την εποχή:
        - **Άνοιξη (Spring):** Μάρτιος έως Μάιος.
        - **Καλοκαίρι (Summer):** Ιούνιος έως Αύγουστος.
        - **Φθινόπωρο (Autumn):** Σεπτέμβριος έως Νοέμβριος.
        - **Χειμώνας (Winter):** Δεκέμβριος έως Φεβρουάριος.

- Αυτό γίνεται με την εφαρμογή της apply() σε συνδυασμό με μια συνάρτηση lambda.

- **Προσδιορισμός της Ημέρας της Εβδομάδας (Day_of_Week)**:
  - Η στήλη Day_of_Week εξάγει την ημέρα της εβδομάδας από το αντικείμενο datetime, με τιμές που κυμαίνονται από 0 (Δευτέρα) έως 6 (Κυριακή).

- **Εντοπισμός Σαββατοκύριακου (Weekend):**
  - Η στήλη Weekend προσδιορίζει αν η ημέρα είναι Σάββατο (5) ή Κυριακή (6). Αυτή η στήλη παίρνει τιμές 1 για το Σαββατοκύριακο και 0 για τις υπόλοιπες ημέρες, χρησιμοποιώντας τη μέθοδο isin() και μετατροπή σε ακέραιο (astype(int)).

- **Εξαγωγή της Ώρας (Hour) και των Λεπτών (Minute):**
  - Οι στήλες Hour και Minute εξάγονται απευθείας από την ημερομηνία Date_Updated.

- **Προσδιορισμός της Ώρας της Ημέρας (Morning, Afternoon, Evening):**
  - Η συνάρτηση time_of_day(df) καλείται για να προσθέσει τις στήλες Morning, Afternoon και Evening στο DataFrame, κατηγοριοποιώντας τις εγγραφές ανάλογα με την ώρα της ημέρας.

- **Κατηγοριοποίηση των Λεπτών της Ώρας:**
  - Η συνάρτηση categorize_minutes(df) προσθέτει στήλες που κατηγοριοποιούν τα λεπτά της ώρας σε συγκεκριμένα διαστήματα.

- **Επιστροφή του Επεξεργασμένου DataFrame:**
  - Το τελικό επεξεργασμένο DataFrame επιστρέφεται, περιλαμβάνοντας όλες τις νέες στήλες που δημιουργήθηκαν κατά τη διάρκεια της προεπεξεργασίας.

3.3.15 Ανάλυση και Κατάργηση Στηλών με Μοναδικές Τιμές

```
list_cols = [
    'ecoTyres', 'accidentsCount', 'Max_Visibility_raw_events_weather',
    'Min_Visibility_raw_events_weather',
'Variance_Visibility_raw_events_weather',
    'Standard_Deviation_Visibility_raw_events_weather',
'Mode_Visibility_raw_events_weather',
    'Median_Visibility_raw_events_weather',
'Mean_Visibility_raw_events_weather', 'Season'
]

for column in list_cols:
    print(f"Κατανομή τιμών στο column με όνομα '{column}':")
    print(df[column].value_counts())
    print()
```

Ο παραπάνω κώδικας πραγματοποιεί ανάλυση της κατανομής των τιμών για μια επιλεγμένη λίστα στηλών (list_cols) στο DataFrame df. Αυτή η διαδικασία αποσκοπεί στην εντόπιση στηλών που περιέχουν μοναδικές ή επαναλαμβανόμενες τιμές, οι οποίες μπορεί να μην προσφέρουν ουσιαστική πληροφορία για την ανάλυση και μπορούν να αφαιρεθούν.

Αιτιολόγηση της Κατάργησης Στηλών:

- **Μείωση Πλεονασμού:** Στήλες που περιέχουν μοναδικές τιμές δεν προσφέρουν επιπλέον πληροφορίες. Η κατάργησή τους βελτιώνει τη συνολική ποιότητα των δεδομένων.
- **Βελτίωση Απόδοσης**: Η κατάργηση αυτών των στηλών μειώνει το μέγεθος του DataFrame, επιταχύνοντας τις επόμενες διαδικασίες ανάλυσης και μοντελοποίησης.
- **Καθαρισμός Δεδομένων**: Η απομάκρυνση των άχρηστων στηλών βοηθά στη διατήρηση ενός καθαρού και αποτελεσματικού συνόλου δεδομένων, εστιασμένου στα γνωρίσματα που είναι πραγματικά σημαντικά για την ανάλυση.

Με αυτήν την προσέγγιση, το DataFrame df θα περιέχει μόνο στήλες που προσφέρουν ουσιαστική πληροφορία για την ανάλυση, βελτιώνοντας έτσι τη συνολική ποιότητα των δεδομένων.

3.3.16 Διαχωρισμός των Γνωρισμάτων του Συνόλου Δεδομένων

```
numeric_cols = [
    'acceleration', 'braking', 'cornering', 'speeding', 'phoneUsage', 'eco',
'ecoBrakes', 'ecoDepreciation', 'ecoFuel', 'Trip_Duration', 'mileage',
'durationMinutes', 'accelerationsCount', 'brakingsCount', 'corneringsCount',
'totalSpeedingMileage', 'midSpeedingMileage', 'highSpeedingMileage',
'phoneUsageDurationMinutes', 'phoneUsageMileage',
'phoneUsageWithSpeedingDurationMinutes', 'phoneUsageWithSpeedingMileage',
'dayHours', 'rushHours', 'nightHours', 'averageSpeed', 'maxSpeed',
'Mean_trip_events_values', 'Median_trip_events_values',
'Mode_trip_events_values', 'Standard_Deviation_trip_events_values',
'Variance_trip_events_values', 'Min_trip_events_values',
'Max_trip_events_values', 'Mean_acceleration_direct_values',
'Median_acceleration_direct_values', 'Mode_acceleration_direct_values',
'Standard_Deviation_acceleration_direct_values',
'Variance_acceleration_direct_values', 'Min_acceleration_direct_values',
'Max_acceleration_direct_values', 'Mean_acceleration_direct_end_values',
'Median_acceleration_direct_end_values',
'Mode_acceleration_direct_end_values',
'Standard_Deviation_acceleration_direct_end_values',
'Variance_acceleration_direct_end_values', Min_acceleration_direct_end_values',
'Max_acceleration_direct_end_values', 'Mean_acceleration_lateral_values',
'Median_acceleration_lateral_values', 'Mode_acceleration_lateral_values',
'Standard_Deviation_acceleration_lateral_values',
'Variance_acceleration_lateral_values', 'Min_acceleration_lateral_values',
'Max_acceleration_lateral_values', 'Mean_acceleration_lateral_end_values',
'Median_acceleration_lateral_end_values',
'Mode_acceleration_lateral_end_values',
'Standard_Deviation_acceleration_lateral_end_values',
'Variance_acceleration_lateral_end_values',
'Min_acceleration_lateral_end_values', 'Max_acceleration_lateral_end_values',
'Mean_acceleration_vertical_values', 'Median_acceleration_vertical_values',
'Mode_acceleration_vertical_values',
'Standard_Deviation_acceleration_vertical_values',
'Variance_acceleration_vertical_values', 'Min_acceleration_vertical_values',
'Max_acceleration_vertical_values', 'Mean_acceleration_vertical_end_values',
'Median_acceleration_vertical_end_values',
'Mode_acceleration_vertical_end_values',
'Standard_Deviation_acceleration_vertical_end_values',
'Variance_acceleration_vertical_end_values',
'Min_acceleration_vertical_end_values', 'Max_acceleration_vertical_end_values',
'Mean_Duration_values', 'Median_Duration_values', 'Mode_Duration_values',
'Standard_Deviation_Duration_values', 'Variance_Duration_values',
'Min_Duration_values', 'Max_Duration_values', 'Mean_PrevEventSpeed_values',
'Median_PrevEventSpeed_values', 'Mode_PrevEventSpeed_values',
'Standard_Deviation_PrevEventSpeed_values', 'Variance_PrevEventSpeed_values',
'Min_PrevEventSpeed_values', 'Max_PrevEventSpeed_values',
'Mean_PureDuration_values', 'Median_PureDuration_values',
'Mode_PureDuration_values', 'Standard_Deviation_PureDuration_values',
```

```
'Variance_PureDuration_values', 'Min_PureDuration_values',
'Max_PureDuration_values', 'Mean_Reliability_values',
'Median_Reliability_values', 'Mode_Reliability_values',
'Standard_Deviation_Reliability_values', 'Variance_Reliability_values',
'Min_Reliability_values', 'Max_Reliability_values', 'Mean_SpeedMedian_values',
'Median_SpeedMedian_values', 'Mode_SpeedMedian_values',
'Standard_Deviation_SpeedMedian_values', 'Variance_SpeedMedian_values',
'Min_SpeedMedian_values', 'Max_SpeedMedian_values', 'Mean_SpeedStart_values',
'Median_SpeedStart_values', 'Mode_SpeedStart_values',
'Standard_Deviation_SpeedStart_values', 'Variance_SpeedStart_values',
'Min_SpeedStart_values', 'Max_SpeedStart_values', 'Mean_SpeedStop_values',
'Median_SpeedStop_values', 'Mode_SpeedStop_values',
'Standard_Deviation_SpeedStop_values', 'Variance_SpeedStop_values',
'Min_SpeedStop_values', 'Max_SpeedStop_values', 'Mean_Accuracy_max_values',
'Median_Accuracy_max_values', 'Mode_Accuracy_max_values',
'Standard_Deviation_Accuracy_max_values', 'Variance_Accuracy_max_values',
'Min_Accuracy_max_values', 'Max_Accuracy_max_values',
'Mean_Accuracy_median_values', 'Median_Accuracy_median_values',
'Mode_Accuracy_median_values', 'Standard_Deviation_Accuracy_median_values',
'Variance_Accuracy_median_values', 'Min_Accuracy_median_values',
'Max_Accuracy_median_values', 'Mean_Accuracy_min_values',
'Median_Accuracy_min_values', 'Mode_Accuracy_min_values',
'Standard_Deviation_Accuracy_min_values', 'Variance_Accuracy_min_values',
'Min_Accuracy_min_values', 'Max_Accuracy_min_values',
'Mean_Accuracy_quantile_05_values', 'Median_Accuracy_quantile_05_values',
'Mode_Accuracy_quantile_05_values',
'Standard_Deviation_Accuracy_quantile_05_values',
'Variance_Accuracy_quantile_05_values', 'Min_Accuracy_quantile_05_values',
'Max_Accuracy_quantile_05_values', 'Mean_Accuracy_quantile_95_values',
'Median_Accuracy_quantile_95_values', 'Mode_Accuracy_quantile_95_values',
'Standard_Deviation_Accuracy_quantile_95_values',
'Variance_Accuracy_quantile_95_values', 'Min_Accuracy_quantile_95_values',
'Max_Accuracy_quantile_95_values', 'Mean_RangeDirect_max_values',
'Median_RangeDirect_max_values', 'Mode_RangeDirect_max_values',
'Standard_Deviation_RangeDirect_max_values', 'Variance_RangeDirect_max_values',
'Min_RangeDirect_max_values', 'Max_RangeDirect_max_values',
'Mean_RangeDirect_median_values', 'Median_RangeDirect_median_values',
'Mode_RangeDirect_median_values',
'Standard_Deviation_RangeDirect_median_values',
'Variance_RangeDirect_median_values', 'Min_RangeDirect_median_values',
'Max_RangeDirect_median_values', 'Mean_RangeDirect_min_values',
'Median_RangeDirect_min_values', 'Mode_RangeDirect_min_values',
'Standard_Deviation_RangeDirect_min_values', 'Variance_RangeDirect_min_values',
'Min_RangeDirect_min_values', 'Max_RangeDirect_min_values',
'Mean_RangeDirect_quantile_05_values', 'Median_RangeDirect_quantile_05_values',
'Mode_RangeDirect_quantile_05_values',
'Standard_Deviation_RangeDirect_quantile_05_values',
'Variance_RangeDirect_quantile_05_values',
'Min_RangeDirect_quantile_05_values', 'Max_RangeDirect_quantile_05_values',
'Mean_RangeDirect_quantile_95_values',
```

```
'Median_RangeDirect_quantile_95_values', 'Mode_RangeDirect_quantile_95_values',
'Standard_Deviation_RangeDirect_quantile_95_values',
'Variance_RangeDirect_quantile_95_values',
'Min_RangeDirect_quantile_95_values', 'Max_RangeDirect_quantile_95_values',
'Mean_RangeLateral_max_values', 'Median_RangeLateral_max_values',
'Mode_RangeLateral_max_values', 'Standard_Deviation_RangeLateral_max_values',
'Variance_RangeLateral_max_values', 'Min_RangeLateral_max_values',
'Max_RangeLateral_max_values', 'Mean_RangeLateral_median_values',
'Median_RangeLateral_median_values', 'Mode_RangeLateral_median_values',
'Standard_Deviation_RangeLateral_median_values',
'Variance_RangeLateral_median_values', 'Min_RangeLateral_median_values',
'Max_RangeLateral_median_values', 'Mean_RangeLateral_min_values',
'Median_RangeLateral_min_values', 'Mode_RangeLateral_min_values',
'Standard_Deviation_RangeLateral_min_values',
'Variance_RangeLateral_min_values', 'Min_RangeLateral_min_values',
'Max_RangeLateral_min_values', 'Mean_RangeLateral_quantile_05_values',
'Median_RangeLateral_quantile_05_values',
'Mode_RangeLateral_quantile_05_values',
'Standard_Deviation_RangeLateral_quantile_05_values',
'Variance_RangeLateral_quantile_05_values',
'Min_RangeLateral_quantile_05_values', 'Max_RangeLateral_quantile_05_values',
'Mean_RangeLateral_quantile_95_values',
'Median_RangeLateral_quantile_95_values',
'Mode_RangeLateral_quantile_95_values',
'Standard_Deviation_RangeLateral_quantile_95_values',
'Variance_RangeLateral_quantile_95_values',
'Min_RangeLateral_quantile_95_values', 'Max_RangeLateral_quantile_95_values',
'Mean_Speed_max_values', 'Median_Speed_max_values', 'Mode_Speed_max_values',
'Standard_Deviation_Speed_max_values', 'Variance_Speed_max_values',
'Min_Speed_max_values', 'Max_Speed_max_values', 'Mean_Speed_median_values',
'Median_Speed_median_values', 'Mode_Speed_median_values',
'Standard_Deviation_Speed_median_values', 'Variance_Speed_median_values',
'Min_Speed_median_values', 'Max_Speed_median_values', 'Mean_Speed_min_values',
'Median_Speed_min_values', 'Mode_Speed_min_values',
'Standard_Deviation_Speed_min_values', 'Variance_Speed_min_values',
'Min_Speed_min_values', 'Max_Speed_min_values',
'Mean_Speed_quantile_05_values', 'Median_Speed_quantile_05_values',
'Mode_Speed_quantile_05_values', 'Standard_Deviation_Speed_quantile_05_values',
'Variance_Speed_quantile_05_values', 'Min_Speed_quantile_05_values',
'Max_Speed_quantile_05_values', 'Mean_Speed_quantile_95_values',
'Median_Speed_quantile_95_values', 'Mode_Speed_quantile_95_values',
'Standard_Deviation_Speed_quantile_95_values',
'Variance_Speed_quantile_95_values', 'Min_Speed_quantile_95_values',
'Max_Speed_quantile_95_values', 'Mean_RangeVertical_max_values',
'Median_RangeVertical_max_values', 'Mode_RangeVertical_max_values',
'Standard_Deviation_RangeVertical_max_values',
'Variance_RangeVertical_max_values', 'Min_RangeVertical_max_values',
'Max_RangeVertical_max_values', 'Mean_RangeVertical_median_values',
'Median_RangeVertical_median_values', 'Mode_RangeVertical_median_values',
'Standard_Deviation_RangeVertical_median_values',
```

```
'Variance_RangeVertical_median_values', 'Min_RangeVertical_median_values',
'Max_RangeVertical_median_values', 'Mean_RangeVertical_min_values',
'Median_RangeVertical_min_values', 'Mode_RangeVertical_min_values',
'Standard_Deviation_RangeVertical_min_values',
'Variance_RangeVertical_min_values', 'Min_RangeVertical_min_values',
'Max_RangeVertical_min_values', 'Mean_RangeVertical_quantile_05_values',
'Median_RangeVertical_quantile_05_values',
'Mode_RangeVertical_quantile_05_values',
'Standard_Deviation_RangeVertical_quantile_05_values',
'Variance_RangeVertical_quantile_05_values',
'Min_RangeVertical_quantile_05_values', 'Max_RangeVertical_quantile_05_values',
'Mean_RangeVertical_quantile_95_values',
'Median_RangeVertical_quantile_95_values',
'Mode_RangeVertical_quantile_95_values',
'Standard_Deviation_RangeVertical_quantile_95_values',
'Variance_RangeVertical_quantile_95_values',
'Min_RangeVertical_quantile_95_values', 'Max_RangeVertical_quantile_95_values',
'Mean_temp_raw_events_weather', 'Median_temp_raw_events_weather',
'Mode_temp_raw_events_weather', 'Standard_Deviation_temp_raw_events_weather',
'Variance_temp_raw_events_weather', 'Min_temp_raw_events_weather',
'Max_temp_raw_events_weather', 'Mean_feels_like_raw_events_weather',
'Median_feels_like_raw_events_weather', 'Mode_feels_like_raw_events_weather',
'Standard_Deviation_feels_like_raw_events_weather',
'Variance_feels_like_raw_events_weather', 'Min_feels_like_raw_events_weather',
'Max_feels_like_raw_events_weather', 'Mean_Minimum_temp_raw_events_weather',
'Median_Minimum_temp_raw_events_weather',
'Mode_Minimum_temp_raw_events_weather',
'Standard_Deviation_Minimum_temp_raw_events_weather',
'Variance_Minimum_temp_raw_events_weather',
'Min_Minimum_temp_raw_events_weather', 'Max_Minimum_temp_raw_events_weather',
'Mean_Maximum_temp_raw_events_weather',
'Median_Maximum_temp_raw_events_weather',
'Mode_Maximum_temp_raw_events_weather',
'Standard_Deviation_Maximum_temp_raw_events_weather',
'Variance_Maximum_temp_raw_events_weather',
'Min_Maximum_temp_raw_events_weather', 'Max_Maximum_temp_raw_events_weather',
'Mean_Pressure_raw_events_weather', 'Median_Pressure_raw_events_weather',
'Mode_Pressure_raw_events_weather',
'Standard_Deviation_Pressure_raw_events_weather',
'Variance_Pressure_raw_events_weather', 'Min_Pressure_raw_events_weather',
'Max_Pressure_raw_events_weather', 'Mean_Humidity_raw_events_weather',
'Median_Humidity_raw_events_weather', 'Mode_Humidity_raw_events_weather',
'Standard_Deviation_Humidity_raw_events_weather',
'Variance_Humidity_raw_events_weather', 'Min_Humidity_raw_events_weather',
'Max_Humidity_raw_events_weather', 'Mean_Wind_Speed_raw_events_weather',
'Median_Wind_Speed_raw_events_weather', 'Mode_Wind_Speed_raw_events_weather',
'Standard_Deviation_Wind_Speed_raw_events_weather',
'Variance_Wind_Speed_raw_events_weather', 'Min_Wind_Speed_raw_events_weather',
'Max_Wind_Speed_raw_events_weather', 'Mean_Wind_Direction_raw_events_weather',
'Median_Wind_Direction_raw_events_weather',
```

```
'Mode_Wind_Direction_raw_events_weather',
'Standard_Deviation_Wind_Direction_raw_events_weather',
'Variance_Wind_Direction_raw_events_weather',
'Min_Wind_Direction_raw_events_weather',
'Max_Wind_Direction_raw_events_weather',
'Mean_Cloudiness_Percentage_raw_events_weather',
'Median_Cloudiness_Percentage_raw_events_weather',
'Mode_Cloudiness_Percentage_raw_events_weather',
'Standard_Deviation_Cloudiness_Percentage_raw_events_weather',
'Variance_Cloudiness_Percentage_raw_events_weather',
'Min_Cloudiness_Percentage_raw_events_weather',
'Max_Cloudiness_Percentage_raw_events_weather',
'Mean_Sunrise_Time_raw_events_weather',
'Median_Sunrise_Time_raw_events_weather',
'Mode_Sunrise_Time_raw_events_weather',
'Standard_Deviation_Sunrise_Time_raw_events_weather',
'Variance_Sunrise_Time_raw_events_weather',
'Min_Sunrise_Time_raw_events_weather', 'Max_Sunrise_Time_raw_events_weather',
'Mean_Sunset_Time_raw_events_weather', 'Median_Sunset_Time_raw_events_weather',
'Mode_Sunset_Time_raw_events_weather',
'Standard_Deviation_Sunset_Time_raw_events_weather',
'Variance_Sunset_Time_raw_events_weather',
'Min_Sunset_Time_raw_events_weather', 'Max_Sunset_Time_raw_events_weather',
'Mean_raw_lastKnownPoints_accuracy_values',
'Median_raw_lastKnownPoints_accuracy_values',
'Mode_raw_lastKnownPoints_accuracy_values',
'Standard_Deviation_raw_lastKnownPoints_accuracy_values',
'Variance_raw_lastKnownPoints_accuracy_values',
'Min_raw_lastKnownPoints_accuracy_values',
'Max_raw_lastKnownPoints_accuracy_values', 'Hour', 'Minute'
]

binary_cols = [
    'confirmNeeded', 'Weekend', 'Morning', 'Afternoon', 'Evening',
    'Minute_0-14', 'Minute_15-29', 'Minute_30-44', 'Minute_45-59'
]

one_hot_cols = [
    'current', 'Raw_startReason', 'Raw_stopReason', 'Day_of_Week'
]

multi_hot_cols = [
    'trip_events_type', 'trip_events_confirmNeeded', 'trip_events_asI15',
    'Weather_Main_raw_events_weather', 'Raw_type_counts',
    'Raw_accident_trigger_counts'
]
```

Ο κώδικας χωρίζει τα χαρακτηριστικά του συνόλου δεδομένων σε τέσσερις κατηγορίες με βάση τη φύση των δεδομένων τους και τον τρόπο που θα χρησιμοποιηθούν στις επερχόμενες διαδικασίες ανάλυσης και μοντελοποίησης.

Κατηγορίες:

- **Αριθμητικά Γνωρίσματα (numeric_cols):** Αυτή η κατηγορία περιλαμβάνει γνωρίσματα που περιέχουν αριθμητικές τιμές, όπως μετρήσεις επιτάχυνσης, ταχύτητας, διάρκειας και άλλα στατιστικά μέτρα. Αυτά τα γνωρίσματα είναι ιδανικά για χρήση σε στατιστικές αναλύσεις, καθώς και σε αλγορίθμους μηχανικής μάθησης που απαιτούν αριθμητικά δεδομένα.

- **Δυαδικά Γνωρίσματα (binary_cols):** Εδώ ανήκουν τα γνωρίσματα που παίρνουν δύο πιθανές τιμές (0 ή 1). Αυτά τα γνωρίσματα αντιπροσωπεύουν την ύπαρξη ή την απουσία ενός χαρακτηριστικού.

- **One-Hot Κωδικοποιημένα Γνωρίσματα (one_hot_cols):** Αυτή η κατηγορία περιλαμβάνει γνωρίσματα κατηγορικών δεδομένων που θα κωδικοποιηθούν χρησιμοποιώντας την τεχνική one-hot encoding. Κάθε κατηγορία μετατρέπεται σε μια νέα στήλη, όπου η παρουσία της κατηγορίας σημειώνεται με 1, ενώ η απουσία με 0.

- **Multi-Hot Κωδικοποιημένα Γνωρίσματα (multi_hot_cols):** Αυτή η κατηγορία περιλαμβάνει γνωρίσματα που μπορεί να περιέχουν πολλαπλές κατηγορίες ταυτόχρονα. Αυτά τα γνωρίσματα θα κωδικοποιηθούν σε πολλαπλές δυαδικές στήλες, με κάθε στήλη να δείχνει την παρουσία ή απουσία μιας συγκεκριμένης κατηγορίας.

3.3.17 Συμπλήρωση Απουσιών σε Αριθμητικά Γνωρίσματα

```
def fill_value(df, numeric_cols):
  for col in numeric_cols:
    df[col].interpolate(method ='linear', limit_direction ='both',
inplace=True)
```

Η συνάρτηση fill_value χρησιμοποιείται για τη συμπλήρωση των ελλιπών τιμών στις αριθμητικές στήλες ενός DataFrame.

Λεπτομέρειες:

- **interpolate(method='linear')**: Εκτελεί γραμμική παρεμβολή, υπολογίζοντας τις ελλείπουσες τιμές με βάση τις τιμές πριν και μετά από το κενό. Αυτό σημαίνει ότι οι ελλειπουσες τιμές αντικαθίστανται με τιμές που βρίσκονται σε μια ευθεία γραμμή μεταξύ των πλησιέστερων διαθέσιμων δεδομένων.
- **limit_direction='both'**: Εφαρμόζει την παρεμβολή προς και από τα δύο άκρα της στήλης, συμπληρώνοντας τιμές είτε βρίσκονται στην αρχή είτε στο τέλος της στήλης.
- **inplace=True**: Οι αλλαγές εφαρμόζονται απευθείας στο αρχικό DataFrame.

Η γραμμική παρεμβολή είναι χρήσιμη σε δεδομένα που αναμένονται να έχουν μια συνεχόμενη ή γραμμική τάση, επιτρέποντας την αναπλήρωση των απουσιών χωρίς την εισαγωγή μη ρεαλιστικών τιμών.

4.2.18 Η Τελική Συνάρτηση preprocess_dataframe

```
def preprocess_dataframe(df, numeric_cols):
    df['Date_Updated'] = pd.to_datetime(df['Date_Updated'])
    df['Season'] = df['Date_Updated'].dt.month.apply(lambda x: 'Spring' if 3 <=
x <= 5 else ('Summer' if 6 <= x <= 8 else ('Autumn' if 9 <= x <= 11 else
'Winter')))
    df['Day_of_Week'] = df['Date_Updated'].dt.dayofweek
    df['Weekend'] = df['Date_Updated'].dt.dayofweek.isin([5, 6]).astype(int)
    df['Hour'] = df['Date_Updated'].dt.hour
    df['Minute'] = df['Date_Updated'].dt.minute

    df = time_of_day(df)
    df = categorize_minutes(df)
    fill_value(df, numeric_cols)
    df['confirmNeeded'] = df['confirmNeeded'].astype(int)

    return df
```

Αυτή η συνάρτηση είναι σχεδιασμένη για να εμπλουτίζει το DataFrame με πρόσθετα χρονικά χαρακτηριστικά, να συμπληρώνει ελλιπή δεδομένα και να προετοιμάζει τα

δυαδικά γνωρίσματα για περαιτέρω ανάλυση. Στην τελική της μορφή, έχουν προστεθεί σημαντικές λειτουργίες για την εξασφάλιση της πληρότητας και της συνέπειας των δεδομένων.

Προσθήκες και Λειτουργίες

- **Συμπλήρωση Απουσιών:** Προστέθηκε η λειτουργία της συμπλήρωσης ελλειπόντων τιμών με γραμμική παρεμβολή.
    - Η συνάρτηση fill_value καλείται για να εντοπίσει και να συμπληρώσει τυχόν απουσίες στα αριθμητικά γνωρίσματα (numeric_cols). Αυτό εξασφαλίζει ότι τα δεδομένα είναι πλήρη και δεν περιέχουν κενά που θα μπορούσαν να επηρεάσουν αρνητικά τις επόμενες αναλύσεις ή τα μοντέλα πρόβλεψης.
- **Μετατροπή Δυαδικών Γνωρισμάτων σε Αριθμητικά:** Προστέθηκε η λειτουργία μετατροπής των δυαδικών γνωρισμάτων σε ακέραιες τιμές.
    - Η στήλη confirmNeeded, η οποία περιέχει τιμές που αντιπροσωπεύουν την παρουσία ή απουσία μιας κατάστασης, μετατρέπεται σε ακέραιο αριθμό (0 ή 1).

Με αυτές τις προσθήκες, η preprocess_dataframe είναι πλέον μια ολοκληρωμένη συνάρτηση που προετοιμάζει τα δεδομένα με έναν συνεπή και αξιόπιστο τρόπο, διασφαλίζοντας την ποιότητα και τη χρησιμότητα του DataFrame για επόμενες αναλύσεις.

3.3.19 Εφαρμογή της Προεπεξεργασίας για Κάθε Οδηγό

```
df_driver1 = preprocess_dataframe(df_driver1, numeric_cols)
df_driver2 = preprocess_dataframe(df_driver2, numeric_cols)
df_driver3 = preprocess_dataframe(df_driver3, numeric_cols)
df_driver4 = preprocess_dataframe(df_driver4, numeric_cols)
df_driver5 = preprocess_dataframe(df_driver5, numeric_cols)
df_driver6 = preprocess_dataframe(df_driver6, numeric_cols)
df_driver7 = preprocess_dataframe(df_driver7, numeric_cols)

df = pd.concat([df_driver1, df_driver2, df_driver3, df_driver4, df_driver5,
df_driver6, df_driver7], axis=0)
df.reset_index(drop=True, inplace=True)
```

Για κάθε οδηγό, καλείται η συνάρτηση preprocess_dataframe, η οποία εφαρμόζει όλα τα βήματα προεπεξεργασίας στα δεδομένα. Αυτή η διαδικασία εξασφαλίζει ότι τα δεδομένα από κάθε οδηγό είναι πλήρως προετοιμασμένα και ομοιογενή.

3.2.20 Συγχώνευση των Επεξεργασμένων Δεδομένων

Αφού ολοκληρωθεί η προεπεξεργασία για κάθε οδηγό, τα επεξεργασμένα DataFrames συγχωνεύονται σε ένα ενιαίο DataFrame df. Η συνένωση πραγματοποιείται κατά μήκος του άξονα 0 (κάθετη συγχώνευση), τοποθετώντας τα δεδομένα κάθε οδηγού κάτω από τα δεδομένα του προηγούμενου.

- **pd.concat()**: Χρησιμοποιείται για να συγχωνεύσει τα DataFrames σε ένα ενιαίο σύνολο δεδομένων.

- **reset_index(drop=True)**: Επαναφέρει τους δείκτες του τελικού DataFrame ώστε να είναι συνεχείς και να ξεκινούν από το 0, εξαλείφοντας τους παλιούς δείκτες από τα αρχικά DataFrames.

3.3.21 Έλεγχος των Αριθμητικών Γνωρισμάτων για Απουσίες και Τύπους Δεδομένων

```
for column in numeric_cols:
    print(f"Column: {column}")
    print(f"Type: {df[column].dtype}")
    print(f"NaN count: {df[column].isna().sum()}")
    print()
```

Ο παραπάνω κώδικας εκτελεί έναν έλεγχο σε όλες τις αριθμητικές στήλες του τελικού DataFrame df για να διασφαλίσει ότι δεν υπάρχουν ελλείποντα δεδομένα και ότι οι τύποι δεδομένων είναι σωστοί.

- **df[column].dtype**: Ελέγχει τον τύπο δεδομένων της στήλης, επιβεβαιώνοντας ότι είναι κατάλληλος για αριθμητική ανάλυση.

- **df[column].isna().sum()**: Υπολογίζει τον αριθμό των ελλειπουσων τιμών (NaN) στη στήλη, διασφαλίζοντας ότι δεν υπάρχουν κενά που θα μπορούσαν να επηρεάσουν την ανάλυση ή τη μοντελοποίηση.

3.3.22 Έλεγχος για Υψηλή Συσχέτιση και Διαγραφή Συσχετιζόμενων Γνωρισμάτων

```
correlation_matrix = df[numeric_cols].corr()

def find_high_correlation_columns(correlation_matrix, column_name,
threshold=0.9):
    high_corr_columns =
correlation_matrix[column_name][abs(correlation_matrix[column_name]) >=
threshold].index.tolist()
    return high_corr_columns


for col in df[numeric_cols]:
    high_corr_columns = find_high_correlation_columns(correlation_matrix, col)
    high_corr_columns = [c for c in high_corr_columns if c != col]
    print(f'Columns highly correlated with:\t{col}\n{high_corr_columns}')
    print('\n', '\n')
```

```
def find_highly_correlated_pairs(correlation_matrix, threshold=0.9):
    high_corr_pairs = []

    for col in correlation_matrix.columns:
        for row in correlation_matrix.index:
            if col != row and abs(correlation_matrix.loc[row, col]) >=
threshold:
                high_corr_pairs.append((row, col,
round(correlation_matrix.loc[row, col], 4)))

    high_corr_df = pd.DataFrame(high_corr_pairs, columns=['Feature 1', 'Feature
2', 'Correlation'])
    high_corr_df = high_corr_df.sort_values(by='Correlation',
ascending=False).reset_index(drop=True)
    return high_corr_df


high_corr_df = find_highly_correlated_pairs(correlation_matrix, threshold=0.9)
top_high_corr_df = high_corr_df.head(5)

print("Top 5 χαρακτηριστικά με την υψηλότερη συσχέτιση:")
print(tabulate(top_high_corr_df, headers='keys', tablefmt='pretty'))
```

Top 5 χαρακτηριστικά με την υψηλότερη συσχέτιση:

	Feature 1	Feature 2	Correlation
0	Variance_Speed_quantile_95_values	Variance_SpeedStart_values	1.0
1	Max_Speed_median_values	Max_SpeedMedian_values	1.0
2	Standard_Deviation_Speed_median_values	Standard_Deviation_SpeedMedian_values	1.0
3	Variance_Speed_median_values	Variance_SpeedMedian_values	1.0
4	Median_Speed_median_values	Median_SpeedMedian_values	1.0

Εικ.3: Παράδειγμα των κορυφαίων 5 ζευγών χαρακτηριστικών με την υψηλότερη συσχέτιση.

```
drop_numeric_cols = [ 'ecoDepreciation', 'durationMinutes',
'phoneUsageWithSpeedingDurationMinutes', 'Median_trip_events_values',
'Variance_trip_events_values', 'Variance_acceleration_direct_values',
'Variance_acceleration_direct_end_values',
'Variance_acceleration_lateral_values',
'Variance_acceleration_vertical_values',
'Variance_acceleration_vertical_end_values', 'Median_PureDuration_values',
'Min_PureDuration_values', 'Variance_PrevEventSpeed_values',
'Median_Reliability_values', 'Variance_Reliability_values',
'Mode_Speed_median_values', 'Min_Speed_median_values',
'Variance_SpeedStop_values', 'Max_Accuracy_quantile_95_values',
'Variance_RangeDirect_median_values',
'Variance_RangeDirect_quantile_05_values',
'Variance_RangeDirect_quantile_95_values',
'Variance_RangeLateral_median_values',
'Variance_RangeLateral_quantile_05_values',
'Variance_RangeLateral_quantile_95_values', 'Min_Speed_quantile_95_values',
'Min_Speed_quantile_05_values', 'Variance_RangeVertical_median_values',
'Variance_RangeVertical_min_values',
'Variance_RangeVertical_quantile_05_values',
'Variance_RangeVertical_quantile_95_values',
'Median_Pressure_raw_events_weather', 'Mode_Pressure_raw_events_weather',
'Variance_Humidity_raw_events_weather',
'Median_Wind_Direction_raw_events_weather',
'Mode_Wind_Direction_raw_events_weather',
'Variance_Cloudiness_Percentage_raw_events_weather',
'Mean_PureDuration_values', 'Mode_PureDuration_values',
'Variance_PureDuration_values', 'Max_PureDuration_values',
'Max_Speed_median_values', 'Max_Speed_min_values',
'Max_Speed_quantile_05_values', 'Variance_Accuracy_max_values',
'Standard_Deviation_Accuracy_quantile_95_values', 'Min_Accuracy_max_values',
'Min_Accuracy_min_values', 'Min_Accuracy_quantile_05_values',
'Min_Accuracy_quantile_95_values', 'Max_Accuracy_min_values',
'Max_Accuracy_quantile_05_values', 'Variance_RangeDirect_max_values',
'Max_RangeDirect_max_values', 'Variance_RangeDirect_min_values',
'Min_RangeDirect_min_values', 'Variance_RangeLateral_max_values',
```

```
'Max_RangeLateral_max_values', 'Variance_RangeLateral_min_values',
'Min_RangeLateral_min_values', 'Variance_RangeVertical_max_values',
'Max_RangeVertical_max_values', 'Variance_temp_raw_events_weather',
'Variance_feels_like_raw_events_weather',
'Variance_Minimum_temp_raw_events_weather',
'Median_Humidity_raw_events_weather', 'Mode_Humidity_raw_events_weather',
'Min_Humidity_raw_events_weather', 'Max_Humidity_raw_events_weather',
'Median_Wind_Speed_raw_events_weather', 'Mode_Wind_Speed_raw_events_weather',
'Max_Wind_Speed_raw_events_weather',
'Median_Cloudiness_Percentage_raw_events_weather',
'Mode_Cloudiness_Percentage_raw_events_weather',
'Min_Cloudiness_Percentage_raw_events_weather',
'Max_Cloudiness_Percentage_raw_events_weather',
'Variance_Sunrise_Time_raw_events_weather',
'Standard_Deviation_Sunset_Time_raw_events_weather',
'Median_raw_lastKnownPoints_accuracy_values',
'Mode_raw_lastKnownPoints_accuracy_values',
'Min_raw_lastKnownPoints_accuracy_values',
'Max_raw_lastKnownPoints_accuracy_values', 'Mean_Accuracy_max_values',
'Median_Accuracy_median_values', 'Mean_Accuracy_quantile_95_values',
'Median_Accuracy_quantile_95_values', 'Mean_Accuracy_min_values',
'Median_Accuracy_min_values', 'Mean_Accuracy_quantile_05_values',
'Median_Accuracy_quantile_05_values',
'Standard_Deviation_Accuracy_median_values', 'Variance_Accuracy_min_values',
'Variance_Accuracy_quantile_05_values',
'Standard_Deviation_Accuracy_quantile_05_values', 'Mean_Speed_min_values',
'Mean_Speed_quantile_05_values', 'Median_Speed_quantile_05_values',
'Mean_PrevEventSpeed_values', 'Median_PrevEventSpeed_values',
'Mean_SpeedMedian_values', 'Median_SpeedStart_values', 'Mean_SpeedStop_values',
'Median_SpeedStop_values', 'Median_Speed_max_values',
'Mean_Speed_median_values', 'Median_Speed_median_values',
'Median_Speed_quantile_95_values', 'Median_temp_raw_events_weather',
'Mode_temp_raw_events_weather', 'Min_temp_raw_events_weather',
'Max_temp_raw_events_weather', 'Mean_feels_like_raw_events_weather',
'Median_feels_like_raw_events_weather', 'Mode_feels_like_raw_events_weather',
'Min_feels_like_raw_events_weather', 'Max_feels_like_raw_events_weather',
'Mean_Minimum_temp_raw_events_weather',
'Median_Minimum_temp_raw_events_weather',
'Mode_Minimum_temp_raw_events_weather', 'Min_Minimum_temp_raw_events_weather',
'Max_Minimum_temp_raw_events_weather', 'Mean_Maximum_temp_raw_events_weather',
'Median_Maximum_temp_raw_events_weather',
'Mode_Maximum_temp_raw_events_weather', 'Min_Maximum_temp_raw_events_weather',
'Max_Maximum_temp_raw_events_weather', 'Variance_SpeedMedian_values',
'Variance_Speed_median_values', 'Standard_Deviation_Speed_min_values',
'Standard_Deviation_Speed_quantile_05_values',
'Variance_Speed_quantile_05_values', 'Standard_Deviation_Speed_median_values',
'Variance_raw_lastKnownPoints_accuracy_values', 'Mean_SpeedStart_values',
'Standard_Deviation_SpeedStart_values', 'Variance_SpeedStart_values',
'Max_SpeedStart_values', 'Mean_Speed_max_values', 'Mode_Speed_max_values',
'Standard_Deviation_Speed_max_values', 'Variance_Speed_max_values',
```

```
'Max_Speed_max_values', 'Mean_Speed_quantile_95_values',
'Mode_Speed_quantile_95_values', 'Standard_Deviation_Speed_quantile_95_values',
'Variance_Speed_quantile_95_values',
'Standard_Deviation_temp_raw_events_weather',
'Standard_Deviation_feels_like_raw_events_weather',
'Standard_Deviation_Minimum_temp_raw_events_weather',
'Median_Sunrise_Time_raw_events_weather',
'Mode_Sunrise_Time_raw_events_weather', 'Min_Sunrise_Time_raw_events_weather',
'Max_Sunrise_Time_raw_events_weather', 'Mean_Sunset_Time_raw_events_weather',
'Median_Sunset_Time_raw_events_weather', 'Mode_Sunset_Time_raw_events_weather',
'Min_Sunset_Time_raw_events_weather', 'Max_Sunset_Time_raw_events_weather',
'Minute']

numeric_cols = [col for col in numeric_cols if col not in drop_numeric_cols]
```

Ο κώδικας αναλύει τις συσχετίσεις μεταξύ των αριθμητικών γνωρισμάτων του DataFrame και εντοπίζει τα γνωρίσματα που παρουσιάζουν υψηλή συσχέτιση μεταξύ τους. Σκοπός είναι να αφαιρεθούν τα πλεονασματικά γνωρίσματα, τα οποία θα μπορούσαν να οδηγήσουν σε υπερβολική πολυπλοκότητα κατά την εκπαίδευση των μοντέλων.

Τα Βασικά Βήματα του Κώδικα είναι:

- **Υπολογισμός του Πίνακα Συσχέτισης**: Ο πρώτος στόχος είναι η δημιουργία του πίνακα συσχέτισης για όλα τα αριθμητικά γνωρίσματα, χρησιμοποιώντας τον συντελεστή συσχέτισης Pearson. Ο πίνακας αυτός δίνει μια συνολική εικόνα για το πόσο στενά συνδέονται τα γνωρίσματα μεταξύ τους.

- **Εύρεση Υψηλά Συσχετιζόμενων Γνωρισμάτων**: Η συνάρτηση find_high_correlation_columns αναλύει τον πίνακα συσχέτισης για κάθε στήλη και επιστρέφει μια λίστα με τις στήλες που έχουν συσχέτιση ίση ή μεγαλύτερη από το κατώφλι που έχει οριστεί (threshold = 0.9). Αυτά τα γνωρίσματα θεωρούνται πλεονασματικά, καθώς φέρουν παρόμοια πληροφορία.

- **Έλεγχος και Εκτύπωση Υψηλά Συσχετιζόμενων Γνωρισμάτων**: Για κάθε αριθμητική στήλη στο DataFrame, γίνεται έλεγχος για να εντοπιστούν τα υπόλοιπα γνωρίσματα που έχουν υψηλή συσχέτιση με αυτή. Τα αποτελέσματα εκτυπώνονται ώστε να επισημανθούν τα γνωρίσματα που θα μπορούσαν να διαγραφούν.

Παρατηρήσεις για Συσχετιζόμενα Γνωρίσματα

Η ανάλυση της συσχέτισης αποκάλυψε ότι πολλά γνωρίσματα είχαν υψηλή συσχέτιση (>= 0.9) μεταξύ τους. Αυτά τα γνωρίσματα παρέχουν παρόμοια πληροφορία, γεγονός που μπορεί να αυξήσει την πολυπλοκότητα του μοντέλου χωρίς να βελτιώνει την απόδοσή του.

Γνωρίσματα που Κρατήθηκαν:

Μετά από προσεκτική αξιολόγηση, επιλέχθηκαν να παραμείνουν στο τελικό σύνολο δεδομένων γνωρίσματα που θεωρήθηκαν σημαντικά για την ανάλυση και την πρόβλεψη ασφαλούς οδηγικής συμπεριφοράς. Αυτά περιλαμβάνουν:

- **Χαρακτηριστικά Οδήγησης:** Όπως η διάρκεια των ταξιδιών, η ταχύτητα σε διάφορα σημεία της διαδρομής, και οι στατιστικοί δείκτες της οδήγησης όπως η μέση, τυπική απόκλιση και διάμεσος τιμή των γεγονότων ταξιδιού.
- **Μετρήσεις Επιτάχυνσης και Αντίστροφης Αντίδρασης:** Όπως οι τυπικές αποκλίσεις και οι τιμές επιτάχυνσης σε διάφορες κατευθύνσεις.
- **Μετρήσεις Καιρικών Συνθηκών:** Όπως η μέση πίεση, η κατεύθυνση και η ταχύτητα του ανέμου, καθώς και ποσοστά συννεφιάς και ηλιοφάνειας.
- **Μετρήσεις Ακριβείας Τοποθεσίας:** Όπως η ακρίβεια των τελευταίων καταγεγραμμένων σημείων οδήγησης.

Γνωρίσματα που Διαγράφηκαν:

Ορισμένα γνωρίσματα κρίθηκαν πλεονασματικά λόγω της υψηλής συσχέτισης τους με άλλα γνωρίσματα και αφαιρέθηκαν από το τελικό σύνολο δεδομένων. Αυτά περιλαμβάνουν:

- **Διπλά Χαρακτηριστικά Οδήγησης:** Όπως η διάρκεια του ταξιδιού σε λεπτά (durationMinutes) που ήταν παρόμοια με το Trip_Duration.

- **Πλεονασματικές Στατιστικές Μετρήσεις:** Όπως οι διακυμάνσεις και οι τυπικές αποκλίσεις για χαρακτηριστικά που ήδη περιλαμβάνονταν με άλλες μορφές, π.χ., Variance_trip_events_values και Variance_acceleration_direct_values.
- **Πλεονασματικές Μετρήσεις Καιρικών Συνθηκών:** Όπως οι ελάχιστες, μέγιστες και διακυμάνσεις μετρήσεων καιρού που ήταν ήδη αναπαραστάσεις άλλων χαρακτηριστικών, π.χ., Variance_Humidity_raw_events_weather και Variance_Sunset_Time_raw_events_weather.
- **Πλεονασματικές Μετρήσεις Ακριβείας:** Όπως διαφορετικές μορφές της ακρίβειας τοποθεσίας (Min_Accuracy_max_values, Max_Accuracy_quantile_95_values), οι οποίες δεν προσέφεραν επιπλέον πληροφορία στη μοντελοποίηση.

Συμπεράσματα

Η διαδικασία εντοπισμού και διαγραφής πλεονασματικών γνωρισμάτων βάσει συσχέτισης είναι κρίσιμη για τη βελτιστοποίηση της απόδοσης των μοντέλων μηχανικής μάθησης. Με την απομάκρυνση των γνωρισμάτων που δεν προσφέρουν επιπλέον πληροφορία, μειώνεται η πολυπλοκότητα και αυξάνεται η ικανότητα του μοντέλου να γενικεύει σε νέα δεδομένα.

3.3.23 Οπτικοποίηση και Αποθήκευση Τελικών Προεπεξεργασμένων Δεδομένων

```
plt.figure(figsize = (160, 160))
sns.heatmap(df[numeric_cols].corr(), annot = True, cmap="YlGnBu")
plt.show()
```

```
selected_cols = ['safety'] + numeric_cols + binary_cols + one_hot_cols +
multi_hot_cols
df_driver1 = df_driver1[selected_cols]
df_driver1.to_csv('preprocess_df_driver1.csv', index=None)

df_driver2 = df_driver2[selected_cols]
df_driver2.to_csv('preprocess_df_driver2.csv', index=None)

df_driver3 = df_driver3[selected_cols]
df_driver3.to_csv('preprocess_df_driver3.csv', index=None)

df_driver4 = df_driver4[selected_cols]
df_driver4.to_csv('preprocess_df_driver4.csv', index=None)
```

```
df_driver5 = df_driver5[selected_cols]
df_driver5.to_csv('preprocess_df_driver5.csv', index=None)

df_driver6 = df_driver6[selected_cols]
df_driver6.to_csv('preprocess_df_driver6.csv', index=None)

df_driver7 = df_driver7[selected_cols]
df_driver7.to_csv('preprocess_df_driver7.csv', index=None)
```

Οπτικοποίηση του Τελικού Πίνακα Συσχετίσεων

Για να κατανοήσουμε καλύτερα τις συσχετίσεις μεταξύ των επιλεγμένων γνωρισμάτων μετά τη διαδικασία της προεπεξεργασίας, δημιουργείται ένας πίνακας heatmap. Ο πίνακας αυτός παρουσιάζει τη συσχέτιση μεταξύ των αριθμητικών γνωρισμάτων που έχουν απομείνει στο σύνολο δεδομένων. Η οπτικοποίηση χρησιμοποιεί την χρωματική κλίμακα YlGnBu, η οποία καθιστά εύκολη την αναγνώριση των επιπέδων συσχέτισης:

- **Σκοτεινές αποχρώσεις του μπλε**: Αντιπροσωπεύουν ισχυρές αρνητικές συσχετίσεις.
- **Ανοιχτές αποχρώσεις του πράσινου**: Αντιπροσωπεύουν ισχυρές θετικές συσχετίσεις.

Αυτός ο πίνακας προσφέρει μια σαφή οπτική αναπαράσταση του βαθμού συσχέτισης μεταξύ των χαρακτηριστικών, βοηθώντας στην περαιτέρω ανάλυση και κατανόηση των δεδομένων.

Δημιουργία Τελικών Προεπεξεργασμένων Συνόλων Δεδομένων για Κάθε Οδηγό

Μετά την επιλογή των πιο σημαντικών γνωρισμάτων και την αφαίρεση των συσχετιζόμενων γνωρισμάτων, το επόμενο βήμα είναι η δημιουργία των τελικών προεπεξεργασμένων συνόλων δεδομένων για κάθε οδηγό. Ο κώδικας παραπάνω εκτελεί τα εξής βήματα:

- **Επιλογή των Τελικών Γνωρισμάτων:** Τα γνωρίσματα που επιλέχθηκαν να διατηρηθούν για την τελική ανάλυση αποθηκεύονται στη λίστα selected_cols. Αυτή η λίστα περιλαμβάνει:
    - Το γνώρισμα safety, το οποίο είναι το target variable.
    - Τα αριθμητικά (numeric_cols), δυαδικά (binary_cols), one-hot encoded (one_hot_cols), και multi-hot encoded (multi_hot_cols) γνωρίσματα που έχουν διατηρηθεί μετά τη διαδικασία της προεπεξεργασίας.
- **Δημιουργία και Αποθήκευση Προεπεξεργασμένων Δεδομένων:** Για κάθε οδηγό, δημιουργείται ένα υποσύνολο δεδομένων που περιλαμβάνει μόνο τα επιλεγμένα γνωρίσματα.

3.3.24 Φόρτωση, Συγχώνευση και Προετοιμασία Δεδομένων για Μοντελοποίηση

```
folder = '/content/drive/My Drive/DrivingData'

df_driver1 = pd.read_csv(folder+'/preprocess_df_driver1.csv')
df_driver2 = pd.read_csv(folder+'/preprocess_df_driver2.csv')
df_driver3 = pd.read_csv(folder+'/preprocess_df_driver3.csv')
df_driver4 = pd.read_csv(folder+'/preprocess_df_driver4.csv')
df_driver5 = pd.read_csv(folder+'/preprocess_df_driver5.csv')
df_driver6 = pd.read_csv(folder+'/preprocess_df_driver6.csv')
df_driver7 = pd.read_csv(folder+'/preprocess_df_driver7.csv')

df = pd.concat([df_driver1, df_driver2, df_driver3, df_driver4, df_driver5,
df_driver6, df_driver7], axis=0)
df.reset_index(drop=True, inplace=True)
Y = df['safety']
X=df.drop('safety',axis=1)
```

Φόρτωση των Προεπεξεργασμένων Συνόλων Δεδομένων

Μετά την ολοκλήρωση της προεπεξεργασίας και την αποθήκευση των δεδομένων σε αρχεία CSV για κάθε οδηγό, το επόμενο βήμα είναι η φόρτωση αυτών των αρχείων για την δημιουργία του μοντέλου.

- **Ορισμός του Κύριου Φακέλου Δεδομένων**: Η μεταβλητή folder ορίζει τη διαδρομή του κύριου φακέλου όπου βρίσκονται αποθηκευμένα τα αρχεία με τα προεπεξεργασμένα δεδομένα.

- **Φόρτωση των Προεπεξεργασμένων Δεδομένων**: Για κάθε οδηγό (driver1 έως driver7), τα προεπεξεργασμένα δεδομένα φορτώνονται από τα αντίστοιχα αρχεία CSV. Η βιβλιοθήκη pandas χρησιμοποιείται για τη φόρτωση των δεδομένων σε διαφορετικά DataFrames.

Συγχώνευση των Συνόλων Δεδομένων και Προετοιμασία για Μοντελοποίηση

Αφού τα δεδομένα για κάθε οδηγό έχουν φορτωθεί, τα επόμενα βήματα περιλαμβάνουν τη συγχώνευση όλων των συνόλων δεδομένων σε ένα ενιαίο DataFrame και την προετοιμασία τους για μοντελοποίηση:

- **Συγχώνευση των Συνόλων Δεδομένων**: Όλα τα DataFrames συγχωνεύονται σε ένα ενιαίο DataFrame df χρησιμοποιώντας τη συνάρτηση pd.concat(). Η παράμετρος axis=0 εξασφαλίζει ότι τα DataFrames θα ενωθούν κατά μήκος των γραμμών (κάθετες συγχωνεύσεις).

- **Επαναφορά των Δεικτών**: Μετά τη συγχώνευση, οι δείκτες του DataFrame επαναρυθμίζονται με τη μέθοδο reset_index(drop=True), ώστε οι σειρές να είναι αριθμημένες με μοναδικούς δείκτες από το 0 και μετά, χωρίς να διατηρούνται οι παλαιοί δείκτες από τα αρχικά DataFrames.

- **Διαχωρισμός των Δεδομένων σε Χαρακτηριστικά και Ετικέτες**:
    - Τα δεδομένα διαχωρίζονται σε δύο σύνολα, τα χαρακτηριστικά (X) και τις ετικέτες (Y).
    - Το Y περιέχει τη στήλη safety, η οποία είναι ο στόχος (ετικέτα) για πρόβλεψη.
    - Το X περιέχει όλες τις υπόλοιπες στήλες, εξαιρώντας τη στήλη safety.

4.3.25 Κλιμάκωση των Δεδομένων

```
def scale_data(X_train, X_test, numeric_cols):
    X_train1 = copy.deepcopy(X_train)
    X_test1 = copy.deepcopy(X_test)

    scaler1 = preprocessing.MinMaxScaler()
    scaler1.fit(X_train1[numeric_cols])

    Xn_train1 = scaler1.transform(X_train1[numeric_cols])
```

```
    Xn_test1 = scaler1.transform(X_test1[numeric_cols])

    X_train1[numeric_cols] = Xn_train1
    X_test1[numeric_cols] = Xn_test1

    X_train2 = copy.deepcopy(X_train)
    X_test2 = copy.deepcopy(X_test)

    scaler2 = preprocessing.StandardScaler()
    scaler2.fit(X_train2[numeric_cols])

    Xn_train2 = scaler2.transform(X_train2[numeric_cols])
    Xn_test2 = scaler2.transform(X_test2[numeric_cols])

    X_train2[numeric_cols] = Xn_train2
    X_test2[numeric_cols] = Xn_test2

    return X_train1, X_test1, scaler1, X_train2, X_test2, scaler2
```

Η κλιμάκωση των δεδομένων είναι ένα κρίσιμο βήμα στη διαδικασία προετοιμασίας δεδομένων για μοντελοποίηση, καθώς βοηθά στη βελτίωση της απόδοσης των αλγορίθμων μηχανικής μάθησης. Η συνάρτηση scale_data περιγράφει τη διαδικασία κλιμάκωσης με δύο διαφορετικές μεθόδους, παρέχοντας τη δυνατότητα πειραματισμού για την καλύτερη επιλογή κλιμάκωσης.

Λειτουργία της Συνάρτησης scale_data

- **Δημιουργία Αντιγράφων των Δεδομένων**:
    - Η συνάρτηση ξεκινά με τη δημιουργία αντιγράφων των συνόλων δεδομένων εκπαίδευσης (X_train) και δοκιμής (X_test).
- **Κλιμάκωση με Min-Max Scaling**:
    - Ο πρώτος τρόπος κλιμάκωσης χρησιμοποιεί τον MinMaxScaler από τη βιβλιοθήκη sklearn.preprocessing. Αυτός ο scaler μετασχηματίζει τις τιμές των χαρακτηριστικών ώστε να βρίσκονται εντός ενός εύρους τιμών, συνήθως μεταξύ 0 και 1. Είναι ιδιαίτερα χρήσιμο όταν τα δεδομένα έχουν διαφορετικά εύρη τιμών.
    - Ο scaler εφαρμόζεται πρώτα στα δεδομένα εκπαίδευσης και στη συνέχεια στα δεδομένα δοκιμής για να εξασφαλιστεί συνέπεια στην κλίμακα

- **Κλιμάκωση με Standard Scaling**:
    - Ο δεύτερος τρόπος κλιμάκωσης χρησιμοποιεί τον StandardScaler, ο οποίος μετασχηματίζει τις τιμές των χαρακτηριστικών έτσι ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1. Αυτό είναι χρήσιμο όταν τα δεδομένα έχουν διαφορετικά μέσα και αποκλίσεις.
    - Όπως και με τον MinMaxScaler, ο StandardScaler εφαρμόζεται τόσο στα δεδομένα εκπαίδευσης όσο και στα δεδομένα δοκιμής.
- **Επιστροφή των Κλιμακωμένων Δεδομένων**:
    - Η συνάρτηση επιστρέφει δύο σετ κλιμακωμένων δεδομένων μαζί με τους scalers που χρησιμοποιήθηκαν για τη μετατροπή. Αυτό επιτρέπει την επαναχρησιμοποίηση των scalers για μελλοντική χρήση.
    - Τα επιστρεφόμενα δεδομένα περιλαμβάνουν:
        - **X_train1**, **X_test1**: Τα κλιμακωμένα δεδομένα με χρήση του MinMaxScaler.
        - **scaler1**: Ο MinMaxScaler που χρησιμοποιήθηκε.
        - **X_train2**, **X_test2**: Τα κλιμακωμένα δεδομένα με χρήση του StandardScaler.
        - **scaler2**: Ο StandardScaler που χρησιμοποιήθηκε.

3.3.26 Μετατροπή Κατηγορικών Δεδομένων με One-Hot Encoding

```
def create_fit_transform(X_train, one_hot_cols):
    encoder = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
    encoder.fit(X_train[one_hot_cols])
    encoded_cols = list(encoder.get_feature_names_out(one_hot_cols))
    encoded_data = encoder.transform(X_train[one_hot_cols])
    encoded_df = pd.DataFrame(encoded_data, columns=encoded_cols,
index=X_train.index)
    X_train = pd.concat([X_train, encoded_df], axis=1)
    X_train.drop(one_hot_cols, axis=1, inplace=True)
    return X_train, encoder, encoded_cols


def transform_data(dataframe, encoder, encoded_cols, one_hot_cols):
```

```
    encoded_data = encoder.transform(dataframe[one_hot_cols])
    encoded_df = pd.DataFrame(encoded_data, columns=encoded_cols,
index=dataframe.index)
    dataframe = pd.concat([dataframe, encoded_df], axis=1)
    dataframe.drop(one_hot_cols, axis=1, inplace=True)
    return dataframe
```

Η μετατροπή των κατηγορικών δεδομένων σε μορφή που είναι κατανοητή από τους αλγόριθμους μηχανικής μάθησης αποτελεί κρίσιμο βήμα στη διαδικασία προετοιμασίας δεδομένων. Η τεχνική που χρησιμοποιείται για αυτήν τη μετατροπή είναι το one-hot encoding, το οποίο δημιουργεί δυαδικά χαρακτηριστικά για κάθε μοναδική κατηγορία ενός κατηγορικού χαρακτηριστικού.

Χρήση της κλάσης OneHotEncoder:

- Χρησιμοποιείται η κλάση OneHotEncoder από τη βιβλιοθήκη sklearn.preprocessing για την κωδικοποίηση κατηγορικών δεδομένων.
- Η μέθοδος fit_transform χρησιμοποιείται στα δεδομένα εκπαίδευσης (X_train) για να προσαρμοστεί ο encoder στα κατηγορικά χαρακτηριστικά και να τα μετατρέψει σε δυαδικά χαρακτηριστικά.
- Η μέθοδος transform εφαρμόζεται στα νέα δεδομένα (π.χ., δεδομένα δοκιμής) για να τα μετατρέψει με τον ίδιο encoder που έχει ήδη προσαρμοστεί στα δεδομένα εκπαίδευσης.

Ο κώδικας περιγράφει τη διαδικασία αυτή, χρησιμοποιώντας δύο βασικές συναρτήσεις, την create_fit_transform και την transform_data.

Λειτουργία της Συνάρτησης create_fit_transform

Η συνάρτηση create_fit_transform εφαρμόζει το one-hot encoding στα δεδομένα εκπαίδευσης:

- **Προσαρμογή και Μετασχηματισμός**:
    - Ο encoder προσαρμόζεται στα κατηγορικά χαρακτηριστικά που ορίζονται από τη λίστα one_hot_cols.
    - Τα χαρακτηριστικά αυτά μετασχηματίζονται σε νέα δυαδικά χαρακτηριστικά.

- **Ενημέρωση του DataFrame**:
    - Τα νέα κωδικοποιημένα χαρακτηριστικά προστίθενται στο X_train, ενώ τα αρχικά κατηγορικά χαρακτηριστικά αφαιρούνται.
    - Η συνάρτηση επιστρέφει το ενημερωμένο X_train, τον encoder, και τη λίστα με τα νέα κωδικοποιημένα χαρακτηριστικά (encoded_cols).

Λειτουργία της Συνάρτησης transform_data

Η συνάρτηση transform_data χρησιμοποιείται για να μετασχηματίσει νέα δεδομένα (π.χ., δεδομένα δοκιμής) με τον ίδιο encoder που έχει ήδη προσαρμοστεί στα δεδομένα εκπαίδευσης:

- **Μετασχηματισμός Νέων Δεδομένων**:
    - Τα κατηγορικά χαρακτηριστικά των νέων δεδομένων μετατρέπονται στα αντίστοιχα δυαδικά χαρακτηριστικά χρησιμοποιώντας τον ήδη προσαρμοσμένο encoder.
- **Ενημέρωση του DataFrame**:
    - Τα κωδικοποιημένα χαρακτηριστικά προστίθενται στο αρχικό σύνολο δεδομένων, ενώ τα αρχικά κατηγορικά χαρακτηριστικά αφαιρούνται.

3.3.27 Διαχείριση του Multi-Hot Encoding

```
def extract_unique_keys(df_column):
    unique_keys = set()
    for dict_str in df_column:
        dict_from_str = ast.literal_eval(dict_str)
        for key in dict_from_str.keys():
            unique_keys.add(key)
    return unique_keys


def check_key_presence(dict_str, unique_type):
    dict_from_str = ast.literal_eval(dict_str)
    if unique_type in dict_from_str.keys():
        return 1
    else:
        return 0
```

```
def create_presence_columns(dataframe, column_name, unique_keys):
    result_columns = []
    for unique_key in unique_keys:
        result_columns.append(dataframe[column_name].apply(lambda x:
check_key_presence(x, unique_key)))
    result_df = pd.concat(result_columns, axis=1)
    result_df.columns = [f'{column_name}_{col}' for col in unique_keys]
    dataframe = pd.concat([dataframe, result_df], axis=1)
    return dataframe
```

Στη διαδικασία προετοιμασίας δεδομένων, είναι συχνό φαινόμενο να υπάρχουν χαρακτηριστικά που περιέχουν πολλαπλές τιμές για κάθε εγγραφή, συχνά με τη μορφή λεξικών που αντιστοιχούν σε κλειδιά και τιμές. Αυτά τα χαρακτηριστικά απαιτούν ειδική διαχείριση για να μπορούν να χρησιμοποιηθούν σε αλγορίθμους μηχανικής μάθησης, οι οποίοι συνήθως δέχονται μόνο αριθμητικά δεδομένα. Η τεχνική του Multi-Hot Encoding επιτρέπει τη μετατροπή αυτών των λεξικών σε δυαδικές στήλες που υποδεικνύουν την παρουσία ή απουσία συγκεκριμένων κλειδιών.

Ο παρακάτω κώδικας αναλύει τη διαδικασία για την εξαγωγή μοναδικών κλειδιών από στήλες που περιέχουν τέτοια λεξικά και τη δημιουργία νέων στήλων που δείχνουν την παρουσία αυτών των κλειδιών.

Αναλυτική Περιγραφή Συναρτήσεων

- **extract_unique_keys:**
    - **Σκοπός**: Η συνάρτηση αυτή δέχεται ως είσοδο μια στήλη του DataFrame που περιέχει λεξικά σε μορφή string και εξάγει όλα τα μοναδικά κλειδιά που εμφανίζονται σε αυτά τα λεξικά.
    - **Περιγραφή**:
        - Χρησιμοποιεί τη ast.literal_eval για να μετατρέψει τη string αναπαράσταση του λεξικού σε πραγματικό λεξικό.
        - Συγκεντρώνει όλα τα μοναδικά κλειδιά που βρίσκονται στα λεξικά σε ένα σύνολο (set), ώστε να είναι εύκολα προσβάσιμα για περαιτέρω επεξεργασία.

- **check_key_presence**:
    - **Σκοπός**: Η συνάρτηση αυτή ελέγχει την παρουσία ενός συγκεκριμένου κλειδιού σε ένα λεξικό που παρέχεται σε μορφή string.
    - **Περιγραφή**:
        - Μετατρέπει τη string αναπαράσταση του λεξικού σε πραγματικό λεξικό.
        - Επιστρέφει 1 αν το συγκεκριμένο κλειδί υπάρχει στο λεξικό και 0 αν δεν υπάρχει, επιτρέποντας τη δημιουργία δυαδικών στηλών που αναπαριστούν την παρουσία αυτού του κλειδιού.
- **create_presence_columns:**
    - **Σκοπός**: Η συνάρτηση αυτή δημιουργεί νέες στήλες στο DataFrame για κάθε μοναδικό κλειδί που εξάγεται από τα λεξικά.
    - **Περιγραφή**:
        - Για κάθε μοναδικό κλειδί που έχει εξαχθεί, δημιουργεί μια νέα στήλη που δείχνει την παρουσία (με τιμή 1) ή την απουσία (με τιμή 0) του κλειδιού σε κάθε εγγραφή του αρχικού DataFrame.
        - Προσθέτει αυτές τις στήλες στο αρχικό DataFrame και επιστρέφει το ενημερωμένο DataFrame.

Συνοψίζοντας, με τον παραπάνω κώδικα, μετατρέπονται τα χαρακτηριστικά που περιέχουν πολλαπλά κλειδιά σε μορφή λεξικών, σε ένα σύνολο από δυαδικές στήλες που αναπαριστούν την παρουσία κάθε κλειδιού. Αυτή η προσέγγιση επιτρέπει στους αλγορίθμους μηχανικής μάθησης να αξιοποιήσουν τις πληροφορίες που κρύβονται μέσα στα λεξικά, κάνοντάς τις εύκολα διαχειρίσιμες και χρήσιμες στη διαδικασία μοντελοποίησης.

3.3.28 Προετοιμασία Δεδομένων για Εκπαίδευση Μοντέλου και Εφαρμογή

```
def prepare_data(X, Y, numeric_cols=numeric_cols, one_hot_cols=one_hot_cols,
multi_hot_cols=multi_hot_cols):
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
```

```
random_state=42)
    X_train1, X_test1, scaler1, X_train2, X_test2, scaler2 =
scale_data(X_train, X_test, numeric_cols)

    X_train1, encoder, encoded_cols = create_fit_transform(X_train1,
one_hot_cols)
    X_train2 = transform_data(X_train2, encoder, encoded_cols, one_hot_cols)
    X_test1 = transform_data(X_test1, encoder, encoded_cols, one_hot_cols)
    X_test2 = transform_data(X_test2, encoder, encoded_cols, one_hot_cols)

    unique_keys_type = extract_unique_keys(X_train1['trip_events_type'])
    X_train1 = create_presence_columns(X_train1, 'trip_events_type',
unique_keys_type)
    X_train2 = create_presence_columns(X_train2, 'trip_events_type',
unique_keys_type)
    X_test1 = create_presence_columns(X_test1, 'trip_events_type',
unique_keys_type)
    X_test2 = create_presence_columns(X_test2, 'trip_events_type',
unique_keys_type)

    unique_keys_weather =
extract_unique_keys(X_train1['Weather_Main_raw_events_weather'])
    X_train1 = create_presence_columns(X_train1,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_train2 = create_presence_columns(X_train2,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test1 = create_presence_columns(X_test1,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test2 = create_presence_columns(X_test2,
'Weather_Main_raw_events_weather', unique_keys_weather)

    unique_keys_raw_type = extract_unique_keys(X_train1['Raw_type_counts'])
    X_train1 = create_presence_columns(X_train1, 'Raw_type_counts',
unique_keys_raw_type)
    X_train2 = create_presence_columns(X_train2, 'Raw_type_counts',
unique_keys_raw_type)
    X_test1 = create_presence_columns(X_test1, 'Raw_type_counts',
unique_keys_raw_type)
    X_test2 = create_presence_columns(X_test2, 'Raw_type_counts',
unique_keys_raw_type)

    unique_keys_raw_accident_trigger =
extract_unique_keys(X_train1['Raw_accident_trigger_counts'])
    X_train1 = create_presence_columns(X_train1, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_train2 = create_presence_columns(X_train2, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_test1 = create_presence_columns(X_test1, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_test2 = create_presence_columns(X_test2, 'Raw_accident_trigger_counts',
```

```
unique_keys_raw_accident_trigger)

    X_train1.drop(multi_hot_cols, axis=1, inplace=True)
    X_train2.drop(multi_hot_cols, axis=1, inplace=True)
    X_test1.drop(multi_hot_cols, axis=1, inplace=True)
    X_test2.drop(multi_hot_cols, axis=1, inplace=True)

    return X_train1, X_train2, X_test1, X_test2, y_train, y_test, scaler1,
scaler2, encoder, encoded_cols, unique_keys_type, unique_keys_weather,
unique_keys_raw_type, unique_keys_raw_accident_trigger
```

```
(X_train1, X_train2, X_test1, X_test2, y_train, y_test, scaler1, scaler2,
encoder, encoded_cols, unique_keys_type, unique_keys_weather,
unique_keys_raw_type, unique_keys_raw_accident_trigger) = prepare_data(X, Y)
```

Η συνάρτηση prepare_data εκτελεί μια ολοκληρωμένη προετοιμασία δεδομένων πριν από την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Περιλαμβάνει διαχωρισμό των δεδομένων, κλιμάκωση των αριθμητικών χαρακτηριστικών, κωδικοποίηση κατηγορικών δεδομένων, και επεξεργασία multi-hot χαρακτηριστικών.

Αναλυτικά Βήματα της Συνάρτησης prepare_data:

- **Διαχωρισμός Δεδομένων σε Δεδομένα Εκπαίδευσης και Δοκιμής:**
    - Ο αρχικός διαχωρισμός των δεδομένων πραγματοποιείται με τη χρήση της train_test_split, διαχωρίζοντας τα δεδομένα σε δεδομένα εκπαίδευση (80%) και δεδομένα δοκιμής (20%).
    - Η παράμετρος random_state εξασφαλίζει την επαναληψιμότητα των αποτελεσμάτων.
- **Κλιμάκωση των Αριθμητικών Χαρακτηριστικών**:
    - **Min-Max Scaling**: Μετασχηματίζει τα αριθμητικά δεδομένα έτσι ώστε να βρίσκονται σε ένα συγκεκριμένο εύρος (0 και 1).
    - **Standard Scaling**: Προσαρμόζει τα αριθμητικά δεδομένα έτσι ώστε να έχουν μέση τιμή 0 και τυπική απόκλιση 1.
    - Τα κλιμακωμένα δεδομένα αποθηκεύονται στα X_train1, X_test1 για Min-Max Scaling και στα X_train2, X_test2 για Standard Scaling.

- **Κωδικοποίηση One-Hot για Κατηγορικά Χαρακτηριστικά**:
    - Η συνάρτηση create_fit_transform εφαρμόζει το One-Hot Encoding στα κατηγορικά χαρακτηριστικά του εκπαιδευτικού συνόλου, προσθέτοντας νέες δυαδικές στήλες που αντιπροσωπεύουν τις κατηγορίες.
    - Τα ίδια κατηγορικά χαρακτηριστικά στα δεδομένα δοκιμής κωδικοποιούνται με τον ίδιο encoder, εξασφαλίζοντας τη συνέπεια μεταξύ των δεδομένων εκπαίδευσης και δοκιμής.
- **Επεξεργασία των Multi-Hot Χαρακτηριστικών**:
    - Τα χαρακτηριστικά που περιέχουν λεξικά με πολλαπλά κλειδιά μετατρέπονται σε δυαδικές στήλες. Κάθε στήλη δείχνει την παρουσία (1) ή την απουσία (0) ενός συγκεκριμένου κλειδιού.
    - Αυτή η μετατροπή γίνεται μέσω των συναρτήσεων extract_unique_keys και create_presence_columns.
- **Αφαίρεση των Αρχικών Multi-Hot Στηλών**:
    - Αφού δημιουργηθούν οι νέες στήλες για τα κλειδιά, οι αρχικές στήλες που περιέχουν τα λεξικά διαγράφονται από τα σύνολα δεδομένων.
- **Επιστροφή των Επεξεργασμένων Δεδομένων**:
    - Η συνάρτηση επιστρέφει τα κλιμακωμένα και επεξεργασμένα δεδομένα εκπαίδευσης και δοκιμής, τους μετασχηματιστές (scalers) και τον encoder, καθώς και τα μοναδικά κλειδιά από τα multi-hot χαρακτηριστικά.

Εφαρμογή της Συνάρτησης prepare_data:

Με την εφαρμογή της συνάρτησης prepare_data, τα δεδομένα προετοιμάζονται πλήρως για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Η διαδικασία περιλαμβάνει την κλιμάκωση, την κωδικοποίηση, και την επεξεργασία των δεδομένων, παρέχοντας τα εξής σύνολα:

- **X_train1**, **X_test1:** Τα κλιμακωμένα με Min-Max Scaling δεδομένα εκπαίδευσης και δοκιμής.

- **X_train2**, **X_test2**: Τα κλιμακωμένα με Standard Scaling δεδομένα εκπαίδευσης και δοκιμής.
- **y_train**, **y_test**: Οι ετικέτες για την εκπαίδευση και τη δοκιμή του μοντέλου.
- **scaler1**, **scaler2**: Οι μετασχηματιστές που χρησιμοποιήθηκαν για την κλιμάκωση.
- **encoder**: Ο κωδικοποιητής One-Hot που εφαρμόστηκε στα κατηγορικά χαρακτηριστικά.
- **encoded_cols**: Τα ονόματα των νέων στηλών που δημιουργήθηκαν από την κωδικοποίηση One-Hot.
- **unique_keys_type**, **unique_keys_weather**, **unique_keys_raw_type**, **unique_keys_raw_accident_trigger**: Τα μοναδικά κλειδιά που εξήχθησαν από τα multi-hot χαρακτηριστικά.

Αυτή η συνάρτηση διασφαλίζει ότι τα δεδομένα είναι έτοιμα για εκπαίδευση, προετοιμάζοντας τα χαρακτηριστικά σε μορφή κατάλληλη για χρήση σε αλγορίθμους μηχανικής μάθησης.

3.3.29 Εκπαίδευση και Αξιολόγηση του Μοντέλου με XGBoost

```
def train_and_evaluate_model(X_train, y_train, n_estimators, max_depth,
learning_rate):
    model = XGBRegressor(
        n_estimators=n_estimators,
        max_depth=max_depth,
        learning_rate=learning_rate,
        subsample=0.7,
        colsample_bytree=0.7,
        n_jobs=-1,
        random_state=42
    )

    kf = KFold(n_splits=7, shuffle=True, random_state=42)
    cv_scores = []

    for train_idx, val_idx in kf.split(X_train):
        X_train_fold, X_val_fold = X_train.iloc[train_idx],
X_train.iloc[val_idx]
        y_train_fold, y_val_fold = y_train.iloc[train_idx],
y_train.iloc[val_idx]
        model.fit(X_train_fold, y_train_fold)
```

```
        y_pred = model.predict(X_val_fold)
        mse = mean_squared_error(y_val_fold, y_pred)
        cv_scores.append(mse)

    return np.mean(cv_scores)
```

Η συνάρτηση train_and_evaluate_model χρησιμοποιεί το XGBoost για την εκπαίδευση ενός μοντέλου πρόβλεψης, αξιοποιώντας παράλληλα τη μέθοδο K-Fold cross-validation για την αξιόπιστη αξιολόγηση της απόδοσης του μοντέλου.

Αναλυτικά Βήματα της Συνάρτησης train_and_evaluate_model:

- **Ορισμός του Μοντέλου XGBoost**:
    - Η συνάρτηση ξεκινά με τη δημιουργία ενός μοντέλου XGBRegressor, το οποίο είναι ρυθμισμένο με συγκεκριμένες υπερπαραμέτρους όπως:
        - **n_estimators**: Ο αριθμός των δέντρων στο μοντέλο.
        - **max_depth**: Το μέγιστο βάθος των δέντρων, που ελέγχει πόσο πολύπλοκα θα είναι.
        - **learning_rate**: Ο ρυθμός εκμάθησης, που καθορίζει πόσο μεγάλες θα είναι οι προσαρμογές στα βάρη κατά την εκπαίδευση.
        - **subsample**: Το ποσοστό των δεδομένων εκπαίδευσης που θα χρησιμοποιηθεί για κάθε δέντρο, βοηθώντας στην αποφυγή υπερπροσαρμογής.
        - **colsample_bytree**: Το ποσοστό των χαρακτηριστικών που θα χρησιμοποιηθούν για κάθε δέντρο, προσθέτοντας τυχαιότητα και σταθερότητα στο μοντέλο.
- **K-Fold Cross-Validation**:
    - Το K-Fold cross-validation χρησιμοποιείται για να διασφαλιστεί ότι το μοντέλο θα αξιολογηθεί με αξιοπιστία. Ορίζουμε 7 folds (n_splits=7), διαχωρίζοντας έτσι τα δεδομένα εκπαίδευσης σε 7 τμήματα.
    - Το μοντέλο εκπαιδεύεται σε 6 από αυτά τα τμήματα και δοκιμάζεται στο 7ο, επαναλαμβάνοντας αυτή τη διαδικασία για κάθε fold.

- Για κάθε fold, υπολογίζεται το μέσο τετραγωνικό σφάλμα (MSE) ως μέτρο της απόδοσης του μοντέλου.

- **Επιστροφή του Μέσου Όρου των Σφαλμάτων**:
  - Μετά την ολοκλήρωση του K-Fold cross-validation, η συνάρτηση υπολογίζει και επιστρέφει τον μέσο όρο των MSE από όλα τα folds, παρέχοντας μια εκτίμηση της γενικής απόδοσης του μοντέλου.

Συμπεράσματα για την Εκπαίδευση και Αξιολόγηση του Μοντέλου:

Η χρήση του train_and_evaluate_model παρέχει έναν αποτελεσματικό τρόπο εκπαίδευσης και αξιολόγησης του μοντέλου XGBoost. Η μέθοδος K-Fold cross-validation ενισχύει την αξιοπιστία των αποτελεσμάτων, μειώνοντας τον κίνδυνο υπερπροσαρμογής και εξασφαλίζοντας ότι το μοντέλο θα αποδώσει καλά και σε νέα, αόρατα δεδομένα.

Για την εύρεση των βέλτιστων υπερπαραμέτρων, η συνάρτηση θα εκτελεστεί πολλές φορές, δοκιμάζοντας διάφορους συνδυασμούς τιμών για τις παραμέτρους n_estimators, max_depth, και learning_rate. Η βέλτιστη επιλογή είναι αυτή που παράγει το χαμηλότερο μέσο MSE.

3.3.30 Βελτιστοποίηση Υπερπαραμέτρων με Optuna

```
def objective(trial):
    n_estimators = trial.suggest_categorical('n_estimators', [25, 50, 80, 100,
120, 150, 200, 250, 300, 350, 450, 500, 550, 600, 650, 700, 750])
    max_depth = trial.suggest_int('max_depth', 4, 20)
    learning_rate = trial.suggest_categorical('learning_rate', [0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 0.75, 0.99])

    mse = train_and_evaluate_model(X_train2, y_train, n_estimators, max_depth,
learning_rate)
    return mse

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
print("Best parameters found: ", study.best_params)
print("Best CV MSE found: ", study.best_value)
```

Η βελτιστοποίηση των υπερπαραμέτρων ενός μοντέλου είναι ένα κρίσιμο βήμα για τη βελτίωση της απόδοσής του. Στην περίπτωση αυτή, χρησιμοποιούμε τη βιβλιοθήκη Optuna για την αυτοματοποιημένη βελτιστοποίηση των υπερπαραμέτρων του μοντέλου

XGBoost. Η Optuna παρέχει έναν αποδοτικό και αυτοματοποιημένο τρόπο για την εύρεση των βέλτιστων τιμών των υπερπαραμέτρων μέσω της διαδικασίας του hyperparameter tuning.

Διαδικασία Βελτιστοποίησης με Optuna

Ο κώδικας που ακολουθεί ορίζει μια συνάρτηση objective, η οποία χρησιμοποιείται από την Optuna για τη βελτιστοποίηση των υπερπαραμέτρων του μοντέλου.

Ανάλυση του Κώδικα

- **Η Συνάρτηση objective:**
    - Η συνάρτηση objective καθορίζει πώς η Optuna θα αναζητήσει τις καλύτερες τιμές για τις υπερπαραμέτρους του μοντέλου.
    - Χρησιμοποιεί τη μέθοδο suggest_categorical για να προτείνει διαφορετικές τιμές για το n_estimators και το learning_rate.
    - Η μέθοδος suggest_int προτείνει τιμές για το max_depth μέσα σε ένα προκαθορισμένο εύρος.
    - Για κάθε συνδυασμό υπερπαραμέτρων, η συνάρτηση train_and_evaluate_model υπολογίζει το Mean Squared Error (MSE) χρησιμοποιώντας K-Fold Cross-Validation.
    - Η συνάρτηση επιστρέφει το MSE, το οποίο είναι το κριτήριο που προσπαθούμε να ελαχιστοποιήσουμε.
- **Δημιουργία και Βελτιστοποίηση της Μελέτης**:
    - Δημιουργείται μια “μελέτη” (study) με την Optuna, με στόχο την ελαχιστοποίηση του MSE.
    - Η συνάρτηση study.optimize εκτελείται για να δοκιμάσει 350 διαφορετικούς συνδυασμούς υπερπαραμέτρων, με βάση τη συνάρτηση objective. Αυτό γίνεται ορίζοντας n_trials=350.

- **Εκτύπωση των Βέλτιστων Υπερπαραμέτρων**:
    - Μετά την ολοκλήρωση της βελτιστοποίησης, εκτυπώνονται οι βέλτιστες υπερπαράμετροι που βρέθηκαν και το καλύτερο CV MSE που επιτεύχθηκε.

Συμπέρασμα

Με τη χρήση της Optuna για την αυτόματη βελτιστοποίηση των υπερπαραμέτρων, εξασφαλίζουμε ότι το μοντέλο μας επιτυγχάνει την καλύτερη δυνατή απόδοση. Η διαδικασία αυτή επιτρέπει την ανίχνευση του βέλτιστου συνδυασμού υπερπαραμέτρων με αποδοτικό και αυτοματοποιημένο τρόπο, χωρίς την ανάγκη για χειροκίνητη δοκιμή κάθε πιθανού συνδυασμού.

3.3.31 Αξιολόγηση του Μοντέλου με Δεδομένα Κλιμακωμένα με MinMaxScaler και StandardScaler στα Δεδομένα Επικύρωσης

```
model = XGBRegressor(n_estimators=700, max_depth=4, learning_rate=0.05,
               subsample=0.7, colsample_bytree=0.7, n_jobs=-1, random_state=42)

kf = KFold(n_splits=7, shuffle=True, random_state=42)
cv_scores_mse = []
cv_scores_rmse = []
cv_scores_mae = []
cv_scores_mape = []

for train_idx, val_idx in kf.split(X_train1):
    X_train_fold, X_val_fold = X_train1.iloc[train_idx], X_train1.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]
    model.fit(X_train_fold, y_train_fold)
    y_pred = model.predict(X_val_fold)

    mse = mean_squared_error(y_val_fold, y_pred)
    rmse = mean_squared_error(y_val_fold, y_pred, squared=False)
    mae = mean_absolute_error(y_val_fold, y_pred)
    mape = mean_absolute_percentage_error(y_val_fold, y_pred) * 100

    cv_scores_mse.append(mse)
    cv_scores_rmse.append(rmse)
    cv_scores_mae.append(mae)
    cv_scores_mape.append(mape)
    print(f"Validation MSE: {mse}")
    print(f"Validation RMSE: {rmse}")
    print(f"Validation Mean Absolute Error: {mae}")
    print(f"Validation Mean Absolute Percentage Error: {mape} %")
    print()
```

```
print(f"\n\nMean Validation MSE: {np.mean(cv_scores_mse)}")
print(f"Mean Validation RMSE: {np.mean(cv_scores_rmse)}")
print(f"Mean Validation MAE: {np.mean(cv_scores_mae)}")
print(f"Mean Validation MAPE: {np.mean(cv_scores_mape)} %")
```

```
model = XGBRegressor(n_estimators=700, max_depth=4, learning_rate=0.05,
                subsample=0.7, colsample_bytree=0.7, n_jobs=-1, random_state=42)

kf = KFold(n_splits=7, shuffle=True, random_state=42)
cv_scores_mse = []
cv_scores_rmse = []
cv_scores_mae = []
cv_scores_mape = []

for train_idx, val_idx in kf.split(X_train2):
    X_train_fold, X_val_fold = X_train2.iloc[train_idx], X_train2.iloc[val_idx]
    y_train_fold, y_val_fold = y_train.iloc[train_idx], y_train.iloc[val_idx]
    model.fit(X_train_fold, y_train_fold)
    y_pred = model.predict(X_val_fold)
    mse = mean_squared_error(y_val_fold, y_pred)
    rmse = mean_squared_error(y_val_fold, y_pred, squared=False)
    mae = mean_absolute_error(y_val_fold, y_pred)
    mape = mean_absolute_percentage_error(y_val_fold, y_pred) * 100

    cv_scores_mse.append(mse)
    cv_scores_rmse.append(rmse)
    cv_scores_mae.append(mae)
    cv_scores_mape.append(mape)

    print(f"Validation MSE: {mse}")
    print(f"Validation RMSE: {rmse}")
    print(f"Validation Mean Absolute Error: {mae}")
    print(f"Validation Mean Absolute Percentage Error: {mape} %")
    print()

print(f"\n\nMean Validation MSE: {np.mean(cv_scores_mse)}")
print(f"Mean Validation RMSE: {np.mean(cv_scores_rmse)}")
print(f"Mean Validation MAE: {np.mean(cv_scores_mae)}")
print(f"Mean Validation MAPE: {np.mean(cv_scores_mape)} %")
```

Αφού βελτιστοποιήσαμε τις υπερπαραμέτρους του μοντέλου χρησιμοποιώντας την Optuna, το επόμενο βήμα είναι η τελική εκπαίδευση και αξιολόγηση του μοντέλου XGBoost. Στην πρώτη περίπτωση, χρησιμοποιούμε τα δεδομένα που έχουν κλιμακωθεί με τον MinMaxScaler και αξιολογούμε την απόδοση του μοντέλου στα δεδομένα επικύρωσης που προκύπτουν από τη μέθοδο K-Fold Cross-Validation.

Ανάλυση του Κώδικα

- **Δημιουργία του Μοντέλου:**
    - Το μοντέλο XGBoost Regressor (XGBRegressor) δημιουργείται με τις υπερπαραμέτρους που βρέθηκαν ως βέλτιστες:
        - n_estimators=700
        - max_depth=4
        - learning_rate=0.05.
    - Οι παράμετροι subsample και colsample_bytree έχουν ρυθμιστεί στο 0.7 για να μειωθεί ο κίνδυνος υπερεκπαίδευσης.
- **K-Fold Cross-Validation και Αξιολόγηση στα Δεδομένα Επικύρωσης:**
    - Χρησιμοποιείται η μέθοδος KFold για να δημιουργηθούν 7 διαφορετικές πτυχές (folds) των δεδομένων εκπαίδευσης. Σε κάθε πτυχή, το μοντέλο εκπαιδεύεται σε ένα τμήμα των δεδομένων εκπαίδευσης και αξιολογείται στα υπόλοιπα δεδομένα, που χρησιμεύουν ως δεδομένα επικύρωσης.
    - Για κάθε πτυχή, υπολογίζονται διάφορες μετρικές απόδοσης στα δεδομένα επικύρωσης.
- Εκτύπωση των Αποτελεσμάτων:
    - Οι μετρικές απόδοσης εκτυπώνονται για κάθε πτυχή, με έμφαση στα δεδομένα επικύρωσης.
    - Στο τέλος, εκτυπώνεται ο μέσος όρος κάθε μετρικής για όλες τις πτυχές, δίνοντας μια συνολική εικόνα της απόδοσης του μοντέλου στα δεδομένα επικύρωσης.

Η ίδια διαδικασία αξιολόγησης εφαρμόζεται και στα δεδομένα που έχουν κλιμακωθεί χρησιμοποιώντας τον StandardScaler. Όπως και στην προηγούμενη περίπτωση, το μοντέλο εκπαιδεύεται σε τμήματα των δεδομένων εκπαίδευσης και αξιολογείται στα δεδομένα επικύρωσης που προκύπτουν από το K-Fold Cross-Validation.

Σκοπός της Δοκιμής

Ο στόχος είναι να συγκρίνουμε την απόδοση του μοντέλου στα δεδομένα επικύρωσης όταν τα δεδομένα έχουν κλιμακωθεί με δύο διαφορετικές μεθόδους (MinMaxScaler, StandardScaler). Με αυτή τη διαδικασία μπορούμε να επιλέξουμε την καταλληλότερη μέθοδο κλιμάκωσης για το συγκεκριμένο πρόβλημα, βάσει της απόδοσης του μοντέλου στα δεδομένα επικύρωσης.

Μετρικές Απόδοσης

Για την αξιολόγηση της απόδοσης του μοντέλου, χρησιμοποιήθηκαν οι ακόλουθες βασικές μετρικές:

- **Mean Squared Error (MSE)**
    - **Ορισμός**: Το Mean Squared Error (MSE) είναι η μέση τιμή των τετραγωνικών διαφορών μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Είναι ένας κοινός δείκτης για την αποτίμηση της ακρίβειας των μοντέλων πρόβλεψης.
    - **Τυπος:**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Εικ.4: Εξίσωση για τον υπολογισμό του Mean Squared Error (MSE)

- **Root Mean Squared Error (RMSE)**
    - **Ορισμός**: Το Root Mean Squared Error (RMSE) είναι η τετραγωνική ρίζα του MSE. Αποτελεί επίσης έναν δείκτη ακρίβειας, και μετράει την τυπική απόκλιση των προβλέψεων του μοντέλου από τις πραγματικές τιμές.

- **Τύπος**:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Εικ.5: Εξίσωση για τον υπολογισμό του Root Mean Squared Error (RMSE)

Το RMSE είναι πιο ερμηνεύσιμο καθώς έχει τις ίδιες μονάδες με τις πραγματικές τιμές.

- **MAE (Mean Absolute Error)**
  - **Ορισμός**: Το Mean Absolute Error (MAE) είναι η μέση τιμή των απόλυτων διαφορών μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Η MAE μετράει το μέσο μέγεθος των σφαλμάτων σε ένα σύνολο προβλέψεων, χωρίς να λαμβάνει υπόψη το πρόσημο του σφάλματος.
  - **Τύπος**:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Εικ.6: Εξίσωση για τον υπολογισμό του Mean Absolute Error (MAE)

Η MAE δίνει μια άμεση εικόνα για το μέσο απόλυτο σφάλμα σε μονάδες των πραγματικών τιμών.

- **MAPE (Mean Absolute Percentage Error)**
  - **Ορισμός**: Το Mean Absolute Percentage Error (MAPE) είναι η μέση ποσοστιαία διαφορά μεταξύ των προβλεπόμενων και των πραγματικών τιμών. Είναι ένα μέτρο που εκφράζει το σφάλμα ως ποσοστό των πραγματικών τιμών.

- **Τύπος**:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{y_i} - y_i}{y_i}\right| \times 100$$

Εικ.7: Εξίσωση για τον υπολογισμό του Mean Absolute Percentage Error (MAPE)

Το MAPE είναι χρήσιμο για να κατανοηθεί το μέγεθος του σφάλματος σε σχέση με το μέγεθος της πραγματικής τιμής και εκφράζεται ως ποσοστό.

Αυτές οι μετρικές επιλέχθηκαν για να προσφέρουν μια σφαιρική εικόνα της ακρίβειας και της σταθερότητας του μοντέλου, επιτρέποντας την εκτίμηση της απόδοσής του τόσο σε απόλυτους αριθμούς όσο και σε ποσοστιαίες διαφορές.

Συμπεράσματα

Η σύγκριση των μετρικών απόδοσης (MSE, RMSE, MAE, MAPE) στα δεδομένα επικύρωσης, μεταξύ των δύο μεθόδων κλιμάκωσης, θα μας επιτρέψει να επιλέξουμε την πιο αποτελεσματική για το συγκεκριμένο πρόβλημα.

3.3.32 Αξιολόγηση του Μοντέλου με Δεδομένα Κλιμακωμένα με MinMaxScaler και StandardScaler στα Δεδομένα Δοκιμής

```
def print_evaluation_metrics(y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred) * 100

    print(f"Test MSE: {mse}")
    print(f"Test RMSE: {rmse}")
    print(f"Test Mean Absolute Error: {mae}")
    print(f"Test Mean Absolute Percentage Error: {mape} %")

    return {
        "mse": mse,
        "rmse": rmse,
        "mae": mae,
        "mape": mape
    }
```

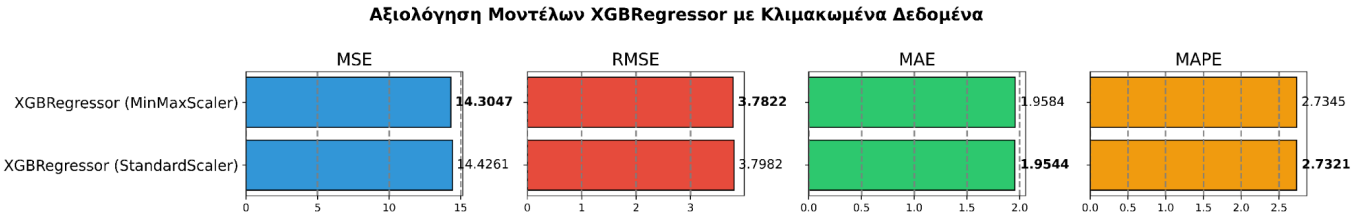
```
final_model1 = XGBRegressor(n_estimators=700, max_depth=4, learning_rate=0.05,
               subsample=0.7, colsample_bytree=0.7, n_jobs=-1, random_state=42)
final_model1.fit(X_train1, y_train)
y_pred1 = final_model1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, y_pred1)
```

```
final_model2 = XGBRegressor(n_estimators=700, max_depth=4, learning_rate=0.05,
               subsample=0.7, colsample_bytree=0.7, n_jobs=-1, random_state=42)
final_model2.fit(X_train2, y_train)
y_pred2 = final_model2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, y_pred2)
```

Αξιολόγηση Μοντέλων XGBRegressor με Κλιμακωμένα Δεδομένα

Model (Scaler)	MSE	RMSE	MAE	MAPE
XGBRegressor (MinMaxScaler)	14.3047	3.7822	1.9584	2.7345
XGBRegressor (StandardScaler)	14.4261	3.7982	1.9544	2.7321

Πίν.1: Αποτελέσματα Αξιολόγησης του XGBRegressor με Διαφορετικές Τεχνικές Κλιμάκωσης



Σχ.1: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης του XGBRegressor για Διαφορετικές Τεχνικές Κλιμάκωσης

Μετά την ολοκλήρωση της εκπαίδευσης και της αρχικής αξιολόγησης του μοντέλου μέσω K-Fold cross-validation, το επόμενο βήμα είναι η τελική αξιολόγηση του μοντέλου χρησιμοποιώντας τα δεδομένα δοκιμής. Αυτή η διαδικασία αξιολογεί την ικανότητα του μοντέλου να γενικεύει σε νέα δεδομένα που δεν έχουν χρησιμοποιηθεί κατά την εκπαίδευση.

Εκπαίδευση του Τελικού Μοντέλου με MinMaxScaler

Το μοντέλο XGBoost Regressor εκπαιδεύεται χρησιμοποιώντας τα δεδομένα εκπαίδευσης που έχουν κλιμακωθεί με τη μέθοδο MinMaxScaler. Στη συνέχεια, το μοντέλο προβλέπει τις

τιμές για το σύνολο δεδομένων δοκιμής και οι προβλέψεις αυτές συγκρίνονται με τις πραγματικές τιμές χρησιμοποιώντας τις μετρικές MSE, RMSE, MAE και MAPE. Αυτές οι μετρικές προσφέρουν μια σαφή εικόνα της απόδοσης του μοντέλου στα δεδομένα δοκιμής και δείχνουν κατά πόσο το μοντέλο μπορεί να προβλέψει αξιόπιστα σε νέα δεδομένα.

Εκπαίδευση του Τελικού Μοντέλου με StandardScaler

Στη συνέχεια, η ίδια διαδικασία επαναλαμβάνεται με τα δεδομένα που έχουν κλιμακωθεί χρησιμοποιώντας τη μέθοδο StandardScaler. Το μοντέλο εκπαιδεύεται με αυτά τα δεδομένα και ακολουθεί η αξιολόγηση των προβλέψεων στα δεδομένα δοκιμής, χρησιμοποιώντας τις ίδιες μετρικές.

Σύγκριση των Μεθόδων Κλιμάκωσης

Η σύγκριση των αποτελεσμάτων από τις δύο μεθόδους κλιμάκωσης (MinMaxScaler και StandardScaler) επιτρέπει τον εντοπισμό της μεθόδου που αποδίδει καλύτερα στο συγκεκριμένο πρόβλημα. Εάν οι μετρικές υποδεικνύουν χαμηλότερα σφάλματα για μία μέθοδο σε σχέση με την άλλη, αυτό μπορεί να καθορίσει ποια μέθοδος κλιμάκωσης θα χρησιμοποιηθεί για το τελικό μοντέλο.

Τελικά Συμπεράσματα

Τα αποτελέσματα αυτής της τελικής αξιολόγησης θα βοηθήσουν στην κατανόηση της αποτελεσματικότητας του μοντέλου σε νέα δεδομένα και στην επιλογή της καλύτερης στρατηγικής κλιμάκωσης. Ανάλογα με το ποια μέθοδος κλιμάκωσης δίνει τα καλύτερα αποτελέσματα, θα αποφασιστεί ποια θα χρησιμοποιηθεί στη διαδικασία ανάπτυξης και τελικής αξιοποίησης του μοντέλου.

3.3.33 Βοηθητικές Συναρτήσεις Νευρωνικών Δικτύων

```
def plot_metrics(history, title):
    plt.figure(figsize=(8, 6))
    plt.plot(history.history['mse'], marker='o', label='Train MSE')
    plt.plot(history.history['val_mse'], marker='o', label='Validation MSE')
    plt.title(title)
    plt.xlabel('Epoch')
    plt.ylabel('MSE')
    plt.legend(loc='upper right')
```

```
    plt.grid(True)
    plt.ylim(0, 150)
    plt.show()


def evaluate_model(model, X_test, y_test):
    score = model.evaluate(X_test, y_test, verbose=0)
    loss, mse, mae, mape = score[0], score[1], score[2], score[3]
    rmse = np.sqrt(mse)

    print("Test loss:", loss)
    print("Test Mean Squared Error:", mse)
    print("Test Root Mean Squared Error:", rmse)
    print("Test Mean Absolute Error:", mae)
    print(f"Test Mean Absolute Percentage Error: {mape} %")

    return mse, mae, mape
```

Στη συνέχεια, περιγράφονται δύο βασικές συναρτήσεις που διευκολύνουν την εκπαίδευση και αξιολόγηση ενός νευρωνικού δικτύου για προβλέψεις. Αυτές οι συναρτήσεις βοηθούν στην παρακολούθηση της απόδοσης του μοντέλου και στην αξιολόγηση των προβλέψεων του στα δεδομένα δοκιμής (test set).

Συνάρτηση plot_metrics

Η συνάρτηση plot_metrics είναι υπεύθυνη για την οπτικοποίηση της απόδοσης του νευρωνικού δικτύου κατά τη διάρκεια της εκπαίδευσης.

- **Είσοδος:**
    - **history**: Το αντικείμενο ιστορικού εκπαίδευσης, το οποίο περιέχει τις τιμές των μετρικών εκπαίδευσης και επικύρωσης για κάθε εποχή (epoch).
    - **title**: Ο τίτλος που θα εμφανιστεί στο γράφημα.
- **Λειτουργία:**
    - Η συνάρτηση δημιουργεί διάγραμμα γραμμών που δείχνει τη μεταβολή του MSE (Mean Squared Error) κατά τη διάρκεια της εκπαίδευσης για τα δεδομένα εκπαίδευσης και επικύρωσης.

- Το διάγραμμα επιτρέπει την παρακολούθηση της σύγκλισης του μοντέλου και την αξιολόγηση του εάν το μοντέλο υπερπροσαρμόζεται (overfitting) ή υποπροσαρμόζεται (underfitting).

Συνάρτηση evaluate_model

Η συνάρτηση evaluate_model χρησιμοποιείται για την αξιολόγηση της απόδοσης του νευρωνικού δικτύου πάνω στα δεδομένα δοκιμής (test set).

- **Είσοδος:**
  - **model**: Το εκπαιδευμένο νευρωνικό δίκτυο.
  - **X_test**: Τα χαρακτηριστικά του test set.
  - **y_test**: Οι πραγματικές τιμές (labels) του test set.
- **Λειτουργία:**
  - Η συνάρτηση αξιολογεί το μοντέλο στα δεδομένα δοκιμής και υπολογίζει τις μετρικές MSE, RMSE, MAE και MAPE. Οι μετρικές αυτές επιστρέφονται ως λεξικό για περαιτέρω ανάλυση και εμφανίζονται στην οθόνη.

Συμπεράσματα

Οι παραπάνω συναρτήσεις plot_metrics και evaluate_model είναι κρίσιμες για την ανάλυση της απόδοσης ενός νευρωνικού δικτύου. Η οπτικοποίηση της απόδοσης κατά τη διάρκεια της εκπαίδευσης και η τελική αξιολόγηση στα δεδομένα δοκιμής παρέχουν σημαντικές πληροφορίες για την ποιότητα του μοντέλου και την ικανότητά του να γενικεύει σε νέα δεδομένα.

3.3.34 Εκπαίδευση και Αξιολόγηση Νευρωνικού Δικτύου με Χρήση του Βελτιστοποιητή Adam

```
def train_evaluate_model_nn(X_train, y_train, X_test, y_test, num_nodes,
dropout_prob, lr, batch_size=32, epochs=120):
    nn_model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
```

```
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(1)
    ])


    nn_model.compile(optimizer=tf.keras.optimizers.Adam(lr),
loss='mean_squared_error', metrics=['mse', 'mae', 'mape'])
    early_stopping = EarlyStopping(monitor='val_mse', patience=10, mode='min',
restore_best_weights=True)

    history = nn_model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_split=0.2, callbacks=[early_stopping],
verbose=0)
    plot_metrics(history, 'Εξέλιξη του MSE κατά την Εκπαίδευση του Μοντέλου')
    mse, mae, mape = evaluate_model(nn_model, X_test, y_test)
    print()
    print()

    return nn_model, mse, mae, mape, history
```

```
least_val_loss = float('inf')
best_hyperparameters = {}
best_model = None

for num_nodes in [32, 64, 128, 256, 512, 1024]:
    for dropout_prob in [0.0, 0.1, 0.2, 0.3]:
        for lr in [0.001, 0.005, 0.01, 0.05, 0.1, 0.25]:
            print(f"Num nodes: {num_nodes}, Dropout: {dropout_prob}, Learning
rate: {lr}")

            nn_model, mse, mae, mape, _ = train_evaluate_model_nn(X_train1,
y_train, X_test1, y_test, num_nodes, dropout_prob, lr)

            if mae < least_val_loss:
                least_val_loss = mae
                best_hyperparameters = {
                    "num_nodes": num_nodes,
                    "dropout_prob": dropout_prob,
                    "lr": lr
                }
                best_model = nn_model

print("Best hyperparameters:", best_hyperparameters)
best_model.save("best_model1.keras")
```

```
best_model = tf.keras.models.load_model("best_model1.keras")
mse, mae, mape = evaluate_model(best_model, X_test1, y_test)
```

Διευκρινίσεις:

Για να διασφαλιστεί η καλύτερη απόδοση του νευρωνικού δικτύου, πραγματοποιήθηκαν δύο διαφορετικές τεχνικές κανονικοποίησης δεδομένων, MinMaxScaler και StandardScaler. Παρόλο που δοκιμάστηκαν και οι δύο τεχνικές, η τελική επιλογή έγινε με βάση την απόδοση του μοντέλου. Ο MinMaxScaler απέδωσε καλύτερα αποτελέσματα σε σχέση με τον StandardScaler, όπως φάνηκε από τις μετρικές αξιολόγησης. Εξαιτίας του μεγάλου μεγέθους του τελικού αρχείου, διατηρήθηκε μόνο το μοντέλο που χρησιμοποιεί τον MinMaxScaler, ο οποίος παρείχε τη βέλτιστη απόδοση, ενώ το μοντέλο με τον StandardScaler αφαιρέθηκε για εξοικονόμηση χώρου.

Περιγραφη:

Ο παραπάνω κώδικας περιγράφει την εκπαίδευση και αξιολόγηση ενός νευρωνικού δικτύου, χρησιμοποιώντας τον βελτιστοποιητή Adam. Το δίκτυο εξετάζει διαφορετικούς συνδυασμούς υπερπαραμέτρων, όπως ο αριθμός νευρώνων (nodes), η πιθανότητα dropout, και ο ρυθμός μάθησης (learning rate), για να βρει τους βέλτιστους συνδυασμούς που θα αποδώσουν τα καλύτερα αποτελέσματα.

Συνάρτησης train_evaluate_model_nn

Η συνάρτηση train_evaluate_model_nn εκπαιδεύει και αξιολογεί το νευρωνικό δίκτυο χρησιμοποιώντας τις συγκεκριμένες υπερπαραμέτρους που παρέχονται ως είσοδος:

- **Δομή του Νευρωνικού Δικτύου:**
    - Το δίκτυο αποτελείται από τρία κρυφά στρώματα (Dense) με ενεργοποίηση ReLU και εφαρμόζεται Dropout για την αποφυγή υπερπροσαρμογής.
    - Η έξοδος είναι ένας νευρώνας χωρίς ενεργοποίηση, κατάλληλος για προβλήματα παλινδρόμησης.
- **Βελτιστοποιητής:**
    - Ο Adam χρησιμοποιείται ως ο βελτιστοποιητής, γνωστός για τη σταθερή και αποδοτική εκπαίδευση νευρωνικών δικτύων.

- **Early Stopping:**
    - Η εκπαίδευση διακόπτεται αυτόματα αν η απόδοση δεν βελτιώνεται για 10 συνεχόμενες εποχές, αποτρέποντας την υπερπροσαρμογή.
- **Αξιολόγηση:**
    - Μετά την εκπαίδευση, το μοντέλο αξιολογείται χρησιμοποιώντας το test set και υπολογίζονται μετρικές απόδοσης.

Διερεύνηση Υπερπαραμέτρων

Για να βρεθούν οι βέλτιστοι συνδυασμοί υπερπαραμέτρων, εκτελείται ένας πειραματισμός που περιλαμβάνει διάφορες τιμές για τον αριθμό νευρώνων, την πιθανότητα dropout, και τον ρυθμό μάθησης. Ο αριθμός νευρώνων (num_nodes), η πιθανότητα dropout (dropout_prob), και ο ρυθμός μάθησης (lr) δοκιμάζονται σε πολλαπλούς συνδυασμούς για να βρεθεί ο βέλτιστος που ελαχιστοποιεί το Mean Absolute Error (MAE). Το καλύτερο μοντέλο, σύμφωνα με τις μετρικές απόδοσης, αποθηκεύεται σε αρχείο για μελλοντική χρήση.

Αξιολόγηση του Βέλτιστου Μοντέλου

Αφού εκπαιδευτεί το βέλτιστο μοντέλο, αποθηκεύεται και αξιολογείται εκ νέου στο test set για να επιβεβαιωθεί η απόδοσή του. Τα αποτελέσματα της αξιολόγησης περιλαμβάνουν το MSE, RMSE, MAE, και MAPE, προσφέροντας μια συνολική εικόνα της απόδοσης του μοντέλου στα δεδομένα δοκιμής.

Συμπεράσματα

Αυτός ο κώδικας προσφέρει μια συστηματική προσέγγιση για την εκπαίδευση νευρωνικών δικτύων και την ανάλυση των επιδόσεών τους. Η χρήση του βελτιστοποιητή Adam σε συνδυασμό με τεχνικές όπως το Early Stopping και το Dropout εξασφαλίζει την αποφυγή υπερπροσαρμογής και επιτρέπει τη σταθερή και αποτελεσματική εκπαίδευση του μοντέλου. Με την αποθήκευση του καλύτερου μοντέλου, η προσέγγιση αυτή επιτρέπει την εύκολη επαναξιολόγηση και σύγκριση με άλλες τεχνικές ή μοντέλα.

3.3.35 Εκπαίδευση και Αξιολόγηση Νευρωνικού Δικτύου με Χρήση του Βελτιστοποιητή AdamW

```
def train_evaluate_model_nn(X_train, y_train, X_test, y_test, num_nodes,
dropout_prob, lr, batch_size=32, epochs=120):
    nn_model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(1)
    ])

    nn_model.compile(optimizer=tf.keras.optimizers.AdamW(learning_rate=lr,
weight_decay=1e-5), loss='mean_squared_error', metrics=['mse', 'mae', 'mape'])
    early_stopping = EarlyStopping(monitor='val_mse', patience=10, mode='min',
restore_best_weights=True)

    history = nn_model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_split=0.2, callbacks=[early_stopping],
verbose=0)
    plot_metrics(history, 'Εξέλιξη του MSE κατά την Εκπαίδευση του Μοντέλου')
    mse, mae, mape = evaluate_model(nn_model, X_test, y_test)
    print()
    print()

    return nn_model, mse, mae, mape, history
```

```
least_val_loss = float('inf')
best_hyperparameters = {}
best_model = None

for num_nodes in [128, 256, 512, 1024]:
    for dropout_prob in [0.0, 0.1, 0.2, 0.3]:
        for lr in [0.001, 0.005, 0.01, 0.05, 0.1, 0.25]:
            print(f"Num nodes: {num_nodes}, Dropout: {dropout_prob}, Learning
rate: {lr}")

            nn_model, mse, mae, mape, _ = train_evaluate_model_nn(X_train1,
y_train, X_test1, y_test, num_nodes, dropout_prob, lr)

            if mae < least_val_loss:
                least_val_loss = mae
                best_hyperparameters = {
                    "num_nodes": num_nodes,
                    "dropout_prob": dropout_prob,
```

```
                    "lr": lr
                }
                best_model = nn_model

print("Best hyperparameters:", best_hyperparameters)
best_model.save("best_model3.keras")
```

```
best_model = tf.keras.models.load_model("best_model3.keras")
mse, mae, mape = evaluate_model(best_model, X_test1, y_test)
```

Διευκρινίσεις:

Για να διασφαλιστεί η καλύτερη απόδοση του νευρωνικού δικτύου, πραγματοποιήθηκαν δύο διαφορετικές τεχνικές κανονικοποίησης δεδομένων, MinMaxScaler και StandardScaler. Παρόλο που δοκιμάστηκαν και οι δύο τεχνικές, η τελική επιλογή έγινε με βάση την απόδοση του μοντέλου. Ο MinMaxScaler απέδωσε καλύτερα αποτελέσματα σε σχέση με τον StandardScaler, όπως φάνηκε από τις μετρικές αξιολόγησης. Εξαιτίας του μεγάλου μεγέθους του τελικού αρχείου, διατηρήθηκε μόνο το μοντέλο που χρησιμοποιεί τον MinMaxScaler, ο οποίος παρείχε τη βέλτιστη απόδοση, ενώ το μοντέλο με τον StandardScaler αφαιρέθηκε για εξοικονόμηση χώρου.

Περιγραφη:

ο παραπάνω κώδικας περιγράφει την εκπαίδευση και αξιολόγηση ενός νευρωνικού δικτύου χρησιμοποιώντας τον βελτιστοποιητή AdamW (Adam with Weight Decay). Η μέθοδος αυτή συνδυάζει τα πλεονεκτήματα του Adam με την προσθήκη ενός όρου “weight decay”, που βοηθά στη βελτίωση της γενίκευσης του μοντέλου και στην αποφυγή υπερπροσαρμογής.

Συνάρτησης train_evaluate_model_nn

Η συνάρτηση train_evaluate_model_nn εκπαιδεύει και αξιολογεί το νευρωνικό δίκτυο χρησιμοποιώντας τις συγκεκριμένες υπερπαραμέτρους που παρέχονται ως είσοδος:

- **Δομή του Νευρωνικού Δικτύου:**
    - Το δίκτυο αποτελείται από τρία κρυφά στρώματα (Dense) με ενεργοποίηση ReLU και εφαρμόζεται Dropout για την αποφυγή υπερπροσαρμογής.

- Η έξοδος είναι ένας νευρώνας χωρίς ενεργοποίηση, κατάλληλος για προβλήματα παλινδρόμησης.

- **Βελτιστοποιητής:**
  - Ο AdamW χρησιμοποιείται για την εκπαίδευση, ο οποίος προσαρμόζει τον ρυθμό μάθησης με βάση τις τιμές των παραμέτρων, ενώ ταυτόχρονα εφαρμόζει weight decay για καλύτερη γενίκευση.

- **Early Stopping:**
  - Η εκπαίδευση διακόπτεται αυτόματα αν η απόδοση δεν βελτιώνεται για 10 συνεχόμενες εποχές, εξασφαλίζοντας ότι το μοντέλο δεν υπερπροσαρμόζεται.

- **Αξιολόγηση:**
  - Μετά την εκπαίδευση, το μοντέλο αξιολογείται χρησιμοποιώντας το test set και υπολογίζονται μετρικές απόδοσης.

Διερεύνηση Υπερπαραμέτρων

Για να βρεθούν οι βέλτιστοι συνδυασμοί υπερπαραμέτρων, εκτελείται ένας πειραματισμός που περιλαμβάνει διάφορες τιμές για τον αριθμό νευρώνων, την πιθανότητα dropout, και τον ρυθμό μάθησης. Ο αριθμός νευρώνων (num_nodes), η πιθανότητα dropout (dropout_prob), και ο ρυθμός μάθησης (lr) δοκιμάζονται σε πολλαπλούς συνδυασμούς για να βρεθεί ο βέλτιστος που ελαχιστοποιεί το Mean Absolute Error (MAE). Το καλύτερο μοντέλο, σύμφωνα με τις μετρικές απόδοσης, αποθηκεύεται σε αρχείο για μελλοντική χρήση.

Αξιολόγηση του Βέλτιστου Μοντέλου

Αφού εκπαιδευτεί το βέλτιστο μοντέλο, αποθηκεύεται και αξιολογείται εκ νέου στο test set για να επιβεβαιωθεί η απόδοσή του. Τα αποτελέσματα της αξιολόγησης περιλαμβάνουν το MSE, RMSE, MAE, και MAPE, προσφέροντας μια συνολική εικόνα της απόδοσης του μοντέλου στα δεδομένα δοκιμής.

Συμπεράσματα

Η προσέγγιση αυτή με τον βελτιστοποιητή AdamW προσφέρει μια ισχυρή μέθοδο εκπαίδευσης νευρωνικών δικτύων, ειδικά όταν επιθυμείται η αποφυγή υπερπροσαρμογής. Ο πειραματισμός με τις υπερπαραμέτρους επιτρέπει την εύρεση του βέλτιστου συνδυασμού, ενώ η χρήση του Early Stopping εξασφαλίζει ότι το μοντέλο δεν εκπαιδεύεται υπερβολικά. Με την αποθήκευση του καλύτερου μοντέλου, διασφαλίζεται ότι το τελικό αποτέλεσμα μπορεί να αξιοποιηθεί άμεσα ή να συγκριθεί με άλλα μοντέλα για την επιλογή της βέλτιστης λύσης.

3.3.36 Εκπαίδευση και Αξιολόγηση Νευρωνικού Δικτύου με Χρήση του Βελτιστοποιητή Nadam

```
def train_evaluate_model_nn(X_train, y_train, X_test, y_test, num_nodes,
dropout_prob, lr, batch_size=32, epochs=120):
    nn_model = tf.keras.Sequential([
        tf.keras.layers.Input(shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(num_nodes, activation='relu'),
        tf.keras.layers.Dropout(dropout_prob),
        tf.keras.layers.Dense(1)
    ])

    nn_model.compile(optimizer=tf.keras.optimizers.Nadam(learning_rate=lr),
loss='mean_squared_error', metrics=['mse', 'mae', 'mape'])
    early_stopping = EarlyStopping(monitor='val_mse', patience=10, mode='min',
restore_best_weights=True)

    history = nn_model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_split=0.2, callbacks=[early_stopping],
verbose=0)
    plot_metrics(history, 'Εξέλιξη του MSE κατά την Εκπαίδευση του Μοντέλου')
    mse, mae, mape = evaluate_model(nn_model, X_test, y_test)
    print()
    print()

    return nn_model, mse, mae, mape, history
```

```
least_val_loss = float('inf')
best_hyperparameters = {}
best_model = None
```

```
for num_nodes in [128, 256, 512, 1024]:
    for dropout_prob in [0.0, 0.1, 0.2, 0.3]:
        for lr in [0.001, 0.005, 0.01, 0.05, 0.1, 0.25]:
            print(f"Num nodes: {num_nodes}, Dropout: {dropout_prob}, Learning
rate: {lr}")

            nn_model, mse, mae, mape, _ = train_evaluate_model_nn(X_train1,
y_train, X_test1, y_test, num_nodes, dropout_prob, lr)

            if mae < least_val_loss:
                least_val_loss = mae
                best_hyperparameters = {
                    "num_nodes": num_nodes,
                    "dropout_prob": dropout_prob,
                    "lr": lr
                }
                best_model = nn_model

print("Best hyperparameters:", best_hyperparameters)
best_model.save("best_model7.keras")
```

```
best_model = tf.keras.models.load_model("best_model7.keras")
mse, mae, mape = evaluate_model(best_model, X_test1, y_test)
```

Διευκρινίσεις:

Για να διασφαλιστεί η καλύτερη απόδοση του νευρωνικού δικτύου, πραγματοποιήθηκαν δύο διαφορετικές τεχνικές κανονικοποίησης δεδομένων, MinMaxScaler και StandardScaler. Παρόλο που δοκιμάστηκαν και οι δύο τεχνικές, η τελική επιλογή έγινε με βάση την απόδοση του μοντέλου. Ο MinMaxScaler απέδωσε καλύτερα αποτελέσματα σε σχέση με τον StandardScaler, όπως φάνηκε από τις μετρικές αξιολόγησης. Εξαιτίας του μεγάλου μεγέθους του τελικού αρχείου, διατηρήθηκε μόνο το μοντέλο που χρησιμοποιεί τον MinMaxScaler, ο οποίος παρείχε τη βέλτιστη απόδοση, ενώ το μοντέλο με τον StandardScaler αφαιρέθηκε για εξοικονόμηση χώρου.

Περιγραφη:

Ο παραπάνω κώδικας περιγράφει τη διαδικασία εκπαίδευσης ενός νευρωνικού δικτύου με τη χρήση του βελτιστοποιητή Nadam (Nesterov-accelerated Adaptive Moment Estimation). Ο Nadam συνδυάζει την προσαρμοστικότητα του Adam με την επιτάχυνση Nesterov, προσφέροντας γρήγορη σύγκλιση και υψηλή απόδοση σε προβλήματα παλινδρόμησης.

Συνάρτησης train_evaluate_model_nn

Η συνάρτηση train_evaluate_model_nn εκπαιδεύει και αξιολογεί το νευρωνικό δίκτυο χρησιμοποιώντας τις συγκεκριμένες υπερπαραμέτρους που παρέχονται ως είσοδος:

- **Δομή του Νευρωνικού Δικτύου:**
    - Το δίκτυο αποτελείται από τρία κρυφά στρώματα (Dense) με ενεργοποίηση ReLU. Σε κάθε κρυφό στρώμα εφαρμόζεται dropout για την αποφυγή υπερπροσαρμογής.
    - Η έξοδος είναι ένας νευρώνας χωρίς ενεργοποίηση, κατάλληλος για παλινδρόμηση.
- **Βελτιστοποιητής:**
    - Ο βελτιστοποιητής Nadam χρησιμοποιείται για την εκπαίδευση. Ο Nadam ενσωματώνει την επιτάχυνση Nesterov, η οποία επιτρέπει την ταχύτερη και πιο σταθερή σύγκλιση κατά την εκπαίδευση.
- **Early Stopping:**
    - Η εκπαίδευση διακόπτεται νωρίς αν η απόδοση δεν βελτιώνεται για 10 συνεχόμενες εποχές, διατηρώντας έτσι το καλύτερο μοντέλο και αποφεύγοντας την υπερπροσαρμογή.
- **Αξιολόγηση:**
    - Μετά την εκπαίδευση, το μοντέλο αξιολογείται στο test set, και υπολογίζονται οι μετρικές απόδοσης, παρέχοντας μια πλήρη εικόνα της απόδοσης του μοντέλου.

Διερεύνηση Υπερπαραμέτρων

Η διαδικασία εύρεσης του βέλτιστου συνδυασμού υπερπαραμέτρων περιλαμβάνει πειραματισμούς με διαφορετικές τιμές για τον αριθμό νευρώνων, την πιθανότητα dropout, και τον ρυθμό μάθησης. Δοκιμάζονται διάφοροι συνδυασμοί για τον αριθμό των νευρώνων (num_nodes), την πιθανότητα dropout (dropout_prob), και τον ρυθμό μάθησης

(lr), με στόχο την ελαχιστοποίηση του Mean Absolute Error (MAE). Το μοντέλο με τις βέλτιστες υπερπαραμέτρους αποθηκεύεται για μελλοντική χρήση ή επαναξιολόγηση.

Αξιολόγηση του Βέλτιστου Μοντέλου

Μετά την επιλογή των καλύτερων υπερπαραμέτρων και την εκπαίδευση του αντίστοιχου μοντέλου, γίνεται εκ νέου αξιολόγηση στο test set για την επιβεβαίωση της απόδοσής του. Οι μετρικές απόδοσης που προκύπτουν από την αξιολόγηση, χρησιμοποιούνται για να συγκρίνουν την απόδοση αυτού του μοντέλου με άλλες προσεγγίσεις, παρέχοντας σαφή εικόνα της αποτελεσματικότητάς του.

Συμπεράσματα

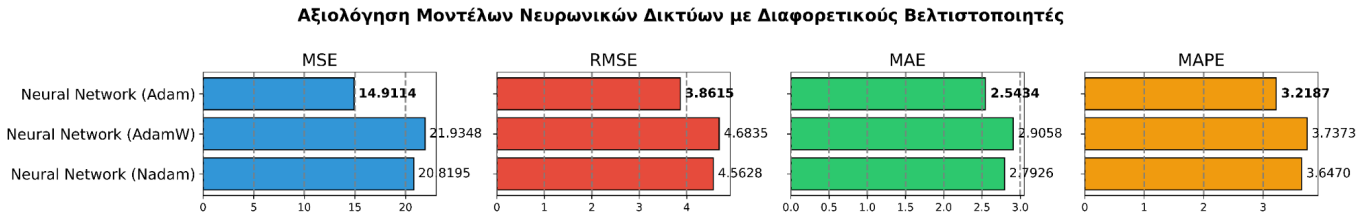
Η παραπάνω προσέγγιση αξιοποιεί τον βελτιστοποιητή Nadam για την εκπαίδευση ενός νευρωνικού δικτύου, επιτυγχάνοντας γρήγορη και σταθερή σύγκλιση. Η διαδικασία διερεύνησης υπερπαραμέτρων εξασφαλίζει ότι το μοντέλο έχει βελτιστοποιηθεί για το συγκεκριμένο πρόβλημα, ενώ η χρήση του Early Stopping βοηθά στην αποφυγή υπερπροσαρμογής. Η αποθήκευση του καλύτερου μοντέλου επιτρέπει την περαιτέρω αξιοποίηση των αποτελεσμάτων σε παραγωγικό περιβάλλον ή σε μελλοντικές αναλύσεις.

3.3.37 Αξιολόγηση Μοντέλων Νευρωνικών Δικτύων με Διαφορετικούς Βελτιστοποιητές

Αξιολόγηση Μοντέλων Νευρωνικών Δικτύων με Διαφορετικούς Βελτιστοποιητές

Model (Optimizer)	MSE	RMSE	MAE	MAPE
Neural Network (Adam)	14.9114	3.8615	2.5434	3.2187
Neural Network (AdamW)	21.9348	4.6835	2.9058	3.7373
Neural Network (Nadam)	20.8195	4.5628	2.7926	3.6470

Πίν.2: Αποτελέσματα Αξιολόγησης Νευρωνικών Δικτύων με Διαφορετικούς Βελτιστοποιητές



Σχ.2: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης Νευρωνικών Δικτύων για Διαφορετικούς Βελτιστοποιητές

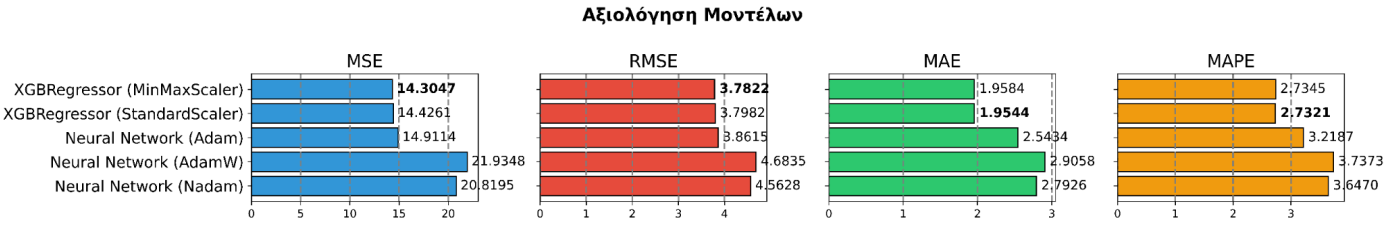
3.3.38 Αποτελέσματα της Εξέτασης των Μοντέλων

Μετά από την αξιολόγηση των μοντέλων, διαπιστώθηκε ότι το Gradient Boosting υπερείχε σταθερά έναντι των νευρωνικών δικτύων τόσο στην ακρίβεια πρόβλεψης όσο και στην ευρωστία των αποτελεσμάτων για το συγκεκριμένο σύνολο δεδομένων. Αυτή η υπεροχή του Gradient Boosting οδήγησε στη μετατόπιση της εστίασης προς τη χρήση των μοντέλων Gradient Boosting στις επόμενες μεθόδους συνόλου.

Αξιολόγηση Μοντέλων

Model	MSE	RMSE	MAE	MAPE
XGBRegressor (MinMaxScaler)	14.3047	3.7822	1.9584	2.7345
XGBRegressor (StandardScaler)	14.4261	3.7982	1.9544	2.7321
Neural Network (Adam)	14.9114	3.8615	2.5434	3.2187
Neural Network (AdamW)	21.9348	4.6835	2.9058	3.7373
Neural Network (Nadam)	20.8195	4.5628	2.7926	3.6470

Πίν.3: Συγκριτική Αξιολόγηση Μοντέλων XGBRegressor και Νευρωνικών Δικτύων



Σχ.3: Οπτική Αναπαράσταση Αποτελεσμάτων Αξιολόγησης για XGBRegressor και Νευρωνικά Δίκτυα

3.3.39 Δημιουργία και Εκπαίδευση Μοντέλων XGBoost για Κάθε Οδηγό

Driver1:

```
Y = df_driver1['safety']
X=df_driver1.drop('safety',axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = X_train.copy()
y_train_all = y_train.copy()
X_test_1 = X_test.copy()
y_test_1 = y_test.copy()
```

```
def prepare_data2(X_train, X_test, numeric_cols=numeric_cols,
one_hot_cols=one_hot_cols, multi_hot_cols=multi_hot_cols):

    X_train1, X_test1, scaler1, X_train2, X_test2, scaler2 =
scale_data(X_train, X_test, numeric_cols)

    X_train1, encoder, encoded_cols = create_fit_transform(X_train1,
one_hot_cols)
    X_train2 = transform_data(X_train2, encoder, encoded_cols, one_hot_cols)
    X_test1 = transform_data(X_test1, encoder, encoded_cols, one_hot_cols)
    X_test2 = transform_data(X_test2, encoder, encoded_cols, one_hot_cols)

    unique_keys_type = extract_unique_keys(X_train1['trip_events_type'])
    X_train1 = create_presence_columns(X_train1, 'trip_events_type',
unique_keys_type)
    X_train2 = create_presence_columns(X_train2, 'trip_events_type',
unique_keys_type)
    X_test1 = create_presence_columns(X_test1, 'trip_events_type',
unique_keys_type)
    X_test2 = create_presence_columns(X_test2, 'trip_events_type',
unique_keys_type)

    unique_keys_weather =
extract_unique_keys(X_train1['Weather_Main_raw_events_weather'])
    X_train1 = create_presence_columns(X_train1,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_train2 = create_presence_columns(X_train2,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test1 = create_presence_columns(X_test1,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test2 = create_presence_columns(X_test2,
'Weather_Main_raw_events_weather', unique_keys_weather)

    unique_keys_raw_type = extract_unique_keys(X_train1['Raw_type_counts'])
    X_train1 = create_presence_columns(X_train1, 'Raw_type_counts',
unique_keys_raw_type)
```

```
    X_train2 = create_presence_columns(X_train2, 'Raw_type_counts',
unique_keys_raw_type)
    X_test1 = create_presence_columns(X_test1, 'Raw_type_counts',
unique_keys_raw_type)
    X_test2 = create_presence_columns(X_test2, 'Raw_type_counts',
unique_keys_raw_type)

    unique_keys_raw_accident_trigger =
extract_unique_keys(X_train1['Raw_accident_trigger_counts'])
    X_train1 = create_presence_columns(X_train1, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_train2 = create_presence_columns(X_train2, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_test1 = create_presence_columns(X_test1, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)
    X_test2 = create_presence_columns(X_test2, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)

    X_train1.drop(multi_hot_cols, axis=1, inplace=True)
    X_train2.drop(multi_hot_cols, axis=1, inplace=True)
    X_test1.drop(multi_hot_cols, axis=1, inplace=True)
    X_test2.drop(multi_hot_cols, axis=1, inplace=True)

    return X_train1, X_train2, X_test1, X_test2, scaler1, scaler2, encoder,
encoded_cols, unique_keys_type, unique_keys_weather, unique_keys_raw_type,
unique_keys_raw_accident_trigger
```

```
(X_train1, X_train2, X_test1, X_test2, scaler2_1, scaler2_2, encoder2_1,
encoded_cols2_1, unique_keys_type2_1, unique_keys_weather2_1,
unique_keys_raw_type2_1, unique_keys_raw_accident_trigger2_1) =
prepare_data2(X_train, X_test)
```

```
def train_and_evaluate_model(X_train, y_train, X_test, y_test, **params):
    model = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, **params)
    model.fit(X_train, y_train)

    train_rmse = mean_squared_error(y_train, model.predict(X_train),
squared=False)
    test_rmse = mean_squared_error(y_test, model.predict(X_test),
squared=False)

    print('Train RMSE: {}, Test RMSE: {}'.format(train_rmse, test_rmse))

    train_mse = mean_squared_error(y_train, model.predict(X_train))
    test_mse = mean_squared_error(y_test, model.predict(X_test))

    print('Train MSE: {}, Test MSE: {}'.format(train_mse, test_mse))
    return test_mse
```

```
def objective(trial):
    n_estimators = trial.suggest_categorical('n_estimators', [25, 50, 80, 100,
120, 150, 200, 250, 300, 350, 450, 500, 550, 600, 650, 700, 750])
    max_depth = trial.suggest_int('max_depth', 4, 20)
    learning_rate = trial.suggest_categorical('learning_rate', [0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 0.75, 0.99])

    mse = train_and_evaluate_model(X_train2, y_train, X_test2, y_test,
n_estimators=n_estimators, max_depth=max_depth, learning_rate=learning_rate)

    return mse
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=250)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
def evaluate_model(model, X_train, y_train, X_test, y_test):
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    train_mse = mean_squared_error(y_train, y_train_pred)
    test_mse = mean_squared_error(y_test, y_test_pred)
    train_rmse = mean_squared_error(y_train, y_train_pred, squared=False)
    test_rmse = mean_squared_error(y_test, y_test_pred, squared=False)
    test_mae = mean_absolute_error(y_test, y_test_pred)
    test_mape = mean_absolute_percentage_error(y_test, y_test_pred) * 100

    print(f"Train MSE: {train_mse}, Test MSE: {test_mse}")
    print(f"Train RMSE: {train_rmse}, Test RMSE: {test_rmse}")
    print(f"Test Mean Absolute Error: {test_mae}")
    print(f"Test Mean Absolute Percentage Error: {test_mape} %")

    return {
        "train_mse": train_mse,
        "test_mse": test_mse,
        "train_rmse": train_rmse,
        "test_rmse": test_rmse,
        "test_mae": test_mae,
        "test_mape": test_mape
    }
```

```
xgb_model2_1_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=650, max_depth=6, learning_rate=0.01)
xgb_model2_1_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_1_1, X_train1, y_train, X_test1, y_test)
```

```
xgb_model2_1_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=650, max_depth=6, learning_rate=0.01)
xgb_model2_1_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_1_2, X_train2, y_train, X_test2, y_test)
```

Driver2:

```
Y = df_driver2['safety']
X=df_driver2.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_2 = X_test.copy()
y_test_2 = y_test.copy()
(X_train1, X_train2, X_test1, X_test2, scaler2_3, scaler2_4, encoder2_2,
encoded_cols2_2, unique_keys_type2_2, unique_keys_weather2_2,
unique_keys_raw_type2_2, unique_keys_raw_accident_trigger2_2) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_2_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=300, max_depth=14, learning_rate=0.1)
xgb_model2_2_1.fit(X_train1, y_train)

metrics = evaluate_model(xgb_model2_2_1, X_train1, y_train, X_test1, y_test)
```

```
xgb_model2_2_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=300, max_depth=14, learning_rate=0.1)
xgb_model2_2_2.fit(X_train2, y_train)
```

```
metrics = evaluate_model(xgb_model2_2_2, X_train2, y_train, X_test2, y_test)
```

Driver3:

```
Y = df_driver3['safety']
X=df_driver3.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_3 = X_test.copy()
y_test_3 = y_test.copy()

(X_train1, X_train2, X_test1, X_test2, scaler2_5, scaler2_6, encoder2_3,
encoded_cols2_3, unique_keys_type2_3, unique_keys_weather2_3,
unique_keys_raw_type2_3, unique_keys_raw_accident_trigger2_3) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_3_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=100, max_depth=7, learning_rate=0.99)
xgb_model2_3_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_3_1, X_train1, y_train, X_test1, y_test)
```

```
xgb_model2_3_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=100, max_depth=7, learning_rate=0.99)
xgb_model2_3_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_3_2, X_train2, y_train, X_test2, y_test)
```

Driver4:

```
Y = df_driver4['safety']
X=df_driver4.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_4 = X_test.copy()
y_test_4 = y_test.copy()

(X_train1, X_train2, X_test1, X_test2, scaler2_7, scaler2_8, encoder2_4,
encoded_cols2_4, unique_keys_type2_4, unique_keys_weather2_4,
unique_keys_raw_type2_4, unique_keys_raw_accident_trigger2_4) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_4_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=120, max_depth=5, learning_rate=0.1)
xgb_model2_4_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_4_1, X_train1, y_train, X_test1, y_test)

xgb_model2_4_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=120, max_depth=5, learning_rate=0.1)
xgb_model2_4_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_4_2, X_train2, y_train, X_test2, y_test)
```

Driver5:

```
Y = df_driver5['safety']
X=df_driver5.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
```

```
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_5 = X_test.copy()
y_test_5 = y_test.copy()

(X_train1, X_train2, X_test1, X_test2, scaler2_9, scaler2_10, encoder2_5,
encoded_cols2_5, unique_keys_type2_5, unique_keys_weather2_5,
unique_keys_raw_type2_5, unique_keys_raw_accident_trigger2_5) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_5_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=650, max_depth=15, learning_rate=0.05)
xgb_model2_5_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_5_1, X_train1, y_train, X_test1, y_test)

xgb_model2_5_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=650, max_depth=15, learning_rate=0.05)
xgb_model2_5_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_5_2, X_train2, y_train, X_test2, y_test)
```

Driver6:

```
Y = df_driver6['safety']
X=df_driver6.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_6 = X_test.copy()
y_test_6 = y_test.copy()

(X_train1, X_train2, X_test1, X_test2, scaler2_11, scaler2_12, encoder2_6,
encoded_cols2_6, unique_keys_type2_6, unique_keys_weather2_6,
unique_keys_raw_type2_6, unique_keys_raw_accident_trigger2_6) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=250)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_6_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=450, max_depth=8, learning_rate=0.005)
xgb_model2_6_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_6_1, X_train1, y_train, X_test1, y_test)

xgb_model2_6_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=450, max_depth=8, learning_rate=0.005)
xgb_model2_6_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_6_2, X_train2, y_train, X_test2, y_test)
```

Driver7:

```
Y = df_driver7['safety']
X=df_driver7.drop('safety',axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
X_train_all = pd.concat([X_train_all, X_train], ignore_index=True)
y_train_all = pd.concat([y_train_all, y_train], ignore_index=True)

X_test_7 = X_test.copy()
y_test_7 = y_test.copy()

(X_train1, X_train2, X_test1, X_test2, scaler2_13, scaler2_14, encoder2_7,
encoded_cols2_7, unique_keys_type2_7, unique_keys_weather2_7,
unique_keys_raw_type2_7, unique_keys_raw_accident_trigger2_7) =
prepare_data2(X_train, X_test)
```

```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=350)
best_params = study.best_params
best_mse = study.best_value

print("Best parameters:", best_params)
print("Best MSE:", best_mse)
```

```
xgb_model2_7_1 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=750, max_depth=4, learning_rate=0.01)
xgb_model2_7_1.fit(X_train1, y_train)
metrics = evaluate_model(xgb_model2_7_1, X_train1, y_train, X_test1, y_test)

xgb_model2_7_2 = XGBRegressor(n_jobs=-1, random_state=42, subsample=0.7,
colsample_bytree=0.7, n_estimators=750, max_depth=4, learning_rate=0.01)
xgb_model2_7_2.fit(X_train2, y_train)
metrics = evaluate_model(xgb_model2_7_2, X_train2, y_train, X_test2, y_test)
```

Η παραπάνω διαδικασία περιγράφει τη δημιουργία και εκπαίδευση ξεχωριστών μοντέλων XGBoost για κάθε οδηγό στο dataset. Κάθε μοντέλο βελτιστοποιείται μέσω τεχνικών αναζήτησης υπερπαραμέτρων για την επίτευξη της βέλτιστης απόδοσης. Τα εκπαιδευμένα μοντέλα θα χρησιμοποιηθούν σε μεθόδους συνόλου (ensemble methods) για τη βελτίωση της ακρίβειας των προβλέψεων.

Διαχωρισμός Δεδομένων για Κάθε Οδηγό

Για κάθε οδηγό, τα δεδομένα χωρίζονται σε σύνολα εκπαίδευσης (X_train, y_train) και δοκιμής (X_test, y_test) με χρήση της μεθόδου train_test_split. Το σύνολο εκπαίδευσης χρησιμοποιείται για την εκπαίδευση του μοντέλου, ενώ το σύνολο δοκιμής για την αξιολόγηση της απόδοσης του μοντέλου.

Προετοιμασία των Δεδομένων (Preprocessing)

Η προετοιμασία των δεδομένων πραγματοποιείται μέσω της συνάρτησης prepare_data2, η οποία εκτελεί τις παρακάτω ενέργειες:

- **Κλιμάκωση δεδομένων:** Τα δεδομένα κλιμακώνονται χρησιμοποιώντας δύο διαφορετικές τεχνικές, MinMaxScaler και StandardScaler, για να εξεταστεί ποια από τις δύο αποδίδει καλύτερα.
- **Κωδικοποίηση κατηγορικών δεδομένων:** Οι κατηγορικές μεταβλητές μετατρέπονται σε αριθμητικές μέσω τεχνικών One-Hot Encoding.
- **Δημιουργία στηλών παρουσίας (Presence Columns):** Για τα multi-hot δεδομένα, δημιουργούνται νέες στήλες που δείχνουν την παρουσία συγκεκριμένων χαρακτηριστικών.

Εκπαίδευση και Αξιολόγηση των Μοντέλων

Για κάθε οδηγό, εκπαιδεύεται ένα μοντέλο XGBoost με τις βέλτιστες υπερπαραμέτρους που έχουν βρεθεί. Η συνάρτηση train_and_evaluate_model χρησιμοποιείται για την εκπαίδευση του μοντέλου και την αξιολόγησή του, με υπολογισμό μετρικών όπως RMSE και MSE τόσο στα δεδομένα εκπαίδευσης όσο και στα δεδομένα δοκιμής.

Βελτιστοποίηση Υπερπαραμέτρων (Hyperparameter Optimization)

Για να επιτευχθεί η βέλτιστη απόδοση του μοντέλου, χρησιμοποιείται η βιβλιοθήκη Optuna για την αναζήτηση των υπερπαραμέτρων. Η συνάρτηση objective ορίζει τη διαδικασία βελτιστοποίησης, όπου δοκιμάζονται διαφορετικοί συνδυασμοί υπερπαραμέτρων όπως n_estimators, max_depth, και learning_rate, με στόχο την ελαχιστοποίηση του MSE.

Αξιολόγηση των Τελικών Μοντέλων

Μετά τη βελτιστοποίηση, τα μοντέλα επανεκπαιδεύονται με τις καλύτερες υπερπαραμέτρους και αξιολογούνται εκ νέου στα δεδομένα δοκιμής. Η συνάρτηση evaluate_model χρησιμοποιείται για να υπολογίσει τις μετρικές απόδοσης, παρέχοντας μια ολοκληρωμένη εικόνα της απόδοσης του κάθε μοντέλου.

Επανάληψη για Κάθε Οδηγό

Αυτή η διαδικασία επαναλαμβάνεται για κάθε οδηγό στο dataset, δημιουργώντας συνολικά επτά ξεχωριστά μοντέλα. Τα μοντέλα αυτά θα χρησιμοποιηθούν σε τεχνικές συνόλου (ensemble methods) για την αύξηση της ακρίβειας των προβλέψεων και την εξαγωγή πιο γενικευμένων συμπερασμάτων.

3.3.40 Ενοποίηση και Τυχαία Δειγματοληψία Δεδομένων Δοκιμής από Όλους τους Οδηγούς

```
test_driver1 = pd.concat([X_test_1, y_test_1], axis=1)
test_driver2 = pd.concat([X_test_2, y_test_2], axis=1)
test_driver3 = pd.concat([X_test_3, y_test_3], axis=1)
test_driver4 = pd.concat([X_test_4, y_test_4], axis=1)
test_driver5 = pd.concat([X_test_5, y_test_5], axis=1)
test_driver6 = pd.concat([X_test_6, y_test_6], axis=1)
test_driver7 = pd.concat([X_test_7, y_test_7], axis=1)
merged_df = pd.concat([test_driver1, test_driver2, test_driver3,
               test_driver4, test_driver5, test_driver6, test_driver7], axis=0)
```

```
shuffled_df = merged_df.sample(frac=1).reset_index(drop=True)
y_test = shuffled_df['safety']
```

Αφού ολοκληρωθεί η εκπαίδευση των μοντέλων για κάθε οδηγό ξεχωριστά, το επόμενο βήμα είναι η ενοποίηση των δεδομένων δοκιμής από όλους τους οδηγούς και η δημιουργία ενός ενιαίου συνόλου δεδομένων δοκιμής. Αυτό το σύνολο θα χρησιμοποιηθεί για την αξιολόγηση των μεθόδων συνόλου (ensemble methods) και την περαιτέρω βελτίωση της απόδοσης των μοντέλων.

Ενοποίηση των Δεδομένων Δοκιμής

Αρχικά, τα δεδομένα δοκιμής (X_test και y_test) για κάθε οδηγό συνενώνονται σε ξεχωριστά DataFrames, τα οποία περιέχουν τόσο τα χαρακτηριστικά (features) όσο και τις ετικέτες (labels) των δεδομένων δοκιμής για τον αντίστοιχο οδηγό. Αυτά τα DataFrames συνενώνονται (concatenate) για να δημιουργηθεί ένα ενιαίο σύνολο δεδομένων δοκιμής που περιλαμβάνει δείγματα από όλους τους οδηγούς.

Τυχαία Δειγματοληψία των Δεδομένων

Μετά την ενοποίηση, πραγματοποιείται τυχαία δειγματοληψία (shuffling) των δεδομένων. Αυτή η ενέργεια διασφαλίζει ότι τα δεδομένα δοκιμής δεν είναι διατεταγμένα με βάση τον οδηγό.

Εξαγωγή των Ετικετών Δοκιμής

Τέλος, οι ετικέτες (labels) του τελικού συνόλου δεδομένων δοκιμής εξάγονται από το πεδίο safety, το οποίο θα χρησιμοποιηθεί για την αξιολόγηση της απόδοσης των μοντέλων.

Προετοιμασία για Αξιολόγηση των Μεθόδων Συνόλου

Αυτή η διαδικασία οδηγεί στη δημιουργία ενός ενοποιημένου και τυχαία δειγματοληπτικού συνόλου δεδομένων δοκιμής, το οποίο είναι έτοιμο για χρήση σε μεθόδους συνόλου (ensemble methods). Η τυχαία δειγματοληψία εξασφαλίζει ότι η αξιολόγηση των μοντέλων θα είναι δίκαιη και αντιπροσωπευτική, με τα δεδομένα από διαφορετικούς οδηγούς να είναι ομοιόμορφα κατανεμημένα στο σύνολο δοκιμής. Αυτό επιτρέπει μια πιο αξιόπιστη και γενικευμένη εκτίμηση της απόδοσης των μοντέλων, συμβάλλοντας στη βελτίωση της ακρίβειας των προβλέψεων.

3.3.41 Προεπεξεργασία και Αξιολόγηση Μοντέλων για Κάθε Οδηγό

```
def preprocess_data(shuffled_df, scaler1, scaler2, encoder, encoded_cols,
        unique_keys_type, unique_keys_weather, unique_keys_raw_type,
        unique_keys_raw_accident_trigger, numeric_cols=numeric_cols,
        one_hot_cols=one_hot_cols, multi_hot_cols=multi_hot_cols):

    X_test1 = shuffled_df.drop('safety', axis=1)
    X_test2 = X_test1.copy()

    X_test1[numeric_cols] = scaler1.transform(X_test1[numeric_cols])
    X_test2[numeric_cols] = scaler2.transform(X_test2[numeric_cols])

    X_test1 = transform_data(X_test1, encoder, encoded_cols, one_hot_cols)
    X_test2 = transform_data(X_test2, encoder, encoded_cols, one_hot_cols)

    X_test1 = create_presence_columns(X_test1, 'trip_events_type',
unique_keys_type)
    X_test1 = create_presence_columns(X_test1,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test1 = create_presence_columns(X_test1, 'Raw_type_counts',
unique_keys_raw_type)
    X_test1 = create_presence_columns(X_test1, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)

    X_test2 = create_presence_columns(X_test2, 'trip_events_type',
unique_keys_type)
    X_test2 = create_presence_columns(X_test2,
'Weather_Main_raw_events_weather', unique_keys_weather)
    X_test2 = create_presence_columns(X_test2, 'Raw_type_counts',
unique_keys_raw_type)
    X_test2 = create_presence_columns(X_test2, 'Raw_accident_trigger_counts',
unique_keys_raw_accident_trigger)

    X_test1.drop(multi_hot_cols, axis=1, inplace=True)
    X_test2.drop(multi_hot_cols, axis=1, inplace=True)

    return X_test1, X_test2
```

```
def print_evaluation_metrics(y_test, y_pred):
    mse = mean_squared_error(y_test, y_pred)
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    mae = mean_absolute_error(y_test, y_pred)
    mape = mean_absolute_percentage_error(y_test, y_pred) * 100

    print(f"Test MSE: {mse}")
    print(f"Test RMSE: {rmse}")
    print(f"Test Mean Absolute Error: {mae}")
    print(f"Test Mean Absolute Percentage Error: {mape} %")
```

```
    return {
        "mse": mse,
        "rmse": rmse,
        "mae": mae,
        "mape": mape
    }
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_1, scaler2_2,
encoder2_1, encoded_cols2_1, unique_keys_type2_1, unique_keys_weather2_1,
unique_keys_raw_type2_1, unique_keys_raw_accident_trigger2_1)

preds_model1_1 = xgb_model2_1_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model1_1)

preds_model1_2 = xgb_model2_1_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model1_2)
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_3, scaler2_4,
encoder2_2, encoded_cols2_2, unique_keys_type2_2, unique_keys_weather2_2,
unique_keys_raw_type2_2, unique_keys_raw_accident_trigger2_2)

preds_model2_1 = xgb_model2_2_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model2_1)

preds_model2_2 = xgb_model2_2_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model2_2)
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_5, scaler2_6,
encoder2_3, encoded_cols2_3, unique_keys_type2_3, unique_keys_weather2_3,
unique_keys_raw_type2_3, unique_keys_raw_accident_trigger2_3)

preds_model3_1 = xgb_model2_3_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model3_1)

preds_model3_2 = xgb_model2_3_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model3_2)
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_7, scaler2_8,
encoder2_4, encoded_cols2_4, unique_keys_type2_4, unique_keys_weather2_4,
unique_keys_raw_type2_4, unique_keys_raw_accident_trigger2_4)

preds_model4_1 = xgb_model2_4_1.predict(X_test1)
```

```
metrics = print_evaluation_metrics(y_test, preds_model4_1)

preds_model4_2 = xgb_model2_4_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model4_2)
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_9, scaler2_10,
encoder2_5, encoded_cols2_5, unique_keys_type2_5, unique_keys_weather2_5,
unique_keys_raw_type2_5, unique_keys_raw_accident_trigger2_5)

preds_model5_1 = xgb_model2_5_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model5_1)

preds_model5_2 = xgb_model2_5_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model5_2)
```

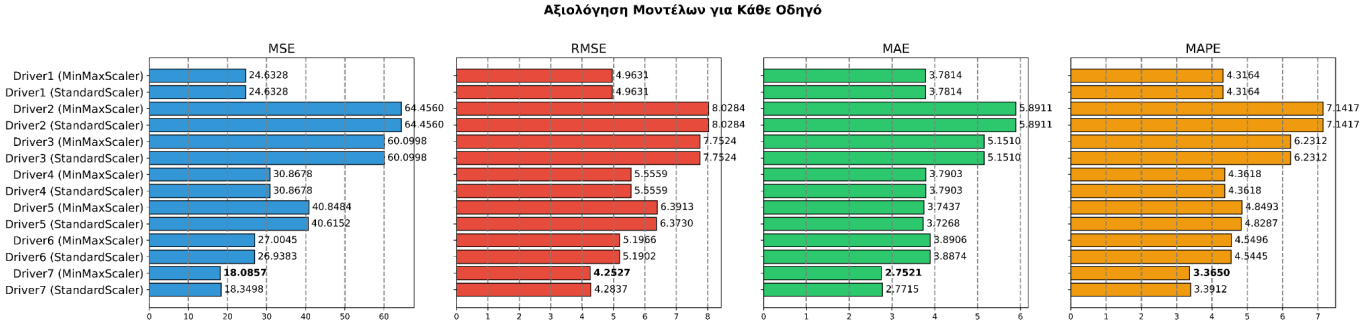
```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_11, scaler2_12,
encoder2_6, encoded_cols2_6, unique_keys_type2_6, unique_keys_weather2_6,
unique_keys_raw_type2_6, unique_keys_raw_accident_trigger2_6)

preds_model6_1 = xgb_model2_6_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model6_1)

preds_model6_2 = xgb_model2_6_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model6_2)
```

```
X_test1, X_test2 = preprocess_data(shuffled_df, scaler2_13, scaler2_14,
encoder2_7, encoded_cols2_7, unique_keys_type2_7, unique_keys_weather2_7,
unique_keys_raw_type2_7, unique_keys_raw_accident_trigger2_7)

preds_model7_1 = xgb_model2_7_1.predict(X_test1)
metrics = print_evaluation_metrics(y_test, preds_model7_1)

preds_model7_2 = xgb_model2_7_2.predict(X_test2)
metrics = print_evaluation_metrics(y_test, preds_model7_2)
```

Αξιολόγηση Μοντέλων για Κάθε Οδηγό

Driver (Scaler)	MSE	RMSE	MAE	MAPE
Driver1 (MinMaxScaler)	24.6328	4.9631	3.7814	4.3164
Driver1 (StandardScaler)	24.6328	4.9631	3.7814	4.3164
Driver2 (MinMaxScaler)	64.4560	8.0284	5.8911	7.1417
Driver2 (StandardScaler)	64.4560	8.0284	5.8911	7.1417
Driver3 (MinMaxScaler)	60.0998	7.7524	5.1510	6.2312
Driver3 (StandardScaler)	60.0998	7.7524	5.1510	6.2312
Driver4 (MinMaxScaler)	30.8678	5.5559	3.7903	4.3618
Driver4 (StandardScaler)	30.8678	5.5559	3.7903	4.3618
Driver5 (MinMaxScaler)	40.8484	6.3913	3.7437	4.8493
Driver5 (StandardScaler)	40.6152	6.3730	3.7268	4.8287
Driver6 (MinMaxScaler)	27.0045	5.1966	3.8906	4.5496
Driver6 (StandardScaler)	26.9383	5.1902	3.8874	4.5445
Driver7 (MinMaxScaler)	18.0857	4.2527	2.7521	3.3650
Driver7 (StandardScaler)	18.3498	4.2837	2.7715	3.3912

Πίν.4: Αποτελέσματα Αξιολόγησης Μοντέλων για Κάθε Οδηγό με Διαφορετικές Τεχνικές Κλιμάκωσης



Σχ.4: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης Μοντέλων για Κάθε Οδηγό με Διάφορες Τεχνικές Κλιμάκωσης

Μετά την τυχαία δειγματοληψία του ενιαίου συνόλου δεδομένων δοκιμής, το επόμενο βήμα ήταν να προεπεξεργαστούμε αυτά τα δεδομένα και να τα χρησιμοποιήσουμε για την αξιολόγηση των εκπαιδευμένων μοντέλων. Σε αυτή την υποενότητα, εφαρμόσαμε τα μοντέλα που εκπαιδεύτηκαν ξεχωριστά για κάθε οδηγό, προκειμένου να συγκρίνουμε την απόδοσή τους στο κοινό σύνολο δεδομένων δοκιμής.

Προεπεξεργασία των Δεδομένων Δοκιμής

Αρχικά, τα δεδομένα δοκιμής προετοιμάστηκαν χρησιμοποιώντας τη συνάρτηση preprocess_data. Αυτή η συνάρτηση εφαρμόζει δύο διαφορετικές τεχνικές κανονικοποίησης (scalers), τον MinMaxScaler και τον StandardScaler. Επιπλέον, χρησιμοποιείται κωδικοποίηση των κατηγορικών μεταβλητών μέσω OneHotEncoder,

καθώς και δημιουργία επιπλέον στηλών που αντιπροσωπεύουν την παρουσία ή απουσία συγκεκριμένων κατηγοριών σε πολυ-κατηγορικές μεταβλητές.

Αυτή η διαδικασία δημιουργεί δύο διαφορετικές εκδοχές των δεδομένων δοκιμής, μία κανονικοποιημένη με MinMaxScaler και μία με StandardScaler. Αυτό μας επιτρέπει να αξιολογήσουμε ποια τεχνική κανονικοποίησης αποδίδει καλύτερα στο συγκεκριμένο πρόβλημα.

Αξιολόγηση των Μοντέλων

Για κάθε μοντέλο που είχε εκπαιδευτεί ξεχωριστά για κάθε οδηγό, πραγματοποιήθηκαν προβλέψεις στα δεδομένα δοκιμής και αξιολογήθηκε η απόδοσή του χρησιμοποιώντας τις μετρικές MSE, RMSE, MAE και MAPE.

Η ίδια διαδικασία ακολουθήθηκε για κάθε μοντέλο, χρησιμοποιώντας τόσο την κανονικοποίηση MinMaxScaler όσο και την κανονικοποίηση StandardScaler. Αυτό μας επιτρέπει να συγκρίνουμε την απόδοση των μοντέλων και να εντοπίσουμε την καλύτερη μέθοδο κανονικοποίησης για το συγκεκριμένο σύνολο δεδομένων.

3.3.42 Συνδυασμός Μοντέλων με Χρήση Voting Regressor

```
X_train_all_1, X_test_all_1, scaler1, X_train_all_2, X_test_all_2, scaler2 =
scale_data(X_train_all, shuffled_df.drop('safety', axis=1), numeric_cols)

X_train_all_1, encoder_X_train_all, encoded_cols_X_train_all =
create_fit_transform(X_train_all_1, one_hot_cols)
X_train_all_2 = transform_data(X_train_all_2, encoder_X_train_all,
encoded_cols_X_train_all, one_hot_cols)

unique_keys_type_X_train_all =
extract_unique_keys(X_train_all_1['trip_events_type'])
X_train_all_1 = create_presence_columns(X_train_all_1, 'trip_events_type',
unique_keys_type_X_train_all)
X_train_all_2 = create_presence_columns(X_train_all_2, 'trip_events_type',
unique_keys_type_X_train_all)

unique_keys_weather_X_train_all =
extract_unique_keys(X_train_all_1['Weather_Main_raw_events_weather'])
X_train_all_1 = create_presence_columns(X_train_all_1,
'Weather_Main_raw_events_weather', unique_keys_weather_X_train_all)
X_train_all_2 = create_presence_columns(X_train_all_2,
'Weather_Main_raw_events_weather', unique_keys_weather_X_train_all)
```

```
unique_keys_raw_type_X_train_all =
extract_unique_keys(X_train_all_1['Raw_type_counts'])
X_train_all_1 = create_presence_columns(X_train_all_1, 'Raw_type_counts',
unique_keys_raw_type_X_train_all)
X_train_all_2 = create_presence_columns(X_train_all_2, 'Raw_type_counts',
unique_keys_raw_type_X_train_all)

unique_keys_raw_accident_trigger_X_train_all =
extract_unique_keys(X_train_all_1['Raw_accident_trigger_counts'])
X_train_all_1 = create_presence_columns(X_train_all_1,
'Raw_accident_trigger_counts', unique_keys_raw_accident_trigger_X_train_all)
X_train_all_2 = create_presence_columns(X_train_all_2,
'Raw_accident_trigger_counts', unique_keys_raw_accident_trigger_X_train_all)

X_train_all_1.drop(multi_hot_cols, axis=1, inplace=True)
X_train_all_2.drop(multi_hot_cols, axis=1, inplace=True)

X_test_all_1 = transform_data(X_test_all_1, encoder_X_train_all,
encoded_cols_X_train_all, one_hot_cols)
X_test_all_2 = transform_data(X_test_all_2, encoder_X_train_all,
encoded_cols_X_train_all, one_hot_cols)

X_test_all_1 = create_presence_columns(X_test_all_1, 'trip_events_type',
unique_keys_type_X_train_all)
X_test_all_1 = create_presence_columns(X_test_all_1,
'Weather_Main_raw_events_weather', unique_keys_weather_X_train_all)
X_test_all_1 = create_presence_columns(X_test_all_1, 'Raw_type_counts',
unique_keys_raw_type_X_train_all)
X_test_all_1 = create_presence_columns(X_test_all_1,
'Raw_accident_trigger_counts', unique_keys_raw_accident_trigger_X_train_all)
X_test_all_1.drop(multi_hot_cols, axis=1, inplace=True)

X_test_all_2 = create_presence_columns(X_test_all_2, 'trip_events_type',
unique_keys_type_X_train_all)
X_test_all_2 = create_presence_columns(X_test_all_2,
'Weather_Main_raw_events_weather', unique_keys_weather_X_train_all)
X_test_all_2 = create_presence_columns(X_test_all_2, 'Raw_type_counts',
unique_keys_raw_type_X_train_all)
X_test_all_2 = create_presence_columns(X_test_all_2,
'Raw_accident_trigger_counts', unique_keys_raw_accident_trigger_X_train_all)
X_test_all_2.drop(multi_hot_cols, axis=1, inplace=True)
```

```
voting_regressor1 = VotingRegressor(estimators=[
    ('Model_Driver1', xgb_model2_1_1),
    ('Model_Driver2', xgb_model2_2_1),
    ('Model_Driver3', xgb_model2_3_1),
    ('Model_Driver4', xgb_model2_4_1),
    ('Model_Driver5', xgb_model2_5_1),
    ('Model_Driver6', xgb_model2_6_1),
    ('Model_Driver7', xgb_model2_7_1)
```

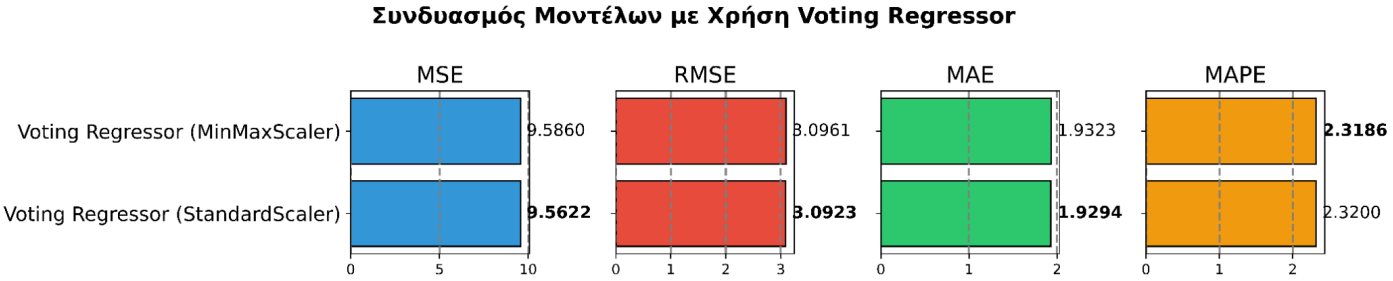
```
])

voting_regressor1.fit(X_train_all_1, y_train_all)
y_pred1 = voting_regressor1.predict(X_test_all_1)
metrics = print_evaluation_metrics(y_test, y_pred1)
```

```
voting_regressor2 = VotingRegressor(estimators=[
    ('Model_Driver1', xgb_model2_1_2),
    ('Model_Driver2', xgb_model2_2_2),
    ('Model_Driver3', xgb_model2_3_2),
    ('Model_Driver4', xgb_model2_4_2),
    ('Model_Driver5', xgb_model2_5_2),
    ('Model_Driver6', xgb_model2_6_2),
    ('Model_Driver7', xgb_model2_7_2)
])

voting_regressor2.fit(X_train_all_2, y_train_all)
y_pred2 = voting_regressor2.predict(X_test_all_2)
metrics = print_evaluation_metrics(y_test, y_pred2)
```

Συνδυασμός Μοντέλων με Χρήση Voting Regressor

Technique (Scaler)	MSE	RMSE	MAE	MAPE
Voting Regressor (MinMaxScaler)	9.5860	3.0961	1.9323	2.3186
Voting Regressor (StandardScaler)	9.5622	3.0923	1.9294	2.3200

Πίν.5: Αποτελέσματα Αξιολόγησης του Voting Regressor με Διαφορετικές Τεχνικές Κλιμάκωσης



Σχ.5: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης του Voting Regressor με Διαφορετικές Τεχνικές Κλιμάκωσης

Στην προσπάθεια για περαιτέρω βελτίωση της ακρίβειας των προβλέψεων, εφαρμόστηκε η τεχνική του Voting Regressor. Αυτή η μέθοδος συνδυάζει τις προβλέψεις από τα επιμέρους μοντέλα που είχαν εκπαιδευτεί για κάθε οδηγό, δημιουργώντας ένα ενιαίο μοντέλο συνόλου. Ο στόχος είναι να εκμεταλλευτούμε τις δυνάμεις των ξεχωριστών μοντέλων για να επιτύχουμε καλύτερη συνολική απόδοση.

Προεπεξεργασία Δεδομένων

Αρχικά, όλα τα δεδομένα εκπαίδευσης συνδυάστηκαν και προεπεξεργάστηκαν με δύο διαφορετικούς τρόπους, χρησιμοποιώντας τον MinMaxScaler και τον StandardScaler για την κανονικοποίηση των αριθμητικών χαρακτηριστικών. Ακολούθησε η κωδικοποίηση των κατηγορικών χαρακτηριστικών μέσω της τεχνικής One-Hot Encoding και η δημιουργία στηλών παρουσίας για τα πολυκατηγοριακά χαρακτηριστικά (multi-hot encoded columns). Οι ίδιες διαδικασίες εφαρμόστηκαν και στα δεδομένα δοκιμής.

Εκπαίδευση και Αξιολόγηση του Voting Regressor

Μετά την προετοιμασία των δεδομένων, δημιουργήθηκαν δύο μοντέλα συνόλου χρησιμοποιώντας τον Voting Regressor. Το πρώτο μοντέλο εκπαιδεύτηκε με τα δεδομένα που κανονικοποιήθηκαν με MinMaxScaler, ενώ το δεύτερο με τα δεδομένα που κανονικοποιήθηκαν με StandardScaler.

Αποτελέσματα και Συμπεράσματα

Η χρήση του Voting Regressor επέτρεψε τον συνδυασμό των προβλέψεων από τα επιμέρους μοντέλα σε ένα ενιαίο μοντέλο, το οποίο παρουσίασε βελτιωμένη ακρίβεια και σταθερότητα στις προβλέψεις. Το συνδυαστικό μοντέλο κατάφερε να υπερισχύσει των μεμονωμένων μοντέλων.

Αυτό το αποτέλεσμα δείχνει ότι η προσέγγιση του συνόλου μπορεί να είναι ιδιαίτερα αποτελεσματική για την επίλυση προβλημάτων πρόβλεψης, καθώς επιτρέπει την αξιοποίηση των πλεονεκτημάτων κάθε επιμέρους μοντέλου. Η σύγκριση των δύο μοντέλων συνόλου (ένα βασισμένο στον MinMaxScaler και το άλλο στον StandardScaler) έδειξε ότι και τα δύο είχαν καλή απόδοση, με μικρές διαφορές στην ακρίβεια ανάλογα με τη μέθοδο κανονικοποίησης που χρησιμοποιήθηκε.

Η ενσωμάτωση του Voting Regressor στην ανάλυση επιβεβαίωσε την αξία των τεχνικών συνόλου στην αύξηση της αξιοπιστίας των προβλέψεων, καθιστώντας το ένα χρήσιμο εργαλείο για εφαρμογές σε προβλήματα παρόμοιας φύσης. Αυτή η τεχνική αποδεικνύει ότι η συλλογική απόδοση πολλαπλών μοντέλων μπορεί να οδηγήσει σε καλύτερα αποτελέσματα από ό,τι η χρήση ενός μόνο μοντέλου.

3.3.43 Συνδυασμός Μοντέλων με Χρήση Stacking Regressor

```
base_models = [
    ('Model_Driver1', xgb_model2_1_1),
    ('Model_Driver2', xgb_model2_2_1),
    ('Model_Driver3', xgb_model2_3_1),
    ('Model_Driver4', xgb_model2_4_1),
    ('Model_Driver5', xgb_model2_5_1),
    ('Model_Driver6', xgb_model2_6_1),
    ('Model_Driver7', xgb_model2_7_1)
]

meta_models = [
    Ridge(alpha=1.0),
    Lasso(alpha=0.1),
    ElasticNet(alpha=0.1, l1_ratio=0.7),
    SVR(kernel='linear', C=1.0),
    RandomForestRegressor(n_estimators=250, random_state=42),
    XGBRegressor(n_estimators=250, random_state=42)
]

for meta_model in meta_models:
    print(f"Training with meta-model: {meta_model}")

    stacking_regressor = StackingRegressor(estimators=base_models,
final_estimator=meta_model)
    stacking_regressor.fit(X_train_all_1, y_train_all)

    y_pred1 = stacking_regressor.predict(X_test_all_1)
    metrics = print_evaluation_metrics(y_test, y_pred1)
    print("\n", "\n")
```

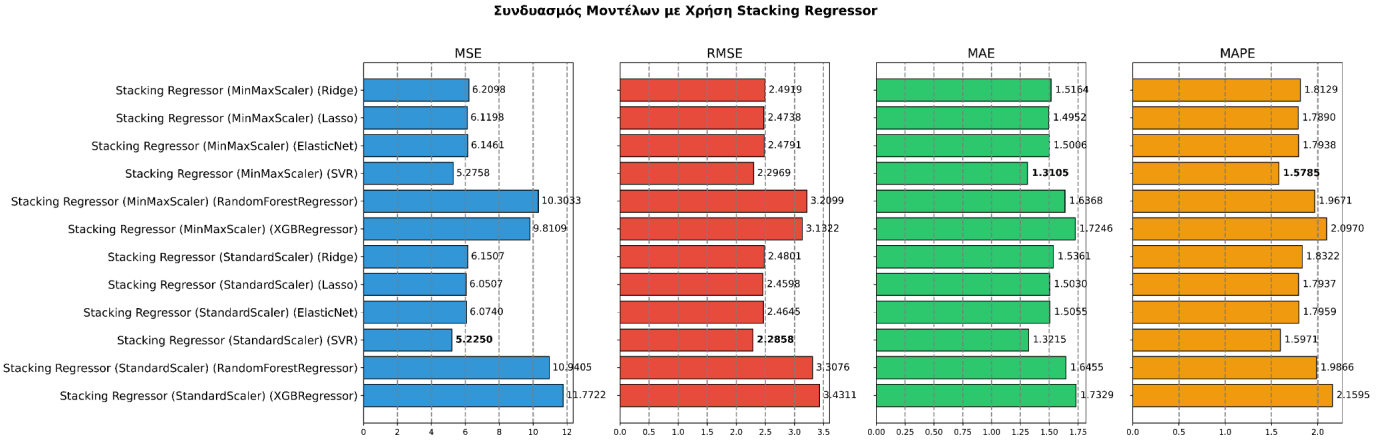
```
base_models2 = [
    ('Model_Driver1', xgb_model2_1_2),
    ('Model_Driver2', xgb_model2_2_2),
    ('Model_Driver3', xgb_model2_3_2),
    ('Model_Driver4', xgb_model2_4_2),
    ('Model_Driver5', xgb_model2_5_2),
    ('Model_Driver6', xgb_model2_6_2),
    ('Model_Driver7', xgb_model2_7_2)
]
```

```
for meta_model in meta_models:
    print(f"Training with meta-model: {meta_model}")

    stacking_regressor = StackingRegressor(estimators=base_models2,
final_estimator=meta_model)
    stacking_regressor.fit(X_train_all_2, y_train_all)

    y_pred2 = stacking_regressor.predict(X_test_all_2)
    metrics = print_evaluation_metrics(y_test, y_pred2)
    print("\n", "\n")
```

Συνδυασμός Μοντέλων με Χρήση Stacking Regressor

Technique (Scaler) (meta-model)	MSE	RMSE	MAE	MAPE
Stacking Regressor (MinMaxScaler) (Ridge)	6.2098	2.4919	1.5164	1.8129
Stacking Regressor (MinMaxScaler) (Lasso)	6.1198	2.4738	1.4952	1.7890
Stacking Regressor (MinMaxScaler) (ElasticNet)	6.1461	2.4791	1.5006	1.7938
Stacking Regressor (MinMaxScaler) (SVR)	5.2758	2.2969	1.3105	1.5785
Stacking Regressor (MinMaxScaler) (RandomForestRegressor)	10.3033	3.2099	1.6368	1.9671
Stacking Regressor (MinMaxScaler) (XGBRegressor)	9.8109	3.1322	1.7246	2.0970
Stacking Regressor (StandardScaler) (Ridge)	6.1507	2.4801	1.5361	1.8322
Stacking Regressor (StandardScaler) (Lasso)	6.0507	2.4598	1.5030	1.7937
Stacking Regressor (StandardScaler) (ElasticNet)	6.0740	2.4645	1.5055	1.7959
Stacking Regressor (StandardScaler) (SVR)	5.2250	2.2858	1.3215	1.5971
Stacking Regressor (StandardScaler) (RandomForestRegressor)	10.9405	3.3076	1.6455	1.9866
Stacking Regressor (StandardScaler) (XGBRegressor)	11.7722	3.4311	1.7329	2.1595

Πίν.6: Αποτελέσματα Αξιολόγησης του Stacking Regressor με Διαφορετικά Μετα-Μοντέλα



Σχ.6: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης του Stacking Regressor με Διαφορετικά Μετα-Μοντέλα

Μοντέλο για τον υπολογισμό δείκτη ασφαλούς οδηγικής συμπεριφοράς από δεδομένα οδήγησης

Η τεχνική Stacking αποτελεί μια προηγμένη μέθοδο συνδυασμού πολλαπλών μοντέλων (μοντέλα βάσης) με τη βοήθεια ενός μετα-μοντέλου, το οποίο εκπαιδεύεται για να συνδυάζει τις προβλέψεις των μοντέλων βάσης και να παράγει μια τελική πρόβλεψη. Η προσέγγιση αυτή αξιοποιεί τα πλεονεκτήματα των διαφόρων μοντέλων για τη βελτίωση της συνολικής απόδοσης.

Στο πλαίσιο αυτής της ανάλυσης, εκπαιδεύτηκαν μοντέλα XGBoost ξεχωριστά για κάθε οδηγό ως μοντέλα βάσης. Στη συνέχεια, για την εκτέλεση της μεθόδου Stacking, χρησιμοποιήθηκαν διάφορα μετα-μοντέλα (Ridge Regression, Lasso Regression, ElasticNet, Support Vector Regressor (SVR), RandomForest Regressor, XGBoost Regressor). Η διαδικασία εκτελέστηκε δύο φορές, χρησιμοποιώντας δύο διαφορετικά σύνολα δεδομένων τα οποία προέκυψαν από δύο διαφορετικές μεθόδους κανονικοποίησης (MinMaxScaler και StandardScaler). Για κάθε μετα-μοντέλο, δημιουργήθηκε και εκπαιδεύτηκε ένα Stacking Regressor, το οποίο συνδύασε τις προβλέψεις των μοντέλων βάσης και παρήγαγε την τελική πρόβλεψη.

Αποτελέσματα και Συμπεράσματα

Η τεχνική Stacking αποδείχθηκε ιδιαίτερα αποτελεσματική, καθώς αξιοποίησε τα πλεονεκτήματα των μοντέλων βάσης και βελτίωσε την ακρίβεια των προβλέψεων. Η επιλογή διαφορετικών μετα-μοντέλων επέτρεψε την εύρεση των βέλτιστων συνδυασμών για την τελική πρόβλεψη.

Από τα αποτελέσματα προέκυψε ότι το Support Vector Regressor (SVR) ως μετα-μοντέλο παρουσίασε την καλύτερη απόδοση, επιτυγχάνοντας τις χαμηλότερες τιμές σφαλμάτων (MSE, RMSE, MAE, MAPE). Οι διαφορές μεταξύ των δύο μεθόδων κανονικοποίησης (MinMaxScaler και StandardScaler) ήταν ελάχιστες, με τα δύο σύνολα δεδομένων να παρέχουν συγκρίσιμα αποτελέσματα.

3.3.44 Συνδυασμός Μοντέλων με Χρήση Bagging Regressor

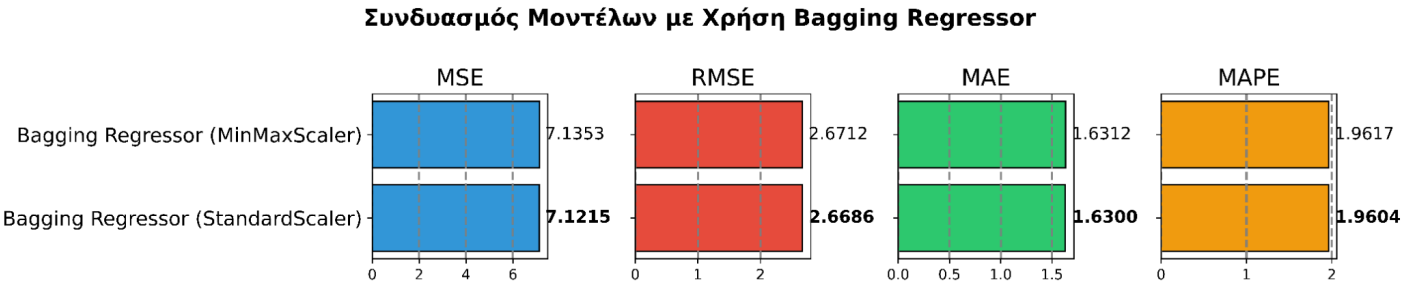
```
bagging_regressor = BaggingRegressor(estimator=xgb_model2_7_1,
n_estimators=250, random_state=42)
bagging_regressor.fit(X_train_all_1, y_train_all)

y_pred1 = bagging_regressor.predict(X_test_all_1)
metrics = print_evaluation_metrics(y_test, y_pred1)
```

```
bagging_regressor2 = BaggingRegressor(estimator=xgb_model2_7_2,
n_estimators=250, random_state=42)
bagging_regressor2.fit(X_train_all_2, y_train_all)

y_pred2 = bagging_regressor2.predict(X_test_all_2)
metrics = print_evaluation_metrics(y_test, y_pred2)
```

Συνδυασμός Μοντέλων με Χρήση Bagging Regressor

Technique (Scaler)	MSE	RMSE	MAE	MAPE
Bagging Regressor (MinMaxScaler)	7.1353	2.6712	1.6312	1.9617
Bagging Regressor (StandardScaler)	7.1215	2.6686	1.6300	1.9604

Πίν.7: Αποτελέσματα Αξιολόγησης Bagging Regressor με Διαφορετικές Τεχνικές Κλιμάκωσης



Σχ.7: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης Bagging Regressor με Διαφορετικές Τεχνικές Κλιμάκωσης

Στην προσπάθεια βελτίωσης της ακρίβειας των προβλέψεων, εφαρμόστηκε η τεχνική Bagging (Bootstrap Aggregating), η οποία είναι μια ισχυρή μέθοδος συνόλου που μειώνει τη διακύμανση ενός μοντέλου και βελτιώνει την απόδοσή του. Το Bagging επιτυγχάνει αυτόν τον στόχο δημιουργώντας πολλαπλά αντίγραφα του ίδιου αλγορίθμου εκμάθησης, τα οποία εκπαιδεύονται σε διαφορετικά υποσύνολα των δεδομένων.

Διαδικασία Εκπαίδευσης και Αξιολόγησης

- **Υλοποίηση Bagging Regressor**:
    - Δημιουργούμε δύο Bagging Regressors για να αξιολογήσουμε την απόδοση των μοντέλων μας. Ο πρώτος Bagging Regressor χρησιμοποιεί το μοντέλο xgb_model2_7_1 ως βάση του εκτιμητή, ενώ ο δεύτερος χρησιμοποιεί το μοντέλο xgb_model2_7_2.
    - Και για τους δύο Regressors, καθορίζουμε τον αριθμό των εκτιμητών (n_estimators) σε 250, που σημαίνει ότι θα δημιουργήσουμε 250 αντίγραφα του βασικού μοντέλου, και ορίζουμε μια σταθερή τιμή για τον τυχαίο αριθμό σπόρου (random_state=42) για την επαναληψιμότητα των αποτελεσμάτων.
- **Εκπαίδευση του Μοντέλου**:
    - Οι Bagging Regressors εκπαιδεύονται στο πλήρες εκπαιδευτικό σύνολο δεδομένων (X_train_all_1 και X_train_all_2), χρησιμοποιώντας τις αντίστοιχες πραγματικές τιμές στόχου (y_train_all). Κατά την εκπαίδευση, κάθε μοντέλο μαθαίνει από διαφορετικά υποσύνολα των δεδομένων, με τη μέθοδο Bootstrap.
- **Προβλέψεις και Αξιολόγηση**:
    - Μετά την ολοκλήρωση της εκπαίδευσης, οι Bagging Regressors εφαρμόζονται στο σύνολο δοκιμαστικών δεδομένων (X_test_all_1 και X_test_all_2) για να προβλέψουν τις τιμές στόχου.
    - Για την αξιολόγηση της απόδοσης των μοντέλων, χρησιμοποιούμε τις μετρικές MSE (Mean Squared Error), RMSE (Root Mean Squared Error), MAE (Mean Absolute Error), και MAPE (Mean Absolute Percentage Error). Αυτές οι μετρικές μας επιτρέπουν να κατανοήσουμε τη συνολική ακρίβεια και την αξιοπιστία των προβλέψεων των μοντέλων μας.

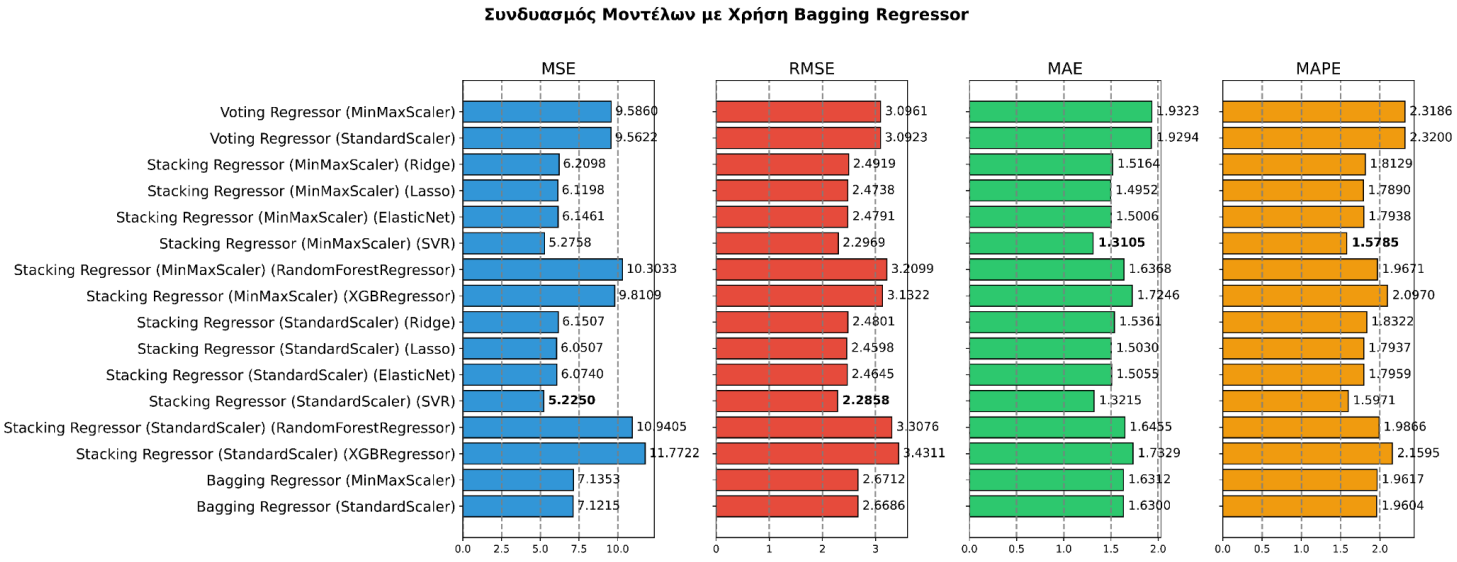
Με τη χρήση της τεχνικής Bagging, καταφέραμε να ενισχύσουμε την απόδοση του βασικού μας μοντέλου, μειώνοντας τη διακύμανση και παρέχοντας μια πιο σταθερή και αξιόπιστη πρόβλεψη.

3.3.45 Αποτελέσματα Αξιολόγησης Τεχνικών Συνδυασμού Μοντέλων και Συμπεράσματα

Συνδυασμός Μοντέλων

Technique (Scaler)	MSE	RMSE	MAE	MAPE
Voting Regressor (MinMaxScaler)	9.5860	3.0961	1.9323	2.3186
Voting Regressor (StandardScaler)	9.5622	3.0923	1.9294	2.3200
Stacking Regressor (MinMaxScaler) (Ridge)	6.2098	2.4919	1.5164	1.8129
Stacking Regressor (MinMaxScaler) (Lasso)	6.1198	2.4738	1.4952	1.7890
Stacking Regressor (MinMaxScaler) (ElasticNet)	6.1461	2.4791	1.5006	1.7938
Stacking Regressor (MinMaxScaler) (SVR)	5.2758	2.2969	1.3105	1.5785
Stacking Regressor (MinMaxScaler) (RandomForestRegressor)	10.3033	3.2099	1.6368	1.9671
Stacking Regressor (MinMaxScaler) (XGBRegressor)	9.8109	3.1322	1.7246	2.0970
Stacking Regressor (StandardScaler) (Ridge)	6.1507	2.4801	1.5361	1.8322
Stacking Regressor (StandardScaler) (Lasso)	6.0507	2.4598	1.5030	1.7937
Stacking Regressor (StandardScaler) (ElasticNet)	6.0740	2.4645	1.5055	1.7959
Stacking Regressor (StandardScaler) (SVR)	5.2250	2.2858	1.3215	1.5971
Stacking Regressor (StandardScaler) (RandomForestRegressor)	10.9405	3.3076	1.6455	1.9866
Stacking Regressor (StandardScaler) (XGBRegressor)	11.7722	3.4311	1.7329	2.1595
Bagging Regressor (MinMaxScaler)	7.1353	2.6712	1.6312	1.9617
Bagging Regressor (StandardScaler)	7.1215	2.6686	1.6300	1.9604

Πίν.8: Συγκριτική Αξιολόγηση Διαφορετικών Τεχνικών Συνδυασμού Μοντέλων με Διάφορες Τεχνικές Κλιμάκωσης



Σχ.8: Οπτική Παρουσίαση Αποτελεσμάτων Αξιολόγησης Διαφορετικών Τεχνικών Συνδυασμού Μοντέλων με Διάφορες Τεχνικές Κλιμάκωσης

Συμπεράσματα από την Ανάλυση Συνδυασμών Μοντέλων

Στην παρούσα μελέτη, εξετάστηκε η απόδοση διαφόρων τεχνικών συνδυασμού μοντέλων (ensemble techniques) με τη χρήση διαφορετικών scalers για την πρόβλεψη δεδομένων. Ο πίνακας αποτελεσμάτων αξιολογεί την απόδοση κάθε τεχνικής με βάση τέσσερις μετρικές σφάλματος:

Τα βασικά συμπεράσματα της ανάλυσης συνοψίζονται ως εξής:

- **Βέλτιστη Απόδοση με Βάση το MSE και το RMSE:** Ο συνδυασμός "Stacking Regressor" με "SVR" και χρήση του "StandardScaler" εμφάνισε τη χαμηλότερη τιμή MSE (5.2250), γεγονός που υποδεικνύει ότι αυτή η τεχνική επιτυγχάνει τον μικρότερο μέσο τετραγωνικό σφάλμα από όλα τα εξεταζόμενα μοντέλα. Επιπλέον, η ίδια τεχνική παρουσίασε και τη χαμηλότερη τιμή RMSE (2.2858), δείχνοντας την υψηλότερη ακρίβεια ως προς τη ρίζα του μέσου τετραγωνικού σφάλματος. Αυτό σημαίνει ότι οι προβλέψεις της τεχνικής αυτής βρίσκονται πιο κοντά στις πραγματικές τιμές, με τη μικρότερη δυνατή απόκλιση.
- **Καλύτερη Τεχνική για MAE:** Ο συνδυασμός "Stacking Regressor" με "SVR" και χρήση του "MinMaxScaler" είχε την καλύτερη απόδοση στο MAE (1.3105), υποδεικνύοντας ότι αυτή η διαμόρφωση παρέχει τις πιο ακριβείς προβλέψεις με τη μικρότερη μέση απόλυτη απόκλιση από τις πραγματικές τιμές.
- **Ελαχιστοποίηση του MAPE:** Ο συνδυασμός "Stacking Regressor" με "SVR" και χρήση του "MinMaxScaler" καταγράφει τη χαμηλότερη τιμή MAPE (1.5785), δείχνοντας την υψηλότερη ακρίβεια όσον αφορά το ποσοστό σφάλματος στις προβλέψεις.
- **Συγκριτική Ανάλυση Διαφορετικών Scalers:** Οι τεχνικές που χρησιμοποιούν τον "StandardScaler" υπερέχουν κυρίως στις μετρικές MSE και RMSE, ενώ ο "MinMaxScaler" επιδεικνύει καλύτερη απόδοση στις μετρικές MAE και MAPE. Αυτό υποδηλώνει ότι η επιλογή του κατάλληλου scaler μπορεί να επηρεάσει σημαντικά την απόδοση του μοντέλου, ανάλογα με τη μετρική σφάλματος που είναι πιο κρίσιμη για την εφαρμογή.
- **Απόδοση των Bagging Regressors**: Οι "Bagging Regressors" καταγράφουν υψηλότερες τιμές σφάλματος σε όλες τις μετρικές σε σχέση με τους "Stacking Regressors", γεγονός που υποδηλώνει ότι οι "Stacking Regressors" είναι πιο

αποτελεσματικοί στην ενσωμάτωση διαφορετικών μοντέλων για τη συγκεκριμένη ανάλυση.

- **Συμπεράσματα για την Επιλογή Μοντέλων**: Για εφαρμογές που απαιτούν υψηλή ακρίβεια και χαμηλό σφάλμα στις προβλέψεις, ο “Stacking Regressor” με “SVR” είναι η προτιμώμενη επιλογή. Η χρήση του “StandardScaler” προσφέρει καλύτερη απόδοση όσον αφορά το MSE και το RMSE, ενώ ο “MinMaxScaler” επιδεικνύει καλύτερη απόδοση στις μετρικές MAE και MAPE.

Τα παραπάνω συμπεράσματα προσφέρουν σημαντική κατεύθυνση για την επιλογή της κατάλληλης τεχνικής συνδυασμού μοντέλων και των παραμέτρων τους σε μελλοντικές εφαρμογές, ανάλογα με τις απαιτήσεις ακρίβειας και τις μετρικές σφάλματος που είναι πιο κρίσιμες για την εκάστοτε περίπτωση.

ΚΕΦ.4: Συμπεράσματα και Μελλοντικές Εργασίες

4.1 Περίληψη Ευρημάτων

Η ανάλυση έδειξε ότι το Gradient Boosting ήταν το πιο αποδοτικό μοντέλο για την πρόβλεψη της ασφαλούς οδηγικής συμπεριφοράς, υπερέχοντας έναντι των νευρωνικών δικτύων σε όρους ακρίβειας και σταθερότητας. Η χρήση του Stacking Regressor με SVR ως meta-model προσέφερε την υψηλότερη προγνωστική ακρίβεια, αναδεικνύοντας τη σημασία της συνδυαστικής χρήσης μοντέλων για την αντιμετώπιση σύνθετων προβλημάτων πρόβλεψης.

4.2 Σύγκριση με τη Βιβλιογραφία

Τα ευρήματά μας συμφωνούν με τις πρόσφατες έρευνες των Elyoussoufi, Walker, Black και DeGirolamo (2023) και Castignani, Derrmann, Frank, και Engel (2015), οι οποίες επίσης δείχνουν ότι η ενσωμάτωση πολλαπλών πηγών δεδομένων και η χρήση σύνθετων τεχνικών μοντελοποίησης μπορεί να βελτιώσει σημαντικά την πρόβλεψη της οδηγικής συμπεριφοράς.

4.3 Ερμηνεία των Αποτελεσμάτων

Η υπεροχή του Gradient Boosting μπορεί να αποδοθεί στην ικανότητά του να χειρίζεται καλά μη γραμμικά πρότυπα και να συνδυάζει την απόδοση πολλαπλών απλών μοντέλων για τη δημιουργία ενός ισχυρότερου συνόλου. Η προσέγγιση Stacking με SVR ως meta-model αποδείχθηκε ιδιαίτερα αποτελεσματική καθώς εκμεταλλεύεται τις συμπληρωματικές δυναμικές διαφορετικών μοντέλων.

4.4 Περιορισμοί της Μελέτης

Παρόλο που τα ευρήματα είναι ενθαρρυντικά, η μελέτη περιορίστηκε από τον μικρό αριθμό οδηγών που συμμετείχαν (7 οδηγοί) και την περιορισμένη γεωγραφική περιοχή των δεδομένων. Επίσης, η χρήση δεδομένων μόνο από το OpenWeatherMap API περιορίζει την ποικιλία των περιβαλλοντικών παραγόντων που θα μπορούσαν να εξεταστούν.

4.5 Προτάσεις για Μελλοντική Έρευνα

Μελλοντικές μελέτες θα μπορούσαν να επεκτείνουν τα ευρήματα της παρούσας έρευνας χρησιμοποιώντας μεγαλύτερα και πιο ποικιλόμορφα σύνολα δεδομένων. Επιπλέον, η ενσωμάτωση άλλων τεχνικών μηχανικής μάθησης και η διερεύνηση της επίδρασης άλλων περιβαλλοντικών παραγόντων, θα μπορούσαν να προσφέρουν νέες προοπτικές για την περαιτέρω βελτίωση των προγνωστικών μοντέλων.

Επίλογος

Η παρούσα πτυχιακή εργασία ανέδειξε την αξία της ενσωμάτωσης προηγμένων τεχνικών μηχανικής μάθησης στην ανάλυση της οδηγικής συμπεριφοράς, με στόχο την προαγωγή της οδικής ασφάλειας. Αναπτύχθηκε ένα καινοτόμο μοντέλο πρόβλεψης το οποίο βασίστηκε σε δεδομένα οδήγησης και περιβαλλοντικούς παράγοντες, με σκοπό την ακριβή και αξιόπιστη πρόβλεψη δεικτών ασφαλούς οδήγησης. Η εφαρμογή του Gradient Boosting και των τεχνικών συνόλου, σε συνδυασμό με τη βελτιστοποίηση υπερπαραμέτρων μέσω του εργαλείου Optuna, συνέβαλαν καθοριστικά στην αύξηση της προγνωστικής ακρίβειας και της ανθεκτικότητας των μοντέλων.

Τα ευρήματα αυτής της έρευνας προσφέρουν πολύτιμες πληροφορίες για την ανάπτυξη τεχνολογιών που στοχεύουν στην ενίσχυση της οδικής ασφάλειας και την υποστήριξη συστημάτων υποβοήθησης οδηγού. Ειδικότερα, η δυνατότητα δημιουργίας εξατομικευμένων προγνωστικών μοντέλων μπορεί να αποτελέσει το θεμέλιο για τη βελτίωση της οδηγικής συμπεριφοράς σε πραγματικό χρόνο, λαμβάνοντας υπόψη τις μοναδικές συνήθειες κάθε οδηγού και τις εξωτερικές συνθήκες.

Ωστόσο, όπως αναλύθηκε, η έρευνα αυτή αντιμετώπισε ορισμένους περιορισμούς, κυρίως λόγω του μικρού μεγέθους του δείγματος και της περιορισμένης γεωγραφικής και χρονικής κάλυψης των δεδομένων. Για τον λόγο αυτό, προτείνεται η επέκταση της μελέτης με τη χρήση μεγαλύτερων και πιο ποικιλόμορφων συνόλων δεδομένων, καθώς και η διερεύνηση πρόσθετων παραγόντων που μπορεί να επηρεάζουν την οδηγική συμπεριφορά.

Συμπερασματικά, η εργασία αυτή συμβάλλει ουσιαστικά στον τομέα της ανάλυσης της οδηγικής συμπεριφοράς, προσφέροντας ένα ισχυρό πλαίσιο για την ανάπτυξη ασφαλέστερων και πιο αποτελεσματικών συστημάτων οδήγησης. Μελλοντικές έρευνες μπορούν να βασιστούν σε αυτά τα ευρήματα για να εξερευνήσουν νέες προσεγγίσεις και εφαρμογές, ενισχύοντας περαιτέρω την ασφάλεια και την αποδοτικότητα στις μεταφορές.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Castignani, G., Derrmann, T., Frank, R., & Engel, T. (2015). Driver behavior profiling using smartphones: A low-cost platform for driver monitoring. IEEE Intelligent transportation systems magazine, 7(1), 91-102.
- Halim, Z., Kalsoom, R., & Baig, A. R. (2016). Profiling drivers based on driver dependent vehicle driving features. Applied Intelligence, 44, 645-664.
- Elyoussoufi, A., Walker, C. L., Black, A. W., & DeGirolamo, G. J. (2023). The relationships between adverse weather, traffic mobility, and driver behavior. Meteorology, 2(4), 489-508.
- Pisano, P. A., Goodwin, L. C., & Rossetti, M. A. (2008, January). US highway crashes in adverse road weather conditions. In 24th conference on international interactive information and processing systems for meteorology, oceanography and hydrology, New Orleans, LA (pp. 20-24).