ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ
ΤΕΧΝΟΛΟΓΙΑ ΥΛΙΚΟΥ ΚΑΙ ΛΟΓΙΣΜΙΚΟΥ: ΣΧΕΔΙΑΣΗ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΧΥΤΩΝ ΣΥΣΤΗΜΑΤΩΝ ΥΠΟΛΟΓΙΣΜΟΥ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

HOU-CS-PGP-2021-375

«ΜΗΧΑΝΙΣΜΟΙ ΠΡΟΒΛΕΨΗΣ ΚΙΝΗΣΕΩΝ ΣΕ ΔΙΑΧΥΤΑ ΠΕΡΙΒΑΛΛΟΝΤΑ»

ΑΝΤΩΝΙΟΣ ΑΔΡΑΚΤΑΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΠΕΤΡΟΣ ΝΙΚΟΠΟΛΙΤΙΔΗΣ

ΠΑΤΡΑ 2022

HELLENIC OPEN UNIVERSITY
SCHOOL OF SCIENCES AND TECHNOLOGY

Πτυχιακή Εργασία *HOU-CS- PGP-2021-375*

Μηχανισμοί πρόβλεψης κινήσεων σε διάχυτα περιβάλλοντα

Αδρακτάς Αντώνιος

Μηχανισμοί πρόβλεψης κινήσεων σε διάχυτα περιβάλλοντα

Αδρακτάς Αντώνιος

ΝΙΚΟΠΟΛΙΤΙΔΗΣ ΠΕΤΡΟΣ

Τμήμα Πληροφορικής, ΑΠΘ

ΨΑΝΝΗΣ ΚΩΣΤΑΝΤΙΝΟΣ

Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας

Περίληψη: Στην παρούσα εργασία υλοποιήθηκε ένα μοντέλο πρόβλεψης κινητικότητας χρηστών με μηχανική μάθηση που βασίζεται σε έναν αλγόριθμο τύπου *Bayesian Network*. Τα δεδομένα βασίζονται στις δραστηριότητες-τοποθεσίες ατόμων με βάση μια τυπική εργασιακή ζωή κάποιου υπαλλήλου, δηλαδή σπίτι, γραφείο, σε εξωτερική συνάντηση κλπ., καθώς και πληροφορία για το είδος του εκάστοτε υπαλλήλου και αν έχει κάποια επείγουσα προσωπική δουλειά. Σκοπός είναι η πρόβλεψη της επόμενης δραστηριότητας του χρήστη. Παράχθηκαν συνθετικά δεδομένα για χρήση στο μοντέλο, αφού δεν βρέθηκε κατάλληλο δημόσια διαθέσιμο σετ δεδομένων τοποθεσία-δραστηριοτήτων χρηστών. Η υλοποίηση όλων των διεργασιών έγινε σε γλώσσα Python. Η απόδοση του μοντέλου αξιολογήθηκε με διάφορα μετρικά στοιχεία τόσο συνολικά όσο και στις επιμέρους καταστάσεις της κλάσης πρόβλεψης. Συγκρίθηκαν τα αποτελέσματα με ή χωρίς την πληροφορία πλαισίου του χρήστη, όπου αναδείχθηκε η αξία της επιπλέον πληροφορίας εισόδου. Ακόμα, για την εξαγωγή αντικειμενικότερης ένδειξης της ακρίβειας του μοντέλου έγιναν δοκιμές με δεδομένα δοκιμής με ίδιο αριθμό δειγμάτων στις διαφορετικές καταστάσεις της κλάσης πρόβλεψης, όπου το μοντέλο είχε ακρίβεια 56%. Τέλος, έγινε σύγκριση του υλοποιημένου μοντέλου με άλλους γνωστούς αλγορίθμους όπου δεν βρέθηκε κάποιος αλγόριθμος με καλύτερη απόδοση στο συγκεκριμένο πρόβλημα. Σ' αυτήν την σύγκριση βρέθηκε ένα μοντέλο που βασίζεται σε αλγόριθμο Decision Trees να εμφανίζει πανομοιότυπα αποτελέσματα.

Λέξεις-κλειδιά: machine learning, Bayesian networks, mobility predictions, Python, model evaluation, synthetic data

Περιεχόμενο: κείμενο, πρόγραμμα σε γλώσσα Python, βιβλιογραφική έρευνα, Πίνακες, Εικόνες

Mobility prediction scheme in ubiquitous environments

Antonios Adraktas

Petros Nikopolitidis

Department of Informatics, Aristotle University of Thessaloniki

Konstantinos Psannis

Department of Applied Informatics, University of Macedonia

Abstract: In this thesis a mobility prediction machine learning scheme was developed based on a Bayesian Network algorithm. Input data are based on the activities-locations of users from typical work life examples of a private company employee, such as home, office, head office, outside meeting. They also include contextual information about the employee type (high-level, common, marketing) and whether users have household work. Goal is to predict the next state of each user based on the current state and the context data. Synthetic data were produced for usage in the model as no suitable publicly available data set with user locations-activities was found. Development of all processes was made in Python. Performance evaluation of the model was assessed using different metrics both as a whole with accuracy and in each different state of the target class with precision, recall and f1-score. Comparison with and without the usage of the context data were made where the importance of context data for better predictions was verified. To obtain a better accuracy metric that assess all states equally comparison was made with a test set with balanced and non-balanced samples in the different states in the target class. In the balanced case accuracy was 56% while in the non-balanced it was 70%, as states with worse predictions had fewer representation in the data. Finally, comparison of performance of the developed model against other commonly used algorithms in classification problems was performed, where no other model was found with better performance in the given problem. In this comparison, a Decision Tree algorithm had identical results and performance with our model.

Περιεχόμενα

1. Εισαγωγή

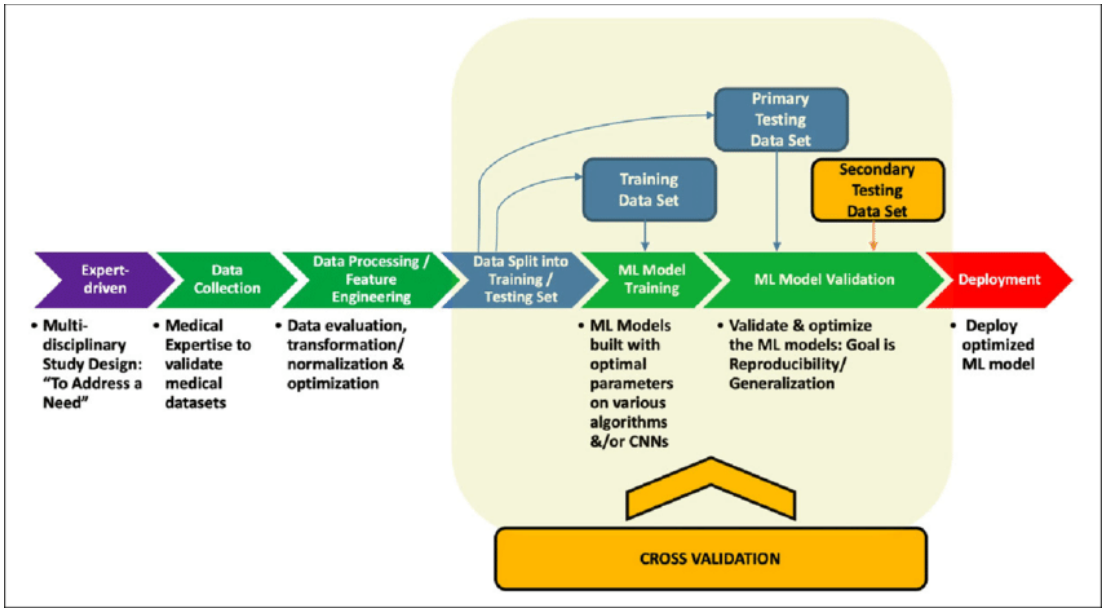
Η πληροφορία πλαισίου του ατόμου είναι δομικό στοιχείο των εφαρμογών διάχυτου υπολογισμού, αφού με βάση αυτή (τοποθεσία, δραστηριότητα, προτιμήσεις) μπορούν να προσφέρονται διαφορετικές υπηρεσίες από τις εκάστοτε εφαρμογές. Τα έξυπνα κινητά ή άλλες έξυπνες συσκευές (έξυπνο ρολόι) μπορούν να συλλέξουν πολλές πληροφορίες με τους αισθητήρες τους (GPS, επιταχυνσιόμετρο) και να εξάγουν πληροφορίες για το χρήστη. Αυτές οι συσκευές τις κουβαλούν οι χρήστες συνεχώς οπότε μπορεί να συλλεχθεί μεγάλος όγκος πληροφορίας [1]. Η δυνατότητα πρόβλεψης της επόμενης κίνησης του χρήστη είναι κάτι που μπορεί να προσφέρει νέες υπηρεσίες στους χρήστες και γι' αυτό συγκεντρώνει μεγάλο ενδιαφέρον απ' τις εταιρίες λογισμικού. Σχετικές λειτουργίες έχουν αρχίσει ήδη να υλοποιούνται στα συστήματα της αγοράς, πχ στο ios operating system το smartphone μπορεί να παρέχει πληροφορίες στις εφαρμογές για επικείμενη είσοδο σε συγκεκριμένη περιοχή.

Αντικείμενο της παρούσας εργασίας είναι ένας μηχανισμός προβλέψεων κινήσεων ανθρώπων σε διάχυτα περιβάλλοντα. Πιο συγκεκριμένα, αφορά την πρόβλεψη της επόμενης τοποθεσίας/δραστηριότητας ενός υπαλλήλου γραφείου έναν αλγόριθμο επιβλεπόμενης μηχανικής μάθησης (supervised machine learning) ο οποίος τροφοδοτείται με ιστορικά δεδομένα των χρηστών που αφορούν τις προηγούμενες τοποθεσίες τους και κάποια δεδομένα σε σχέση με την ταυτότητα/κατάσταση του εκάστοτε χρήστη. Η μελέτη αυτή βασίστηκε στην έρευνα των Garg, Neeraj, et al [2] σε σχέση με την καταγραφή των ανθρώπινων δραστηριοτήτων και τις πιθανότητες μεταβάσεων απ' την μία κατάσταση στην άλλη.

Η συγκεκριμένη εργασία αφορά τις καθημερινές δραστηριότητες ενός ιδιωτικού υπαλλήλου όπως «βρίσκομαι σπίτι», «πηγαίνω στο γραφείο», «πηγαίνω στα κεντρικά γραφεία», «σε εξωτερικό ραντεβού» κλπ. , όπου το άτομο μπορεί να μεταβαίνει απ' την μία κατάσταση στην άλλη με βάση κάποιες πιθανότητες. Οι πιθανότητες αυτές μπορεί να αλλάζουν σύμφωνα με κάποια δεδομένα σχετικά με το άτομο όπως το είδος υπαλλήλου (στέλεχος, marketing, άλλο) ή το αν είναι αδιάθετος. Σκοπός της εργασίας είναι η ανάπτυξη ενός αλγορίθμου πρόβλεψης της επόμενης δραστηριότητας με βάση ιστορικά δεδομένα απ' τους χρήστες.

Ο σχεδιασμός και η υλοποίηση ενός μηχανισμού πρόβλεψης κινήσεων περιλαμβάνει αρκετά βήματα όπως την συγκέντρωση των δεδομένων από κάποια πηγή, την επεξεργασία τους ώστε να έρθουν σε κατάλληλη μορφή, την επιλογή και υλοποίηση ενός αλγορίθμου πρόβλεψης που θα εκπαιδευθεί με τα δεδομένα, την εξαγωγή αποτελεσμάτων και αξιολόγηση τους. Αυτή η διαδικασία μπορεί να επαναληφθεί πολλές φορές για την βελτιστοποίηση του μηχανισμού ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα. Καθένα απ' τα παραπάνω βήματα αποτελούν σημαντικά βήματα τα οποία πολλές φορές είναι ανεξάρτητοι τομείς έρευνας και μελέτης, λόγω της πληθώρας των τεχνικών και γνώσεων που απαιτούνται στο καθένα. Τα βήματα αυτά απεικονίζονται με περισσότερες λεπτομέρειες στο Σχ. 1 που πάρθηκε απ' την εργασία των Rashidi et al. [3] και είναι όμοια σε όλες τις περιπτώσεις αλγορίθμων μηχανικής μάθησης με επίβλεψη.

Η δομή της εργασίας είναι η ακόλουθη. Στην ενότητα δύο γίνεται επισκόπηση της βιβλιογραφίας για τις μεθόδους αντιμετώπισης αντίστοιχων προβλημάτων για όλα τα παραπάνω βασικά βήματα της διαδικασίας. Στην ενότητα τρία γίνεται εκτενής παρουσίαση του συγκεκριμένου προβλήματος και της υλοποίησης που έγινε για την αντιμετώπιση της συγκέντρωσης και επεξεργασίας των δεδομένων και της υλοποίησης του αλγορίθμου πρόβλεψης. Στην ενότητα τέσσερα παρουσιάζεται η διαδικασία εξαγωγής αποτελεσμάτων και αξιολόγηση τους. Στην ενότητα πέντε γίνεται σύγκριση απόδοσης της συγκεκριμένης υλοποίησης με άλλους αλγορίθμους που χρησιμοποιούνται σε προβλήματα μηχανικής μάθησης. Στην ενότητα έξι δίνονται τα συμπεράσματα της εργασίας και οι τομείς μελλοντικής μελέτης και βελτίωσης. Στο τέλος της εργασίας υπάρχουν οι βιβλιογραφικές αναφορές και τα παραρτήματα με επιπλέον πληροφορίες. Στο παράρτημα Α υπάρχει ο πηγαίος κώδικας της υλοποίησης ο οποίος έγινε σε γλώσσα Python ο οποίος είναι επίσης διαθέσιμος στο δημόσιο αποθετήριο στον ακόλουθο σύνδεσμο:
https://github.com/antonis-adraktas/mobility_predictions



Σχήμα 1: Βήματα μελέτης και σχεδιασμού για αλγορίθμους μηχανικής μάθησης με επίβλεψη από [3]

2. Επισκόπηση των υπαρχόντων μεθόδων πρόβλεψης κινητικότητας

Το θέμα της αναγνώρισης/πρόβλεψης κινητικότητας αλλά και δραστηριοτήτων ατόμων είναι ένα θέμα που έχει μελετηθεί από πολλούς ερευνητές καθώς μπορεί να έχει πολλές εφαρμογές στο πεδίο των κινητών έξυπνων συσκευών. Η επισκόπηση της βιβλιογραφικής έρευνας του πεδίου θα γίνει σ' αυτή την ενότητα χωρίζοντας την σε αντίστοιχες υπο-ενότητες των βασικών βημάτων που αναφέρθηκαν στην εισαγωγή.

2.1 Συγκέντρωση και επεξεργασία δεδομένων

Η συγκέντρωση δεδομένων είναι μία απ' τις κυρίαρχες προκλήσεις στο πεδίο της μηχανικής μάθησης τα τελευταία χρόνια [4]. Αυτό συμβαίνει γιατί σε πολλά προβλήματα υπάρχει έλλειψη δεδομένων με ετικέτα (labeled data) ιδιαίτερα σε νέες εφαρμογές. Επιπλέον, σε πολλά προβλήματα τα δεδομένα μπορεί να μην είναι προσβάσιμα στους ερευνητές, να περιέχουν προσωπικά/ευαίσθητα δεδομένα ή να έχουν μεγάλο κόστος να αποκτηθούν. Σε τέτοιες περιπτώσεις χρησιμοποιούνται συχνά συνθετικά δεδομένα καθώς έχουν μικρό κόστος (ή μηδαμινό) και προσφέρουν ευελιξία καθώς μπορούν να καθοριστούν απ' τους ίδιους τους χρήστες [4]. Υπάρχουν, βέβαια μειονεκτήματα καθώς τα συνθετικά δεδομένα μπορεί να μην ανταποκρίνονται στις συνθήκες του πραγματικού προβλήματος και να οδηγήσουν σε λάθος σχεδιασμό του συστήματος. Σε κάθε περίπτωση η συγκέντρωση των δεδομένων που θα χρησιμοποιηθούν είναι απ' τα πιο σημαντικά ζητήματα στη σχεδίαση ενός συστήματος μηχανικής μάθησης.

Υπάρχουν πολλά δημόσια αποθετήρια με δεδομένα που είναι δημόσια και μπορούν να χρησιμοποιήσουν και να πειραματιστούν οι ερευνητές ελεύθερα. Ένας απ' τους πιο δημοφιλείς ιστότοπους είναι το https://www.kaggle.com/

Μετά απ' την αρχική συγκέντρωση δεδομένων απαιτείται η επεξεργασία τους ώστε να έρθουν σε μορφή που μπορούν να χρησιμοποιηθούν απ' τους αλγορίθμους εκμάθησης και πρόβλεψης. Αυτό περιλαμβάνει τον διαχωρισμό και απόρριψη δεδομένων που δεν είναι χρήσιμα για το μοντέλο, την επεξεργασία στηλών σε κατάλληλη μορφή. Για παράδειγμα, πολλοί αλγόριθμοι δεν δουλεύουν με δεδομένα σε μορφή γραμματοσειράς (string) οπότε κάποιες περιγραφικές κατηγορίες πρέπει να μετατραπούν σε αριθμητικά δεδομένα.

Στη συνέχεια αυτής της ενότητας θα παρουσιαστούν οι κύριες τεχνικές για συγκέντρωση και επεξεργασία δεδομένων που χρησιμοποιούνται στα προβλήματα αναγνώρισης κινητικότητας ατόμων. Κεντρικό ρόλο στην συγκέντρωση δεδομένων για την κινητικότητα χρηστών παίζει τα έξυπνα κινητά τηλέφωνα καθώς είναι συσκευές που μπορούν να καταγράφουν την τοποθεσία των χρηστών με διάφορους τρόπους και ακρίβειες αλλά και να καταγράφουν κλήσεις , μηνύματα και χρήση εφαρμογών απ' όπου μπορούν να εξαχθούν επιπλέον πληροφορίες πλαισίου των χρηστών.

Στην εργασία των Vincent Etter et al. [5] τα δεδομένα συγκεντρώθηκαν από 170 χρήστες όπου καταγράφηκαν τοποθεσίες, κλήσεις, γραπτά μηνύματα και χρήση εφαρμογών των έξυπνων τηλεφώνων τους για περίοδο από μερικές βδομάδες έως δύο χρόνια. Στη συγκέντρωση τέτοιων δεδομένων είναι πολύ σημαντικό να ληφθεί υπόψιν η ιδιωτικότητα των χρηστών. Σε αυτήν την περίπτωση, τα αρχικά δεδομένα κρυπτογραφήθηκαν και οι πραγματικές τοποθεσίες δεν δημοσιεύτηκαν αλλά αντιστοιχήθηκαν σε κύκλους ακτίνας 100m. Οι τοποθεσίες αυτές στην συνέχεια αντιστοιχήθηκαν με αναγνωριστικά τα οποία είναι εν τέλει διαθέσιμα στους χρήστες. Έτσι για το κομμάτι της πρόβλεψης τοποθεσίας οι χρήστες των δεδομένων έχουν τα αναγνωριστικά της τοποθεσίας με την διάρκεια παραμονής σε αυτά. Η επεξεργασία αυτή προστατεύει τα προσωπικά δεδομένα αλλά και τα φέρνει σε μορφή που μπορούν να χρησιμοποιηθούν απ' τους αλγορίθμους πρόβλεψης.

Στην έρευνα των PeixiaoWang et al. [6] που αφορά την πρόβλεψη της επόμενης τοποθεσίας σε περιβάλλον εμπορικού κέντρου (εσωτερικός χώρος) τα δεδομένα τοποθεσίας συλλέγονται απ' τα έξυπνα κινητά με βάση τα σημεία πρόσβασης Wi-fi καθώς ο GPS δέκτης δεν θα είχε καλή απόδοση σε αυτήν την περίπτωση. Από τα ακατέργαστα δεδομένα τοποθεσίας εξάγονται σημεία παραμονής των χρηστών με κάποιο αναγνωριστικό και

δημιουργούνται οι αλληλουχίες μεταβάσεων, οι οποίες μπορούν να χρησιμοποιηθούν για να προβλεφθεί το επόμενο σημείο.

Στην έρευνα των Qiao, Yuanyuan, et al. [7] συγκεντρώνονται δεδομένα τοποθεσίας απ' το δίκτυο LTE (Long Term Evolution-4G) κινητής τηλεφωνίας σε μία πόλη της Νότιας Κίνας με βάση τους σταθμούς βάσης που συνδέονται οι συσκευές. Τα δεδομένα είναι της μορφής <αναγνωριστικό χρήστη, αναγνωριστικό σταθμού, χρόνος> όπου ο σταθμός παίρνεται σαν τοποθεσία του χρήστη. Στην έρευνα των Lv, Qiujian, et al. [8] που χρησιμοποιεί το ίδιο σετ δεδομένων αναλύεται η διαδικασία επεξεργασίας τους ώστε να προκύψουν σημεία ενδιαφέροντος στην πόλη απ' τις τοποθεσίες των σταθμών. Με βάση αυτά τα σημεία φτιάχνονται οι αλληλουχίες μεταβάσεων των χρηστών οι οποίες θα χρησιμοποιηθούν για την πρόβλεψη κινητικότητας.

Απ' τις παραπάνω έρευνες επιβεβαιώνεται ότι η συγκέντρωση δεδομένων τοποθεσίας τα τελευταία χρόνια βασίζεται στα έξυπνα κινητά τηλέφωνα και τις διάφορες δυνατότητες που παρέχουν. Επίσης, σε όλες τις παραπάνω περιπτώσεις τα δεδομένα επεξεργάστηκαν ώστε να προκύψουν σημεία ενδιαφέροντος με κάποιο αναγνωριστικό τα οποία θα χρησιμοποιηθούν σαν καταστάσεις στους μηχανισμούς πρόβλεψης για την επόμενη κατάσταση-τοποθεσία.

Τέλος, εκτός απ' την συγκέντρωση και την επεξεργασία των δεδομένων ένα σημαντικό βήμα είναι η ανάλυση τους ώστε να βρεθούν μοτίβα στις μετακινήσεις. Αυτό μπορεί να βοηθήσει στην βελτιστοποίηση των μηχανισμών πρόβλεψης. Στην έρευνα των Qiao, Yuanyuan, et al. [7] υπάρχει μια τέτοια ποιοτική και ποσοτική ανάλυση των δεδομένων, όπου ανακύπτουν τα χωροχρονικά μοτίβα μετακίνησης.

2.2 Αλγόριθμοί πρόβλεψης κινητικότητας

Το πρόβλημα πρόβλεψης της επόμενης θέσης/δραστηριότητας ατόμων ανάγεται σε ένα πρόβλημα ταξινόμησης (classification) όπου στόχος είναι η εύρεση της επόμενης κατάστασης με βάση ιστορικά δεδομένα, αφού όπως είδαμε στην προηγούμενη ενότητα τα δεδομένα τοποθεσίας επεξεργάζονται ώστε να προκύψουν πεπερασμένες καταστάσεις-σημεία ενδιαφέροντος. Υπάρχουν διάφορα είδη αλγορίθμων που χρησιμοποιούνται για την επίλυση προβλημάτων ταξινόμησης με επιβλεπόμενη μηχανική μάθηση. Στην εργασία των Osisanwo, F. Y., et al. [9] παρουσιάζονται οι κύριες κατηγορίες αυτών των αλγορίθμων που είναι οι *Linear Classifiers, Logistic Regression, Naïve Bayes Network, Support Vector machines, Multi-layer Perceptrons, Decision Trees, Νευρωνικά δίκτυα, Bayesian Networks.* Να σημειωθεί ότι κανένας αλγόριθμος εκμάθησης δεν αποδίδει καλύτερα σε όλα τα είδη δεδομένων αλλά σε κάθε περίπτωση πρέπει να βρεθεί αυτός που αποδίδει καλύτερα για το συγκεκριμένο πρόβλημα.

Στα προβλήματα πρόβλεψης επόμενης δραστηριότητας-τοποθεσίας ατόμων κυριαρχούν οι μέθοδοι στατιστικής πρόβλεψης. Διαισθητικά γνωρίζουμε ότι οι άνθρωποι ακολουθούν μοτίβα στις δραστηριότητες και μετακινήσεις τους οπότε έχοντας πρόσβαση σε ιστορικά δεδομένα μπορούμε να προβλέψουμε την επόμενη κατάσταση βρίσκοντας την πιο συχνή μετάβαση με βάση την τωρινή κατάσταση στα ιστορικά δεδομένα. Τα «Bayesian networks» είναι γραφικό μοντέλο με πιθανοτικές σχέσεις ανάμεσα σε ένα σετ μεταβλητών και είναι απ' τους πιο γνωστούς αλγόριθμους στατιστικής μάθησης [9]. Τα «Bayesian networks» υπολογίζουν την κατανομή πιθανοτήτων μεταβάσεων για κάθε μεταβλητή ανάλογα με τις όποιες άλλες συνθήκες υπάρχουν [2].

Στην εργασία των Vincent Etter et al. [5] χρησιμοποιείται μια μίξη των αλγορίθμων *Dynamical Bayesian Network, Artificial Neural Networks* και *Gradient Boosted Decision Trees.* Οι τρεις αλγόριθμοι έχουν παραπλήσια απόδοση σε όλους τους χρήστες. Καθώς τα μοτίβα των χρηστών είναι διαφορετικά έχουν διαφορετική απόδοση σε κάθε έναν χρήστη ξεχωριστά. Αφού γίνει η εκμάθηση των αλγορίθμων στο ίδιο σετ, διαλέγεται κάθε φορά αυτός που έχει την καλύτερη απόδοση στο δεύτερο σετ ώστε να χρησιμοποιηθεί στο τρίτο.

Στην εργασία των Petzold, Jan, et al. [10] χρησιμοποιείται Bayesian network ως αλγόριθμός πρόβλεψης μετακινήσεων σε εσωτερικό χώρο. Στην εργασία των PeixiaoWang et al. [6] βρίσκεται για κάθε χρήστη το πιο όμοιο μοντέλο απ' τους χρήστες με ιστορικά δεδομένα και διαλέγεται αυτό για να κάνει την πρόβλεψη. Η πρόβλεψη γίνεται με βάση την

πιο συχνή μετάβαση του παραδειγματικού χρήστη για την δεδομένη ακολουθία. Είναι και αυτός ένας αλγόριθμος στατιστικής μάθησης.

Στην εργασία των Lv, Qiujian, et al. [8] χρησιμοποιείται ένας αλγόριθμός κρυφού μοντέλου Markov για την πρόβλεψη της επόμενης τοποθεσίας. Τα μέρη που επισκέπτονται οι χρήστες σχετίζονται με την ώρα της ημέρας π.χ. πιο πιθανό το βράδι να είσαι σπίτι, το πρωί στη δουλειά, οπότε χώρισαν την μέρα σε 48 μέρη της μισής ώρας ώστε να είναι άλλη μία παράμετρος που θα βοηθήσει τον μηχανισμό πρόβλεψης. Το κρυφό μοντέλο Markov είναι ένα στατιστικό μοντέλο όπου το σύστημα θεωρείται ότι είναι μια διαδικασία Markov με μη ορατές καταστάσεις. Στο μοντέλο πρέπει να υπάρχει και μία ορατή κατάσταση η οποία επηρεάζεται απ' τις μη ορατές καταστάσεις με γνωστό τρόπο [11]. Εδώ να σημειωθεί ότι ένα σύστημα HMM (Hidden Markov Model) είναι εφάμιλλο με ένα Dynamic Bayesian Network όπου μόνο μία κατάσταση επιτρέπεται κάθε φορά [2]. Αυτό είναι συμβατό με τα προβλήματα κινητικότητας καθώς ο χρήστης δεν μπορεί να βρίσκεται σε δύο μέρη ταυτόχρονα. Όμοια, θεωρούμε σαν απλοποίηση ότι δεν μπορεί να κάνει δύο δραστηριότητες ταυτόχρονα. Στην εργασία των Qiao, Yuanyuan, et al. [7] προτείνεται ένα υβριδικό μοντέλο Markov για την πρόβλεψη της επόμενης τοποθεσίας σαν βελτίωση αυτού που παρουσιάστηκε στο [8]. Αυτός ο αλγόριθμός λειτουργεί όμοια με τον προηγούμενο πού βρίσκει τα μοτίβα μετακίνησης και παίρνει υπόψιν την ώρα για να κάνει την πρόβλεψη. Στην περίπτωση που δεν μπορεί να κάνει πρόβλεψη ο αλγόριθμός για κάποιον χρήστη με βάση το ιστορικό του τότε βρίσκει τον πιο κοντινό χρήστη με όμοια μοτίβα και κάνει πρόβλεψη με βάση το ιστορικό εκείνου [7]. Έτσι, μειώνει τις περιπτώσεις μη πρόβλεψης και αυξάνει την αποτελεσματικότητα του αλγορίθμου.

Απ' τις παραπάνω περιπτώσεις είναι σαφές ότι στο πεδίο της πρόβλεψης κινητικότητας χρηστών οι αλγόριθμοι στατιστικής πρόβλεψης χρησιμοποιούνται συχνά και θεωρούνται ότι έχουν καλή απόδοση.

2.3 Αξιολόγηση απόδοσης αλγορίθμων προβλέψεων

Κεντρικό ζήτημα για την επιλογή ενός αλγορίθμου για την επίλυση του εκάστοτε ζητήματος είναι η αξιολόγηση του ώστε να επιλέγεται κάθε φορά ο βέλτιστος. Υπάρχουν διάφορες μετρικές μεταβλητές που χρησιμοποιούνται για την αξιολόγηση μοντέλων μηχανικής μάθησης με επίβλεψη σε προβλήματα ταξινόμησης. Στην εργασία των Hossin, Mohammad, και M. N. Sulaiman [12] γίνεται μια επισκόπηση των διαφόρων μετρικών αξιολόγησης της απόδοσης τα οποία φαίνονται στον Πίνακα 1. Ας πάρουμε το παράδειγμα ενός προβλήματος δυαδικής ταξινόμησης όπου το αποτέλεσμα μπορεί να βρίσκεται μόνο σε δύο κλάσεις: την θετική ή την αρνητική. Με *tp (true positive)* σημειώνονται οι αληθινά θετικές προβλέψεις, με *tn (true negative)* σημειώνονται οι αληθινά αρνητικές προβλέψεις, με *fp (false positive)* σημειώνονται οι ψευδώς θετικές προβλέψεις και με *fn (false negative)* σημειώνονται οι ψευδώς αρνητικές προβλέψεις. Τα αποτελέσματα του μηχανισμού προβλέψεων σε σχέση με τα πραγματικά αποτελέσματα απ' το σετ δεδομένων σε ένα πρόβλημα αναπαρίστανται σε έναν πίνακα που ονομάζεται *confusion matrix* και απεικονίζει τις παραπάνω περιπτώσεις. Στο Σχ. 2 φαίνεται ένας τέτοιος πίνακας.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Σχήμα 2: Confusion matrix για πρόβλημα δυαδικής ταξινόμησης από *[13]*

Πίνακας 1: Μετρικές αξιολόγησης για προβλήματα ταξινόμησης από *[12]*

Metrics	Formula	Evaluation Focus
Accuracy (acc)	$\frac{tp+tn}{tp+fp+tn+fn}$	In general, the accuracy metric measures the ratio of correct predictions over the total number of instances evaluated.
Error Rate (err)	$\frac{fp+fn}{tp+fp+tn+fn}$	Misclassification error measures the ratio of incorrect predictions over the total number of instances evaluated.
Sensitivity (sn)	$\frac{tp}{tp+fn}$	This metric is used to measure the fraction of positive patterns that are correctly classified
Specificity (sp)	$\frac{tn}{tn+fp}$	This metric is used to measure the fraction of negative patterns that are correctly classified.
Precision (p)	$\frac{tp}{tp+fp}$	Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class.
Recall (r)	$\frac{tp}{tp+tn}$	Recall is used to measure the fraction of positive patterns that are correctly classified
F-Measure (FM)	$\frac{2*p*r}{p+r}$	This metric represents the harmonic mean between recall and precision values
Geometric-mean (GM)	$\sqrt{tp*tn}$	This metric is used to maximize the *tp* rate and *tn* rate, and simultaneously keeping both rates relatively balanced
Averaged Accuracy	$\frac{\sum_{i=1}^{l}\frac{tp_i+tn_i}{tp_i+fn_i+fp_i+}}{l}$	The average effectiveness of all classes
Averaged Error Rate	$\frac{\sum_{i=1}^{l}\frac{fp_i+fn_i}{tp_i+fn_i+fp_i+}}{l}$	The average error rate of all classes
Averaged Precision	$\frac{\sum_{i=1}^{l}\frac{tp_i}{tp_i+fp_i}}{l}$	The average of per-class precision
Averaged Recall	$\frac{\sum_{i=1}^{l}\frac{tp_i}{tp_i+fn_i}}{l}$	The average of per-class recall
Averaged F-Measure	$\frac{2*p_M*r_M}{p_M+r_M}$	The average of per-class F-measure

Note: - each class of data; tp_i - true positive for C_i; fp_i - false positive for C_i; fn_i – false negative for C_i; tn_i - true negative for C_i; and M macro-averaging.

Η ακρίβεια (*accuracy*) είναι το πιο συχνά χρησιμοποιούμενο μέτρο αξιολόγησης για προβλήματα ταξινόμησης δύο ή και παραπάνω κλάσεων [12]. Η ακρίβεια δείχνει τον αριθμό των σωστών προβλέψεων στον συνολικό αριθμό προβλέψεων. Είναι πολύ χρήσιμη όταν οι κλάσεις-καταστάσεις του προβλήματος είναι ισορροπημένες δηλαδή έχουν παρόμοιο αριθμό δειγμάτων σε κάθε κλάση.

Σε περιπτώσεις όμως που οι κλάσεις δεν είναι ισορροπημένες η ακρίβεια μπορεί να είναι παραπλανητική. Γι' αυτό χρησιμοποιούνται άλλα μέτρα αξιολόγησης. Το *Recall* είναι ο αριθμός των σωστών προβλέψεων σε μία κλάση δια τον συνολικό αριθμό των δειγμάτων σ' αυτήν την κλάση και υπολογίζεται όπως φαίνεται στην εξίσωση (1) για την περίπτωση της θετικής κλάσης σε δυαδική ταξινόμηση.

$$Recall\ (p) = \frac{tp}{tp+fn} \qquad (1)$$

Το *recall* δείχνει την ικανότητα του μοντέλου πρόβλεψης να αναγνωρίσει όλες τις σχετικές περιπτώσεις σε ένα σετ δεδομένων.

Το *Precision* είναι ο αριθμός των σωστών προβλέψεων σε μια κλάση δια τον συνολικό αριθμό των προβλέψεων σ' αυτή την κλάση. Υπολογίζεται απ' την εξίσωση (2) στην περίπτωση δυαδικής ταξινόμησης.

$$Precision\ (p) = \frac{tp}{tp+fp} \qquad (2)$$

Το *precision* δείχνει την ικανότητα του μοντέλου να προβλέπει τα σχετικά δείγματα σε κάθε κλάση. Σαν μείξη των δύο αυτών μετρικών υπάρχει το *F1-score* το οποίο είναι ο αρμονικός μέσος των δύο και φαίνεται στην εξίσωση (3)

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \qquad (3)$$

Επιλέγεται ο αρμονικός μέσος σε σχέση με τον απλό μέσο όρο ώστε να τιμωρηθούν οι ακραίες τιμές. Ανάλογα με το είδος του προβλήματος και το σετ δεδομένων επιλέγονται τα παραπάνω μετρικά για την αξιολόγηση της απόδοσης των μοντέλων. Να σημειωθεί ότι δεν υπάρχουν γενικά καλές τιμές αλλά πάντα εξαρτώνται απ' τα δεδομένα που έχουμε. Για παράδειγμα, αν έχουμε να προβλέψουμε που βρίσκεται ένας χρήστης σε ένα δείγμα μεταξύ δύο τοποθεσιών: «σπίτι», «δουλειά» και έχουμε 100 δείγματα, όπου τα 90 είναι σπίτι. Σε αυτήν την περίπτωση, μια ακρίβεια 90% δεν είναι καλή αφού το ίδιο θα έβρισκε και ένα μοντέλο που απλά προβλέπει «σπίτι» κάθε φορά, ενώ σε ένα άλλο πρόβλημα μπορεί μια ακρίβεια 60% να είναι ικανοποιητική. Επιπλέον, σε κάποια προβλήματα δεν έχουν όλα τα σφάλματα την ίδια βαρύτητα και γι' αυτό εξετάζεται προσεκτικά η κάθε κλάση με όλα τα μέτρα αξιολόγησης που αναφέρθηκαν παραπάνω. Ας πάρουμε για παράδειγμα ένα τεστ ανίχνευσης μιας νόσου όπου σε περίπτωση ανίχνευσης θετικού αποτελέσματος ο ασθενής προβαίνει σε επιπλέον πιο περίπλοκες εξετάσεις που μπορούν να ανιχνεύσουν με βεβαιότητα την ύπαρξη της νόσου. Διαισθητικά, μπορούμε να καταλάβουμε ότι ένα σφάλμα ψευδούς αρνητικού αποτελέσματος έχει μεγαλύτερη βαρύτητα σε σχέση με ένα ψευδός θετικό αφού θα φέρει εφησυχασμό στον ασθενή και δεν θα γίνουν περεταίρω εξετάσεις. Τέτοιες περιπτώσεις έχουν μελετηθεί ευρέως με την χρήση μοντέλων σε διάφορα σετ δεδομένων από διαγνωστικές εξετάσεις όπως στην εργασία των Asri, Hiba, et al [14] για ανίχνευση καρκίνου στο στήθος.

Η παραπάνω ανάλυση βασίστηκε σε προβλήματα δυαδικής ταξινόμησης, όμως αυτά τα μέτρα αξιολόγησης μπορούν να επεκταθούν και σε προβλήματα ταξινόμησης πολλών κλάσεων, όπως φαίνεται και στις τελευταίες γραμμές του πίνακα 1 [12]. Όπως δείξαμε στις προηγούμενες ενότητες η πρόβλεψη κινητικότητας ατόμων ανάγεται σε ένα πρόβλημα ταξινόμησης πολλαπλών καταστάσεων με την κατάλληλη επεξεργασία των δεδομένων.

Στην εργασία των PeixiaoWang et al. [6] αναφέρονται η ακρίβεια και το *precision* σαν μέτρα αξιολόγησης. Χρησιμοποιούνται και τα δύο στην ανάλυση της απόδοσης του μοντέλου, οπού παρουσιάζονται για διαφορετικό αριθμό σημείων παραμονής, δηλαδή την αλληλουχία καταστάσεων του χρήστη. Τα αποτελέσματα είναι καλύτερα όσο μεγαλώνει ο αριθμός των καταστάσεων καθώς όσο πιο λίγες τόσο πιο πολλές επιλογές υπάρχουν απ' τα δεδομένα και έτσι αυξάνεται το σφάλμα. Η ακρίβεια έφτασε το 67,6% για αλληλουχία 5 καταστάσεων, και ήταν 35.1% και 45.5% για 1 και 3 καταστάσεις αντίστοιχα. Επιπλέον, η απόδοση του μοντέλου τους συγκρίνεται με κάποιους άλλους βασικούς αλγορίθμους και βρίσκεται ότι αποδίδει καλύτερα σε αυτό το σετ δεδομένων.

Στην εργασία των Vincent Etter et al. [5] χρησιμοποιείται μόνο η ακρίβεια για την αξιολόγηση του συστήματος και φτάνει στο 56,22%. Και σε αυτήν την περίπτωση γίνεται σύγκριση με κάποιους βασικούς αλγόριθμους. Στην εργασία των Lv, Qiujian, et al. [8] χρησιμοποιείται μόνο η ακρίβεια σαν μέτρο αξιολόγησης. Σε αυτήν την περίπτωση δεν εξάγεται μια συνολική ακρίβεια του συστήματος αλλά αξιολογείται ανάλογα με τους χρήστες, όπου για το 60% των χρηστών έχει πάνω από 80% ακρίβεια. Τα αποτελέσματα του μοντέλου εξάγονται και με βάση διαφορετικά προφίλ χρηστών όπως οικογενειάρχης, ταχυδρόμος, άτομο που διασκεδάζει όπου εύλογα για τους «οικογενειάρχες» το μοντέλο αποδίδει καλύτερα αφού έχουν πιο προβλέψιμα μοτίβα. Τέλος, και εδώ βρίσκεται ότι η ακρίβεια μεγαλώνει όσο μεγαλώνει ο αριθμός καταστάσεων στην αλληλουχίες μετακίνησης.

Στην εργασία των Qiao, Yuanyuan, et al. [7] χρησιμοποιείται η ακρίβεια για την αξιολόγηση του μοντέλου. Παρουσιάζεται η ακρίβεια στα 3 διαφορετικά στάδια του μοντέλου όπου φαίνεται η βελτίωση με την εισαγωγή κάθε σταδίου και φτάνει έως 56.4%. Επιπλέον,

παρουσιάζεται συγκριτικά η ακρίβεια με την χρήση του χρονικού παράγοντα και χωρίς. Τέλος, γίνεται σύγκριση απόδοσης με άλλους διαδεδομένους αλγορίθμους που χρησιμοποιούνται σε τέτοια προβλήματα. Στην εργασία των Petzold, Jan, et al. [10] πάλι η ακρίβεια είναι το μέτρο αξιολόγησης του μοντέλου όπου παρουσιάζονται συγκριτικά τα αποτελέσματα για τα τέσσερα άτομα που υπήρχαν δεδομένα. Επίσης γίνεται σύγκριση ανάλογα με τον αριθμό δωματίων που πρέπει να προβλεφθούν, αν έχει γίνει εκπαίδευση ή όχι και τέλος συγκριτική ανάλυση του μοντέλου *Bayesian network* με μοντέλου *νευρωνικών δικτύων* και έναν *state predictor*. Το προτεινόμενο μοντέλο *Bayesian network* αποδίδει καλύτερα απ' τα άλλα δύο.

Στις εργασίες [10] και [7] γίνεται αναφορά και στο υπολογιστικό κόστος του μοντέλου, καθώς αυτός είναι ένα κρίσιμός παράγοντας που πρέπει να ληφθεί υπόψιν όταν υπάρχει μεγάλος όγκος δεδομένων και απαιτείται το μοντέλο να δουλεύει σε πραγματικό χρόνο σε περιβάλλον παραγωγής. Ιδιαίτερα στο [10] γίνεται αναλυτικός υπολογισμός του υπολογιστικού κόστους του μοντέλου. Καθώς η συγκεκριμένη εργασία είναι πιο παλιά (2005) όπου οι υπολογιστικοί πόροι ήταν σημαντικά μικρότεροι το υπολογιστικό κόστος είχε ακόμα μεγαλύτερη σημασία.

Απ' τα παραπάνω είναι σαφές ότι η ακρίβεια είναι το κυρίαρχο μέτρο αξιολόγησης στα προβλήματα κινητικότητας ατόμων. Παρόλο που το είδος του λάθους σε ένα τέτοιο σύστημα μπορεί να έχει επίπτωση στην εξαγωγή συμπερασμάτων από μία εφαρμογή (π.χ. πρόβλεψη ότι επόμενη τοποθεσία είναι «σπίτι» ενώ στην πραγματικότητα «δείπνο έξω») είναι σαφές ότι δεν έχει την ίδια βαρύτητα με το παράδειγμα ανίχνευσης νόσου που αναφέρθηκε νωρίτερα. Επιπλέον, σε ένα πραγματικό σετ δεδομένων υπάρχουν πολλές διαφορετικές καταστάσεις οπότε δεν είναι εύκολο να βγουν ποιοτικά στοιχεία απ' το είδος του λάθους. Η χρήση της ακρίβειας δεν είναι όμως ένας αριθμός αλλά αξιολογείται το μοντέλο σε πολλές διαφορετικές καταστάσεις (διαφορετικοί χρήστες, χρήση ή όχι επιπλέον δεδομένων σαν είσοδο όπως ο χρόνος , αριθμός καταστάσεων στην αλληλουχία). Τέλος σε όλες τι περιπτώσεις γίνεται συγκριτική ανάλυση με άλλα μοντέλα ώστε να αποδειχθεί η χρησιμότητα του νέου μοντέλου.

3. Υλοποίηση μοντέλου πρόβλεψης κινητικότητας ατόμων σε περιβάλλον γραφείου

3.1 Παρουσίαση του προβλήματος που θα αντιμετωπίσει η παρούσα εργασία

Το πρόβλημα που θα αντιμετωπιστεί στην παρούσα εργασία βασίζεται σε αυτό που παρουσιάστηκε στην εργασία των Garg, Neeraj, et al [2] στην ενότητα 4. Υπάρχει ένα σύνολο καταστάσεων-τοποθεσιών στις οποίες μπορεί να βρεθεί το άτομο και μεταβαίνει απ' την μία στην άλλη με βάση μια κατανομή πιθανοτήτων. Επιπλέον, μπορεί να υπάρχει επιπλέον πληροφορία (evidence) για το άτομο που να διαφοροποιεί τις πιθανότητες μετάβασης. Οι παραδοχές που έγιναν είναι ότι το άτομο δεν μπορεί να βρίσκεται σε δύο καταστάσεις ταυτόχρονα και κάθε μετάβαση συμβαίνει προς μία διαφορετική κατάσταση (δεν είναι δυνατή η μετάβαση προς την ίδια). Οι καταστάσεις αυτές είναι υποτιθέμενες καταστάσεις ενός ατόμου στην καθημερινή ζωή που βασίζονται στην εργασιακή ζωή ενός υπαλλήλου και φαίνονται στον πίνακα 2. Η λίστα αυτή βασίστηκε στην αντίστοιχη απ' την εργασία [2] και σε καμία περίπτωση δεν είναι διεξοδική, αλλά σκοπεύει να αποτελέσει παράδειγμα λειτουργίας του μοντέλου.

Πίνακας 2: Διαφορετικές καταστάσεις ατόμου στην καθημερινή ζωή

S1	Το άτομο είναι στο σπίτι
S4	Στο γραφείο (δουλεύει)
S5	Στα κεντρικά γραφεία
S6	Έρχεται στο γραφείο για διάλειμμα φαγητού
S7	Σε κάποια εκδήλωση ή μάζωξη
S8	Φεύγει νωρίς απ' το γραφείο για εξωτερικές δουλειές
S9	Σε εξωτερικό ραντεβού/συνάντηση

Έγινε προσπάθεια να βρεθούν τα δεδομένα που χρησιμοποιήθηκαν στην εργασία [2] ώστε να υλοποιηθεί και να αξιολογηθεί το μοντέλο με ίδιο σετ δεδομένων αλλά αυτό δεν κατέστη δυνατό. Σε επικοινωνία με τους συγγραφείς της εργασίας επιβεβαιώθηκε ότι δεν υπάρχει πραγματικό σετ δεδομένων με βάση κάποιες πραγματικές μετρήσεις. Επιπλέον, παρατηρήθηκαν κάποιες ασυνέπειες στις πιθανότητες μετάβασης (το άθροισμα πιθανοτήτων μετάβασης σε κάποιες καταστάσεις δεν είναι 1) της εργασίας αυτής. Ακόμα, έγινε έρευνα για την ύπαρξη κάποιου σετ δεδομένων με ακολουθίες μετακινήσεων από κινητές συσκευές όμοιο με αυτά που παρουσιάστηκαν στις εργασίες [5] [6] [7] [8] χωρίς όμως να βρεθεί κάποιο δημόσια διαθέσιμο. Για τους παραπάνω λόγους αποφασίστηκε στην παρούσα εργασία να δημιουργηθεί συνθετικό σετ δεδομένων, το οποίο βασίζεται σ' αυτό που παρουσιάζεται στην [2] αλλά αντιμετωπίζει τις ασυνέπειες που παρατηρήθηκαν. Οι πιθανότητες μετάβασης απ' την μία κατάσταση στην άλλη ορίστηκαν απ' τον συγγραφέα της παρούσης εργασίας διαισθητικά και με βάση τον αντίστοιχο πίνακα της εργασίας αναφοράς. Οι πιθανότητες αυτές απεικονίζονται στον πίνακα 3.

Πίνακας 3: Πιθανότητες μετάβασης από όλες τις καταστάσεις

	S1	S4	S5	S6	S7	S8	S9
S1	X	0.8	0.1	X	0.03	X	0.07
S4	0.3	X	0.2	0.3	X	0.1	0.1
S5	0.5	0.2	X	X	0.2	0.1	X
S6	X	0.6	X	X	X	0.25	0.15
S7	0.85	0.05	0.05	X	X	X	0.05

S8	0.9	X	X	X	0.1	X	X
S9	0.55	0.2	0.2	X	0.05	X	X

Επιπλέον, στο παραπάνω μοντέλο αποφασίστηκε να προστεθεί μια πληροφορία πλαισίου του ατόμου (evidence) ή οποία μπορεί να διαφοροποιήσει τις πιθανότητες μετάβασης των καταστάσεων. Η πληροφορία αυτή είναι το είδος του υπαλλήλου και το αν έχει κάποιες δουλειές για το σπίτι. Στον πίνακα 4 απεικονίζονται οι συγκεκριμένες κατηγορίες που προστέθηκαν.

Πίνακας 4: Κατηγορίες πληροφορίας πλαισίου ατόμων διαθέσιμη για κάθε μετάβαση

E1	Υψηλά ιστάμενος υπάλληλος
E2	Υπάλληλος marketing
E3	Απλός υπάλληλος
E4	Δουλειές οικίας

Οι κατηγορίες αυτές επιλέχθηκαν ώστε η παρουσία τους να μπορεί να βοηθήσει τον μηχανισμό προβλέψεων να προβεί σε επιτυχείς προβλέψεις. Δηλαδή μια μετάβαση που γενικά συμβαίνει σπάνια μπορεί με την παρουσία μιας πληροφορίας πλαισίου να συμβαίνει πιο συχνά και να μπορεί να προβλεφθεί με μεγαλύτερη ακρίβεια. Π.χ. αναμένεται ότι ένας υπάλληλος marketing θα πηγαίνει συχνά σε εξωτερικά ραντεβού με πελάτες σε σχέση με την ολότητα των υπαλλήλων. Επίσης, ενώ γενικά ένας υπάλληλος φεύγει όταν τελειώσει το ωράριο του σε σπάνιες περιπτώσεις όταν υπάρχει μια επιτακτική δουλειά για το σπίτι/οικογένεια θα φύγει νωρίτερα να την τακτοποιήσει. Η λίστα αυτή όπως και παραπάνω δεν είναι σε καμία περίπτωση διεξοδική και η προσθήκη της σκοπεύει να αναδείξει την σημασία των πληροφοριών πλαισίου σε ένα μοντέλο προβλέψεων κινητικότητας.

Βασιζόμενοι στις παραπάνω πληροφορίες θα δημιουργήσουμε συνθετικά δεδομένα και στη συνέχεια το μοντέλο πρόβλεψης κινητικότητας τα οποία θα αναλυθούν διεξοδικά στις επόμενες υπό-ενότητες. Η υλοποίηση του συστήματος θα γίνει σε γλώσσα Python έκδοσης 3.8. Η python είναι μια ευρέως διαδεδομένη γλώσσα προγραμματισμού στο πεδίου της ανάλυσης δεδομένων και μηχανικής μάθησης καθώς είναι προσιτή σε χρήστες που δεν είναι προγραμματιστές (εύκολο συντακτικό) και κυρίως υπάρχει μεγάλη κοινότητα χρηστών και έχουν αναπτυχθεί πολλές βιβλιοθήκες για μηχανική μάθηση και ανάλυση δεδομένων που μπορεί ο χρήστης να χρησιμοποιήσει εύκολα και δωρεάν.

Αφού δημιουργηθούν τα συνθετικά δεδομένα με στήλες *Current state, Evidence, Next state* θα υλοποιηθεί ένα μοντέλο πρόβλεψης όπου με είσοδο τις δύο πρώτες στήλες θα προβλέπει την επόμενη κατάσταση που θα βρεθεί ο χρήστης. Στη συνέχεια θα αξιολογηθεί η απόδοση του μοντέλου σε διάφορες συνθήκες και σε σύγκριση με άλλους γνωστά μοντέλα πρόβλεψης.

3.2 Δημιουργία συνθετικών δεδομένων

Σε αυτήν την ενότητα θα αναλυθεί η διαδικασία δημιουργίας των συνθετικών δεδομένων μετακινήσεων σύμφωνα με την παραπάνω περιγραφή και τα στοιχεία των πινάκων 2,3 και 4. Προαπαιτούμενη είναι η εγκατάσταση της Python στον υπολογιστή. Επιλέξαμε την έκδοση 3.8 που είναι απ' τις πιο πρόσφατες διαθέσιμες εκδόσεις [15]. Αρχικά, δημιουργούμε ένα *virtual environment* όπου θα εγκαταστήσουμε όλες τις απαραίτητες βιβλιοθήκες για την εφαρμογή μας. Αυτό είναι μια κοινή πρακτική στην python και οδηγίες μπορούν να βρεθούν στον οδηγό τεκμηρίωσης της αντίστοιχης εντολής της Python [16]. Για την υλοποίηση της δημιουργίας των συνθετικών δεδομένων απαιτείται η εγκατάσταση των βιβλιοθηκών pandas [17] και numpy [18] η οποία μπορεί να γίνει με δύο απλές εντολές όταν βρίσκεσαι στο *virtual environment* που δημιουργήθηκε παραπάνω:

- *pip install pandas/ conda install pandas* (εάν έχει εγκατασταθεί anaconda έκδοση της python)
- *pip install numpy/ conda install numpy* (εάν έχει εγκατασταθεί anaconda έκδοση της python)

Η υλοποίηση της δημιουργίας συνθετικών δεδομένων έγινε στο αρχείο *create_data.py* όπου δημιουργήθηκε μια κλάση *GenerateData*. Σκοπός είναι να δημιουργηθεί ένα αρχείο με τρεις στήλες *Current state, Evidence, Next state,* ώστε το μοντέλο με είσοδο τις δύο πρώτες να προβλέπει την τρίτη που είναι η μελλοντική κατάσταση. Η συγκεκριμένη μορφή είναι απλοποιημένη σε σχέση με ένα πραγματικό σετ δεδομένων όπως αναλύθηκε στην ενότητα 2. Ένα πραγματικό σετ δεδομένων θα ήταν της μορφής *τοποθεσία, ώρα, πληροφορία πλαισίου.* Όπως αποδείχθηκε παραπάνω τα πραγματικά δεδομένα μπορούν να επεξεργαστούν ώστε απ' τα δεδομένα τοποθεσίας να εξαχθεί ένα αναγνωριστικό τοποθεσίας όμοια με την δική μας κατάσταση. Επίσης συνήθως η ώρα ή κάποια πληροφορία πλαισίου που προκύπτει απ' την χρήση εφαρμογής στην κινητή συσκευή (ημερολόγιο) χωρίζονται σε ζώνες ώστε να υπάρχουν πεπερασμένες διαφορετικές καταστάσεις. Η επόμενη κατάσταση στο δικό μας σετ είναι αντίστοιχα η επόμενη καταγραφή με διαφορετικό αναγνωριστικό τοποθεσίας σε ένα πραγματικό σετ δεδομένων, το οποίο εξάγεται με ευκολία. Στα συνθετικά δεδομένα επιλέχθηκε να μην συμπεριληφθεί η ώρα (*timestamp*) καθώς είναι δύσκολο να δημιουργηθούν αληθοφανή συνθετικά δεδομένα με αυτό το κριτήριο. Απ' τα παραπάνω φαίνεται ότι παρόλο τις απλοποιήσεις που έγιναν αυτές δεν αλλάζουν την ουσία του προβλήματος πρόβλεψης και μπορούν να χρησιμοποιηθούν για την ανάπτυξη μοντέλου πρόβλεψης.
Το αρχείο *create_data.py* ξεκινάει με την είσοδο των απαραίτητων βιβλιοθηκών και στη ορίζεται η κλάση *GenerateData* και αρχικοποιούνται κάποιες βασικές μεταβλητές με τις πληροφορίες απ' τους πίνακες 2,3,4. Αυτά φαίνονται αναλυτικά στο Σχ. 3

```python
import numpy as np
import pandas as pd


class GenerateData:
    column_names = ["Current state", "Next state"]
    states = ["S1", "S4", "S5", "S6", "S7", "S8", "S9"]
    evidence_dict = {
        "E1": "High rank employee",
        "E2": "Marketing employee",
        "E3": "Other employee",
        "E4": "Household work",
    }
    states_dict = {
        "S1": "Person is at home",
        "S4": "In office (cooperate)",
        "S5": "In head office",
        "S6": "Coming to office for lunch",
        "S7": "Attending some party or get together",
        "S8": "Leaving early from office for outside work",
        "S9": "In meeting outside office"
    }
    transition_probs = {
        "S1": [0, 0.8, 0.1, 0, 0.03, 0, 0.07],
        "S4": [0.3, 0, 0.2, 0.3, 0, 0.10, 0.10],
        "S5": [0.5, 0.2, 0, 0, 0.2, 0.1, 0],
        "S6": [0, 0.6, 0, 0, 0, 0.25, 0.15],
        "S7": [0.85, 0.05, 0.05, 0, 0, 0, 0.05],
        "S8": [0.9, 0, 0, 0, 0.1, 0, 0],
        "S9": [0.55, 0.2, 0.2, 0, 0.05, 0, 0]
    }
```

Σχήμα 3: Αρχικοποίηση της κλάσης και αρχικών μεταβλητών

Στη συνέχεια ορίζεται η μέθοδος αρχικοποίησης της κλάσης με μία μεταβλητή που ορίζει τον αριθμό δειγμάτων για κάθε μία κατάσταση στην στήλη *current state*. Δηλαδή, αν οριστεί ένας αριθμός 1000 θα έχει δημιουργηθούν συνολικά 7000 σειρές δεδομένων με 1000 δείγματα στην κάθε κατάσταση. Σ' αυτήν την μέθοδο δημιουργούνται δύο λίστες *current_state*, *next_state* όπου με ένα φωλιασμένο βρόχο προστίθεται ο ορισμένος αριθμός δειγμάτων στην *current_state* και στη λίστα *next_state* προστίθενται οι καταστάσεις ανάλογα με τις πιθανότητες μετάβασης χρησιμοποιώντας την συνάρτηση *random.choice* απ' το πακέτο *numpy*. Στη συνέχεια, οι λίστες αυτές μετατρέπονται σε ένα *pandas.Dataframe* [19] που εισάγονται τα δεδομένα στις στήλες *Current state, Next state, Evidence* με την τελευταία να είναι κενή όπως φαίνεται στο Σχ. 4.

```
def __init__(self, num_samples: int):
    """This function will initialize a GenerateDara object, which contains a pandas Dataframe with the data created.
    This is synthetic data for  mobility prediction based on the states and probabilities presented on paper
    Efficient mobility prediction scheme for pervasive networks.
    num_samples defines the number of samples to be produced for each state.
    """
    self.num_samples = num_samples

    current_state = []
    next_state = []

    for i in self.states:
        for k in range(num_samples):
            current_state.append(i)
            if i in self.transition_probs.keys():
                next_state.append(np.random.choice(self.states, p=self.transition_probs[i]))
            else:
                next_state.append(np.nan)

    variables = [current_state, next_state]
    self.df = pd.DataFrame(variables).transpose()
    self.df.columns = self.column_names

    self.column_names.append("Evidence")
    self.df[self.column_names[2]] = None
```

Σχήμα 4: Μέθοδος αρχικοποίησης της κλάσης GenerateData

Το τελευταίο κομμάτι αφορά τον τρόπο με τον οποίο θα συμπληρωθεί η στήλη *Evidence*. Αρχικά υλοποιείται μια μέθοδος που εισάγει δεδομένα στην στήλη αυτή με βάση μια συγκεκριμένη μετάβαση καταστάσεων, από μια λίστα με την πληροφορία πλαισίου με βάση μία λίστα πιθανοτήτων όπως φαίνεται στο Σχ. 5

```
def add_evidence(self, state1: str, state2: str, evidence: list, probability: list):
    """This function adds evidence data on the given transition from state1 to state2
    based on the evidence and probability lists provided"""
    for i in range(len(self.df)):
        if self.df[self.column_names[0]][i] == state1 and self.df[self.column_names[1]][i] == state2:
            self.df[self.column_names[2]][i] = np.random.choice(evidence, p=probability)
```

Σχήμα 5: Μέθοδος προσθήκης δεδομένων στην στήλη Evidence

Για κάθε μετάβαση π.χ. S1→S4, ορίστηκαν διαισθητικά η παρουσία της κατηγορίας πλαισίου απ' τον πίνακα 4 με βάση κάποιες πιθανότητες οι οποίες φαίνονται στον πίνακα 5. Με βάση αυτές τις πιθανότητες δημιουργείται μία μέθοδος που χρησιμοποιεί την *add_evidence* και γεμίζει την στήλη *Evidence* του *Dataframe* με δεδομένα όπως φαίνεται στο Σχ. 6. Τέλος, δημιουργούμε ένα αντικείμενο της κλάσης με τον αριθμό μετρήσεων που θέλουμε και εξάγουμε το αντίστοιχο *Dataframe* σε ένα csv αρχείο (Σχ. 7). Εδώ να σημειωθεί ότι λόγω χρησιμοποίησης της συνάρτησης *random.choice* [20] το αποτέλεσμα του προγράμματος σε κάθε εκτέλεση δεν είναι το ίδιο.

Πίνακας 5: Πιθανότητες παρουσίας πληροφορίας πλαισίου ανά μετάβαση καταστάσεων

	E1	E2	E3	E4
S1→S4	0.05	0.25	0.70	X
S1→S5	0.65	0.3	0.05	X
S1→S7	0.4	0.4	0.2	X
S1→S9	0.35	0.6	0.05	X
S4→S1	0.03	0.2	0.77	X
S4→S5	0.3	0.6	0.1	X
S4→S6	0.05	0.25	0.7	X
S4→S8	0.15	0.15	X	0.7
S4→S9	0.35	0.6	0.05	X
S5→S1	0.7	0.2	0.1	X
S5→S4	0.3	0.4	0.3	X
S5→S7	0.4	0.3	0.3	X
S5→S8	0.2	0.2	X	0.6
S6→S4	0.05	0.15	0.8	X
S6→S8	0.05	0.15	0.15	0.65
S6→S9	0.35	0.55	0.1	X
S7→S1	0.1	0.2	0.7	X
S7→S4	X	0.4	0.6	X
S7→S5	0.75	0.25	X	X
S7→S9	0.6	0.4	X	X
S8→S1	0.03	0.05	0.22	0.7
S8→S7	0.25	0.15	0.6	X
S9→S1	0.1	0.2	0.3	0.4
S9→S4	0.15	0.6	0.25	X
S9→S5	0.65	0.3	0.05	X
S9→S7	0.5	0.4	0.1	X

```
def fill_evidence(self):
    """This function fills the evidence column with synthetic data"""

    self.add_evidence("S1", "S4", ["E1", "E2", "E3"], [0.05, 0.25, 0.7])
    self.add_evidence("S1", "S5", ["E1", "E2", "E3"], [0.65, 0.3, 0.05])
    self.add_evidence("S1", "S7", ["E1", "E2", "E3"], [0.4, 0.4, 0.2])
    self.add_evidence("S1", "S9", ["E1", "E2", "E3"], [0.35, 0.6, 0.05])
    self.add_evidence("S4", "S1", ["E1", "E2", "E3"], [0.03, 0.2, 0.77])
    self.add_evidence("S4", "S5", ["E1", "E2", "E3"], [0.3, 0.6, 0.1])
    self.add_evidence("S4", "S6", ["E1", "E2", "E3"], [0.05, 0.25, 0.7])
    self.add_evidence("S4", "S8", ["E4", "E2", "E1"], [0.7, 0.15, 0.15])
    self.add_evidence("S4", "S9", ["E2", "E1", "E3"], [0.6, 0.35, 0.05])
    self.add_evidence("S5", "S1", ["E1", "E2", "E3"], [0.7, 0.2, 0.1])
    self.add_evidence("S5", "S4", ["E1", "E2", "E3"], [0.3, 0.4, 0.3])
    self.add_evidence("S5", "S7", ["E1", "E2", "E3"], [0.4, 0.3, 0.3])
    self.add_evidence("S5", "S8", ["E4", "E1", "E2"], [0.6, 0.2, 0.2])
    self.add_evidence("S6", "S4", ["E3", "E1", "E2"], [0.8, 0.05, 0.15])
    self.add_evidence("S6", "S8", ["E4", "E1", "E2", "E3"], [0.65, 0.05, 0.15, 0.15])
    self.add_evidence("S6", "S9", ["E1", "E2", "E3"], [0.35, 0.55, 0.1])
    self.add_evidence("S7", "S1", ["E1", "E2", "E3"], [0.1, 0.2, 0.7])
    self.add_evidence("S7", "S4", ["E1", "E2", "E3"], [0.0, 0.4, 0.6])
    self.add_evidence("S7", "S5", ["E1", "E2"], [0.75, 0.25])
    self.add_evidence("S7", "S9", ["E1", "E2"], [0.6, 0.4])
    self.add_evidence("S8", "S1", ["E4", "E3", "E2", "E1"], [0.7, 0.22, 0.05, 0.03])
    self.add_evidence("S8", "S7", ["E3", "E2", "E1"], [0.6, 0.15, 0.25])
    self.add_evidence("S9", "S1", ["E3", "E2", "E1", "E4"], [0.3, 0.2, 0.1, 0.4])
    self.add_evidence("S9", "S4", ["E3", "E2", "E1"], [0.25, 0.6, 0.15])
    self.add_evidence("S9", "S5", ["E3", "E2", "E1"], [0.05, 0.3, 0.65])
    self.add_evidence("S9", "S7", ["E3", "E2", "E1"], [0.1, 0.4, 0.5])
```

Σχήμα 6: Μέθοδος συμπλήρωσης δεδομένων στήλης Evidence

Αυτό συμβαίνει γιατί τα δεδομένα συμπληρώνονται με πιθανότητα ενδεχόμενα σύμφωνα με τον ορισμό της συνάρτησης. Δηλαδή, για την μετάβαση S1→S4 που ορίζεται πιθανότητα 0,8, αν επιλέξουμε αριθμό δειγμάτων 1000 δεν σημαίνει ότι θα έχουμε 800 δείγματα αυτής της μετάβασης, αλλά κάποιες φορές μπορεί να είναι 797, άλλες 802 κλπ. Γι' αυτό το λόγο επιλέχθηκε η εξαγωγή σε αρχείο csv ώστε να έχουμε σε κάθε εκτέλεση του μοντέλου την ίδια ακριβώς είσοδο.

```
data = GenerateData(5000)
data.fill_evidence()
data.df.to_csv("test_data_5000.csv", index=False)
```

Σχήμα 7: Εξαγωγή αρχείου csv με τα συνθετικά δεδομένα

3.3 Υλοποίηση μοντέλου πρόβλεψης κινητικότητας

Με βάση την βιβλιογραφία που μελετήθηκε αποφασίσαμε να προχωρήσουμε στην υλοποίηση ενός μοντέλου βασιζόμενοι στα *Bayesian Networks*. Ένα Bayesian Network είναι ένας κατευθυνόμενος ακυκλικός γράφος με κόμβους που αναπαριστούν μεταβλητές και ακμές που δείχνουν την μετάβαση απ' την μία μεταβλητή στην άλλη [10]. Τα *Bayesian Networks* είναι χρήσιμα ώστε να μαθαίνουν τις κατανομές πιθανοτήτων γεγονότων ώστε να εξάγουμε συμπεράσματα για τις σχέσεις μεταξύ των παρατηρήσεων και της κατάστασης του συστήματος. Χρειάζονται εκμάθηση με ιστορικά δεδομένα αλλά μπορούν να γίνει *retraining* και να προσαρμοστούν σε νέα δεδομένα [2]. Μετά την εκμάθηση το *Bayesian Network* αποτελείται από ένα σετ κόμβων, ακμών και πιθανοτήτων που συνδέονται με κάθε ακμή [2]. Η λογική του μοντέλου πρόβλεψης είναι να γίνει εκμάθησης του συστήματος με τα δεδομένα που παράχθηκαν στο προηγούμενο βήμα και να εξαχθούν οι κατανομές πιθανότητας μετάβασης για κάθε πιθανή είσοδο του συστήματος. Παρουσία των δεδομένων εισόδου (τρέχουσα κατάσταση, πληροφορία πλαισίου) το μοντέλο προβλέπει την μελλοντική κατάσταση που έχει την μέγιστη πιθανότητα πραγματοποίησης.

Με βάση τα παραπάνω έγινε έρευνα για την ύπαρξη βιβλιοθήκης της Python που να υλοποιεί Bayesian networks. Μετά από την σχετική αναζήτηση βρέθηκε η *py-bbn* [21] η οποία και εγκαταστάθηκε στο *virtual environment* της εργασίας σύμφωνα με τις οδηγίες ώστε να ερευνηθεί η υλοποίηση του συστήματος με βάση αυτή την βιβλιοθήκη.

Ακολουθώντας τις οδηγίες στη σελίδα τις βιβλιοθήκης πρέπει να εισαχθούν οι κόμβοι (*node*) για κάθε κατάσταση όπου θα περιέχονται οι πιθανότητες μετάβασης στις άλλες καταστάσεις και οι κατευθυνόμενες ακμές (*edges*) στις επιτρεπόμενες μεταβάσεις ώστε να δημιουργηθεί ο ακυκλικός γράφος.

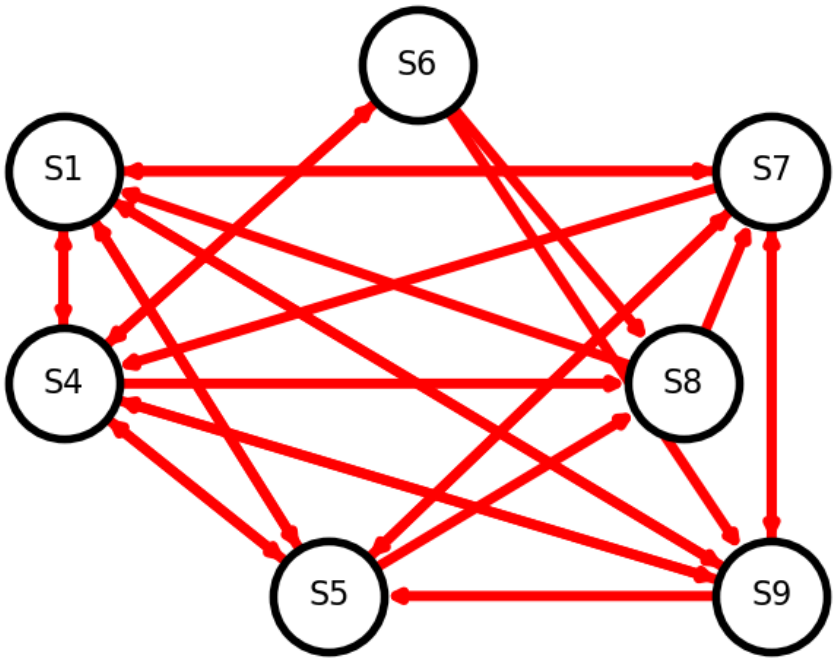
```
def get_state_paths(state) -> list:
    """Returns a sorted list of the possible next states from the current state"""
    return sorted(data[data[df_columns[0]] == state][df_columns[1]].unique().tolist())


def get_prob_transition_per_node(state) -> list:
    """Returns a list of probabilities transition based on the available paths for each state"""
    paths = get_state_paths(state)
    transition_freq = []
    for i in paths:
        transition_freq.append(data[(data[df_columns[0]] == state) & (data[df_columns[1]] == i)][df_columns[0]].count())
    return [float(x)/sum(transition_freq) for x in transition_freq]
```

Σχήμα 8: Βοηθητικές μέθοδοι υπολογισμού διαθέσιμων μεταβάσεων και πιθανοτήτων από μία αρχική κατάσταση

```
bbn = Bbn()
nodes = []


def create_nodes():
    count = 0
    for i in states:
        nodes.append(BbnNode(Variable(count, i, get_state_paths(i)), get_prob_transition_per_node(i)))
        bbn.add_node(nodes[count])
        count += 1


def add_edges():
    for i in nodes:
        node_values = i.to_dict().get('variable').get('values')
        for j in nodes:
            if j.to_dict().get('variable').get('name') in node_values:
                bbn.add_edge(Edge(i, j, EdgeType.DIRECTED))


create_nodes()
add_edges()
```

Σχήμα 9: Αρχικοποίηση και προσθήκη κόμβων, ακμών στο *bayesian network*

Για τον σκοπό αυτό δημιουργήθηκαν δύο βοηθητικές μέθοδοι που επιστρέφουν τις πιθανές επόμενες καταστάσεις από μία αρχική κατάσταση και τις αντίστοιχες πιθανότητες μετάβασης, που φαίνονται στο Σχ. 8. Στη συνέχεια αρχικοποιείται το *Bayesian network object* και προστίθενται οι κόμβοι και οι ακμές όπως φαίνεται στο Σχ. 9. Το επόμενο βήμα για τον υπολογισμό των πιθανοτήτων σε κάθε κόμβο είναι η μετατροπή του *Bayesian network* σε συνδεδεμένο δέντρο με την ακόλουθη εντολή *join_tree = InferenceController.apply(bbn)*. Η συγκεκριμένη εντολή όμως δίνει σφάλμα (*list index out of range*). Αυτό συμβαίνει καθώς η συγκεκριμένη βιβλιοθήκη έχει περιορισμό σε κάθε κόμβο να συμβαίνουν όλα τα πιθανά ενδεχόμενα. Έτσι, αν το σετ δεδομένων δεν καλύπτει αυτή την συνθήκη δεν μπορεί να χρησιμοποιηθεί η συγκεκριμένη βιβλιοθήκη. Στην περίπτωση μας απ' τον ορισμό του προβλήματος δεν είναι όλες οι πληροφορίες πλαισίου διαθέσιμες σε κάθε κόμβο, οπότε η χρήση της συγκεκριμένης βιβλιοθήκης είναι αδύνατη στο πρόβλημα μας.



Σχήμα 10: Γράφος μετάβασης καταστάσεων του προβλήματος της εργασίας

Παρόλα αυτά η συγκεκριμένη υλοποίηση επιτρέπει με την επιπλέων εγκατάσταση των βιβλιοθηκών *networkx* [22] και *matplotlib* [23] την απεικόνιση του γράφου του *Bayesian network* σύμφωνα με τις οδηγίες στην σελίδα της βιβλιοθήκης της *py-bbn* [21]. Έτσι με την υλοποίηση της μεθόδου που φαίνεται στο Σχ. 11 είναι δυνατή η απεικόνιση του γράφου του προβλήματος της παρούσας εργασίας, που απεικονίζεται στο Σχ. 10. Η απεικόνιση του γράφου ήταν εν τέλει η προσφορά της συγκεκριμένης υλοποίησης στην εργασία χωρίς να έχει κάποια συμμετοχή στο μοντέλο πρόβλεψης.

```
def display_graph():
    # Set node positions
    pos = {0: (-2, 2), 1: (-2, 0), 2: (-0.5, -2), 3: (0, 3), 4: (2, 2), 5: (1.5, 0), 6: (2, -2)}

    # Set options for graph looks
    options = {
        "font_size": 16,
        "node_size": 3000,
        "node_color": "white",
        "edgecolors": "black",
        "edge_color": "red",
        "linewidths": 4,
        "width": 5, }

    # Generate graph
    n, d = bbn.to_nx_graph()
    nx.draw(n, with_labels=True, labels=d, pos=pos, **options)

    # Update margins and print the graph
    ax = plt.gca()
    ax.margins(0.10)
    plt.axis("off")
    plt.show()
```

Σχήμα 11: Μέθοδος απεικόνισης του γράφου

Καθώς η παραπάνω βιβλιοθήκη δεν ήταν δυνατό να χρησιμοποιηθεί έγινε μια εναλλακτική υλοποίηση χωρίς τη χρήση κάποιου πακέτου αλλά μόνο με την χρήση των *pandas* που είναι απαραίτητη για την επεξεργασία του αρχείου δεδομένων. Η λογική της υλοποίησης είναι αρχικά να δημιουργηθεί μια μέθοδος (*probs*) που θα υπολογίζει τις πιθανότητες μετάβασης στην επόμενη κατάσταση με βάση την τρέχουσα και την πληροφορία πλαισίου για όλες τις περιπτώσεις. Στο συγκεκριμένο πρόβλημα έχουμε τις στήλες *Current state* και *Evidence* που είναι οι parent classes και η *Next state* που είναι η κλάση πρόβλεψης (*child class*). Η μέθοδος δέχεται ως είσοδο ένα *pandas.Dataframe* με τα δεδομένα και το όνομα της στήλης και το εύρος τιμών της κάθε στήλης για τις *parent* και *child* κλάσεις. Υποστηρίζει μέχρι και 3 parent κλάσεις αλλά εύκολα μπορεί να επεκταθεί και σε παραπάνω.

Η μέθοδος αυτή υπολογίζει πόσες φορές απαντώνται οι εκάστοτε συνδυασμοί των παραπάνω κλάσεων στα δεδομένα εισόδου της μεθόδου και αποθηκεύει την πληροφορία σε ένα *dictionary* με κλειδιά της μορφής "*Next state: x, Current state: y, Evidence: z*" ,όπου x,y,z οι τιμές των αντίστοιχων στηλών π.χ. "*Next state: S4, Current state: S1, Evidence: E1*". Σαν τιμή στο *dictionary* αποθηκεύεται τα περιστατικά (*occurrences*) του κάθε συνδυασμού γεγονότων τα οποία στο τέλος κανονικοποιούνται ώστε να μετατραπούν σε πιθανότητες από 0 έως 1. Η μέθοδος αυτή φαίνεται στο Σχ. 12 όπου έχουν υπογραμμιστεί τα βασικά σημεία υπολογισμού. Ένα παράδειγμα επιστροφής της συγκεκριμένης μεθόδου για Current State=S1 και Evidence=E1 φαίνεται παρακάτω:

{'Next state: S1, Current state: S1, Evidence: E1': 0.0,
'Next state: S4, Current state: S1, Evidence: E1': 0.2740740740740741,
'Next state: S5, Current state: S1, Evidence: E1': 0.4666666666666667,

'Next state: S6, Current state: S1, Evidence: E1': 0.0,
'Next state: S7, Current state: S1, Evidence: E1': 0.0962962962962963,
'Next state: S8, Current state: S1, Evidence: E1': 0.0,
'Next state: S9, Current state: S1, Evidence: E1': 0.16296296296296298}

```python
# Define a function for printing marginal probabilities
def probs(df, child, childbands,
          parent1=None, parent1bands=None,
          parent2=None, parent2bands=None,
          parent3=None, parent3bands=None,
          parent4=None, ):
    """This function provides a dictionary with the marginal probabilities for a classification
     problem with up to 3 parents. The keys of the dictionary provide info on what the exact column
      value is in a {child}: {val4}, {parent1}: {val}, {parent2}: {val2}, {parent3}: {val3} format"""
    # Initialize empty list
    prob = {}
    if parent1 is None:
        # Calculate probabilities
        for val in childbands:
            key = f"{child}: {val}"
            value = df[child].tolist().count(val)
            prob.update({key: value})
    elif parent1 is not None:
        # Check if parent2 exists
        if parent2 is None:
            # Calculate probabilities
            for val in parent1bands:
                for val2 in childbands:
                    key = f"{child}: {val2}, {parent1}: {val}"
                    value = df[df[parent1] == val][child].tolist().count(val2)
                    prob.update({key: value})
        elif parent2 is not None:
            # Check if parent3 exists
            if parent3 is None:
                # Calculate probabilities
                for val in parent1bands:
                    for val2 in parent2bands:
                        for val3 in childbands:
                            key = f"{child}: {val3}, {parent1}: {val}, {parent2}: {val2}"
                            value = df[(df[parent1] == val) & (df[parent2] == val2)][child].tolist().count(val3)
                            prob.update({key: value})
            elif parent3 is not None:
                # Check if parent4 exists
                if parent4 is None:
                    # Calculate probabilities
                    for val in parent1bands:
                        for val2 in parent2bands:
                            for val3 in parent3bands:
                                for val4 in childbands:
                                    key = f"{child}: {val4}, {parent1}: {val}, {parent2}: {val2}, {parent3}: {val3}"
                                    value = df[(df[parent1] == val) & (df[parent2] == val2) & (df[parent3] == val3)][child].tolist().count(val4)
                                    prob.update({key: value})
    sum_values = sum(prob.values())
    for i in prob.keys():
        if sum_values != 0:
            norm_value = prob.get(i)/sum_values
            prob.update({i: norm_value})
        else:
            prob.update({i: 0})
    return prob
```

Σχήμα 12: Μέθοδος υπολογισμού πιθανοτήτων μεταβάσεων

Η συνήθης δομή κατασκευής ενός μοντέλου μηχανικής μάθησης περιλαμβάνει μια μέθοδο *fit* όπου εκπαιδεύεται το μοντέλο με βάση κάποια δεδομένα και μια μέθοδο *predict* όπου κάνει πρόβλεψη με κάποια διαφορετικά δεδομένα εισόδου όπου δεν περιέχεται η κλάση πρόβλεψης. Τα παραπάνω φαίνονται σε παραδείγματα στον οδηγό τεκμηρίωσης της βιβλιοθήκης scikit-learn [24], η οποία είναι μια απ' τις πιο ευρέως διαδεδομένες βιβλιοθήκες για μηχανική μάθηση. Με όμοιο τρόπο έγινε η υλοποίηση των αντίστοιχων μεθόδων στην δική μας περίπτωση.

Αρχικά έχουμε μία μέθοδο fit η οποία επιστρέφει ένα *dictionary* με την προβλεπόμενη κατάσταση για κάθε συνδυασμό δεδομένων με βάση κάποια δεδομένα εκμάθησης. Η μέθοδος

χρησιμοποιεί την μέθοδο *probs* για τον υπολογισμό των πιθανοτήτων και σαν πρόβλεψη διαλέγει απλά την κατάσταση με την μέγιστη πιθανότητα πραγματοποίησης στα δεδομένα εκμάθησης. Η ακριβής υλοποίηση φαίνεται στο Σχ. 13 όπου υπογραμμίστηκαν τα βασικά σημεία υπολογισμού.

```
def fit(train, child, parent1, parent2) -> dict:
    """This function provides a dictionary with the predicted state for all data combination
     based on a train dataset. The algorithm logic is to simply select the state with the
     maximum probability from possible states in the dataset."""
    transition_pred = {}
    parent1_list = train[parent1].unique().tolist()
    parent2_list = train[parent2].unique().tolist()
    for i in parent1_list:
        for j in parent2_list:
            transition_prob = probs(train, child, parent1_list, parent1, [i], parent2, [j])
            # transition_prob = probs(train, df_columns[1], states, df_columns[0], [i])  # evidence not accounted
            if sum(transition_prob.values()) != 0:
                transition_pred.update({f"{i},{j}": list(transition_prob.keys())[list(transition_prob.values()).index(
                        max(transition_prob.values()))][12:14]})
            else:
                transition_pred.update({f"{i},{j}": None})
    return transition_pred
```

Σχήμα 13: Μέθοδος fit του μοντέλου πρόβλεψης

Ένα παράδειγμα επιστροφής της μεθόδου είναι παρακάτω:
{'S1,E3': 'S4',
'S1,E2': 'S4',
'S1,E1': 'S5',
'S1,E4': None,
'S4,E3': 'S6',
'S4,E2': 'S5',
'S4,E1': 'S5',
'S4,E4': 'S8',
'S5,E3': 'S4',
'S5,E2': 'S1',
'S5,E1': 'S1',
'S5,E4': 'S8',
'S6,E3': 'S4',
'S6,E2': 'S4',
'S6,E1': 'S9',
'S6,E4': 'S8',
'S7,E3': 'S1',
'S7,E2': 'S1',
'S7,E1': 'S1',
'S7,E4': None,
'S8,E3': 'S1',
'S8,E2': 'S1',
'S8,E1': 'S7',
'S8,E4': 'S1',
'S9,E3': 'S1',
'S9,E2': 'S4',
'S9,E1': 'S5',
'S9,E4': 'S1'}

Σε αυτό το dictionary φαίνεται η πρόβλεψη του μοντέλου για κάθε συνδυασμό καταστάσεων εισόδου απ' τους πίνακες 2 κα 4.
Τέλος έχουμε τη μέθοδο *predict* που επιστρέφει μια λίστα με τις προβλέψεις του μοντέλου για κάθε σειρά του εισαγόμενου σετ δεδομένων, η οποία φαίνεται στο Σχ. 14. Με την επιστροφή αυτής της συνάρτησης μπορούμε να συγκρίνουμε τις προβλέψεις του μοντέλου με τις

πραγματικές τιμές, ώστε να αξιολογήσουμε την απόδοση του αλγορίθμου που θα γίνει στην επόμενη ενότητα.

```
def predict(test: pd.DataFrame, fit_matrix) -> list:
    """This function provides a list with the predicted next state for each
    row of the given dataset. It uses the fit function results, that is trained on
    a different dataset, to provide predictions """
    columns = test.columns.tolist()
    predictions = []
    for i in range(len(test)):
        key = f"{test[columns[0]][i]},{test[columns[1]][i]}"
        predictions.append(fit_matrix.get(key))
    return predictions
```

Σχήμα 14: Μέθοδος predict μοντέλου πρόβλεψης

4. Αξιολόγηση απόδοσης του μοντέλου πρόβλεψης κινητικότητας

Σ' αυτή την ενότητα θα αναλυθεί η απόδοση του υλοποιημένου μοντέλου πρόβλεψης με βάση τη θεωρία που αναπτύχθηκε στην ενότητα 2 για τα μέτρα αξιολόγησης. Αρχικά, να τονιστεί ότι χρησιμοποιήθηκε η ευρέως διαδεδομένη τεχνική του διαχωρισμού του σετ δεδομένων εκμάθησης και του σετ για τις προβλέψεις. Το *train test split* είναι ευρέως διαδεδομένη πρακτική στον τομέα μηχανικής μάθησης [25] που διαβεβαιώνει ότι το μοντέλο δεν έχει πρότερη γνώση στα δεδομένα προβλέψεων, κάτι που θα εισήγαγε αμφιβολίες στα αποτελέσματα του μοντέλου και πιθανά θα υπερεκτιμούσε την ακρίβεια του μοντέλου. Η διαδικασία αυτή προσομοιάζει και την πραγματική χρήση των μοντέλων σε νέα δεδομένα όπου δεν υπάρχει γνώση της κλάσης στόχου (*target class*). Στην περίπτωση μας, τρέξαμε το αρχείο *create_data.py* δύο φορές για να δημιουργήσουμε δύο αρχεία csv με 1000 δείγματα για κάθε κατάσταση. Τα αρχεία είναι τα *generated_data_1000.csv* και *test_data_1000.csv* τα οποία περιέχονται στο δημόσιο αποθετήριο του κώδικα. Παρόλο που δημιουργούνται με την ίδια μέθοδο τα αρχεία δεν είναι όμοια καθώς υπάρχει μέθοδος *random.choice* και επιπλέον στη δεύτερη περίπτωση χρησιμοποιείται η μέθοδος *pandas.sample* [26] η οποία ανακατεύει τις στήλες του αρχείου. Η εντολή είναι η ακόλουθη:

*data.df.**sample(frac=1, random_state=1)**.to_csv("test_data_1000.csv", index=False)*

και τροποποιεί την αντίστοιχη απ' το Σχ. 7. Η εντολή αυτή δεν χρειάζεται στο σετ εκμάθησης καθώς δεν παίζει κάποιο ρόλο το γεγονός ότι όλα τα δείγματα των κλάσεων είναι τοποθετημένα διαδοχικά στην εξαγωγή των πιθανοτήτων.

Όπως αναλύθηκε στην ενότητα 2 η ακρίβεια παίζει κομβικό ρόλο στην αξιολόγηση μοντέλων πρόβλεψης κινητικότητας. Θα χρησιμοποιηθούν επιπλέον και τα *precision, recall, f1-score* που δίνουν περισσότερες πληροφορίες για την απόδοση του αλγορίθμου σε κάθε κλάση και είναι ευρέως διαδεδομένα στα προβλήματα ταξινόμησης, όπως είδαμε στην ενότητα 2. Τέλος θα χρησιμοποιήσουμε και τον *confusion matrix* (Σχ. 2) για την καλύτερη οπτική απεικόνιση των αποτελεσμάτων του μοντέλου πρόβλεψης σε κάθε κλάση. Καθώς τα παραπάνω μέτρα αξιολόγησης χρησιμοποιούνται ευρέως υπάρχουν έτοιμες μέθοδοι για τον υπολογισμό τους στην βιβλιοθήκη scikit-learn. Για ευκολία θα χρησιμοποιηθούν οι μέθοδοι *classification_report, confusion_matrix* απ' το πακέτο *sklearn.metrics* [27]. Σε περίπτωση που δεν ήταν διαθέσιμες εύκολα θα μπορούσαν να εξαχθούν απ' τους μαθηματικούς τύπους του πίνακα 1. Για την χρήση των παραπάνω μεθόδων απαιτείται η εγκατάσταση της βιβλιοθήκης στο *virtual environment (pip install scikit-learn)* και η εισαγωγή τους στο εκάστοτε αρχείο *.py* με την εντολή:
from sklearn.metrics import classification_report, confusion_matrix

4.1 Εξαγωγή αρχικών αποτελεσμάτων

Σ' αυτήν την περίπτωση χρησιμοποιείται το αρχείο *generated_data_1000.csv* για την εκμάθηση του μοντέλου, το οποίο περιέχει 1000 δείγματα για κάθε κατάσταση στη στήλη *Current state* όπου οι μεταβάσεις συμβαίνουν σύμφωνα με τις πιθανότητες του πίνακα 3. Για σετ δεδομένων δοκιμής απόδοσης χρησιμοποιείται το *test_data_1000.csv* το οποίο δημιουργήθηκε με τον ίδιο τρόπο και οι σειρές ανακατεύθηκαν. Τα μετρικά αξιολόγησης εξάγονται με λίγες εντολές όπως φαίνεται στο Σχ. 14

Στον πίνακα 6 περιέχονται τα βασικά μετρικά απόδοσης του μοντέλου τα οποία αναφέρθηκαν παραπάνω. Η στήλη *support* απεικονίζει τον αριθμό των δειγμάτων που περιέχονται στο σετ δεδομένων δοκιμής για την κλάση πρόβλεψης που είναι η στήλη *Next state*. Όπως είναι αντιληπτό υπάρχουν πολλά περισσότερα δείγματα για τις καταστάσεις *S1* και *S4*, το οποίο είναι αναμενόμενο απ' τον τρόπο που παράχθηκαν τα δεδομένα με τις μεταβάσεις στις καταστάσεις αυτές να συμβαίνουν με μεγαλύτερες πιθανότητες. Αυτό θα

συνέβαινε πιθανότατα και σε ένα πραγματικό σετ δεδομένων, καθώς είναι αναμενόμενο να έχεις περισσότερα δεδομένα απ' την οικία και τον βασικό χώρο εργασίας των χρηστών.

```
trained_matrix = fit(data, df_columns[1], df_columns[0], df_columns[2])
# print(trained_matrix)
test_df = pd.read_csv("test_data_1000.csv")
# print(test_df.head)
X_test = test_df.drop(df_columns[1], axis=1)
# print(X_test)
y_test = test_df[df_columns[1]]

pred = predict(X_test, trained_matrix)
print('Bayesian probabilistic predictor')
print("Classification report")
print(classification_report(y_test, pred))
print("Confusion matrix")
print(confusion_matrix(y_test, pred))
```

Σχήμα 15: Εξαγωγή αποτελεσμάτων και μετρικών αξιολόγησης μοντέλου

Πίνακας 6: Classification report για τα αρχικά αποτελέσματα

Predicted state	precision	recall	f1-score	support
S1	0.79	0.82	0.8	3063
S4	0.7	0.81	0.75	1853
S5	0.44	0.66	0.53	558
S6	0.47	0.73	0.57	308
S7	0.4	0.04	0.08	405
S8	1	0.63	0.77	443
S9	0.58	0.13	0.21	370
accuracy			0.7	7000
macro avg	0.63	0.55	0.53	7000
weighted avg	0.71	0.7	0.68	7000

Πίνακας 7: Confusion matrix για τα αρχικά αποτελέσματα

Actual state \ Predicted state	S1	S4	S5	S6	S7	S8	S9
S1	2497	169	142	230	25	0	0
S4	260	1499	72	0	0	0	22
S5	63	102	367	26	0	0	0
S6	0	0	82	226	0	0	0
S7	229	120	39	0	17	0	0
S8	39	88	24	0	0	279	13
S9	54	154	110	4	0	0	48

Επίσης, τα κυρίως γραφεία της εταιρίας αναμένεται να έχουν περισσότερα δεδομένα σε σχέση με τα κεντρικά γραφεία που χρησιμοποιούνται κυρίως από στελέχη, τα οποία εύλογα είναι

λιγότερα σε αριθμό απ' τους υπόλοιπους υπαλλήλους. Στον πίνακα περιέχεται το *macro avg* ο οποίος είναι ο μέσος όρος κάθε μετρικού αξιολόγησης χωρίς να λαμβάνεται υπόψη η υποστήριξη κάθε κατάστασης με δείγματα και το *weight avg* το οποίο είναι ο σταθμισμένος μέσος όρος με βάση την υποστήριξη.

Το μοντέλο έχει συνολική ακρίβεια 70%, το οποίο με βάση την βιβλιογραφική έρευνα στην ενότητα 2 είναι ικανοποιητική. Το *recall* όπως παρουσιάστηκε νωρίτερα δείχνει την ικανότητα του μοντέλου να προβλέπει όλα τα σχετικά δείγματα σε κάθε κατάσταση. Πιο παραστατικά είναι το αποτέλεσμα της διαίρεσης των σωστών προβλέψεων δια του αθροίσματος της κάθε σειράς απ' τον πίνακα 7. Αντίστοιχα, το *precision* είναι η διαίρεση των σωστών προβλέψεων διά του αθροίσματος της κάθε στήλης απ' τον πίνακα 7 και δείχνει την ικανότητα του μοντέλου να προβλέπει μόνο τα σχετικά δείγματα σε κάθε κατάσταση. Χρησιμοποιώντας το *f1-score* που είναι ο αρμονικός μέσος των δύο μπορούμε να κατατάξουμε την απόδοση του μοντέλου ανά κατάσταση, οπότε έχουμε S1> S8>S4>S6>S5>S9>S7. Για τις καταστάσεις S7 και S9 που έχουν την χαμηλότερη απόδοση παρατηρούμε ότι το μοντέλο τις προβλέπει πολύ λίγες φορές ανεξαρτήτως αν είναι σωστή ή όχι η πρόβλεψη. Αυτό συμβαίνει απ' τον τρόπο δημιουργίας των δεδομένων καθώς και οι καταστάσεις αυτές συμβαίνουν με μικρή πιθανότητα και δεν συνδέονται με μεγάλη πιθανότητα με την παρουσία κάποιας πληροφορίας πλαισίου. Στην περίπτωση της S9 (εξωτερικά ραντεβού) υπάρχει κάποια σύνδεση με τους υπαλλήλους marketing και υψηλόβαθμα στελέχη γι' αυτό και έχει καλύτερη απόδοση σε σχέση με την S7. Η κατάσταση S7 απ' τον τρόπο κατασκευής των δεδομένων είναι προσβάσιμη μόνο από τέσσερεις καταστάσεις με χαμηλές πιθανότητες. Όπως είδαμε και στην βιβλιογραφική έρευνα [8] (π.χ. προφίλ χρήστη ταχυδρόμου) και είναι εύλογα αντιληπτό τα σπάνια και ακανόνιστα μοτίβα είναι δύσκολο να προβλεφθούν με ακρίβεια από μοντέλα μηχανικής μάθησης.

Στη συνέχεια, θα παρουσιάσουμε τα αποτελέσματα του μοντέλου πρόβλεψης για τις 4 κατηγορίες πληροφορίας πλαισίου που έχουμε ορίσει (E1, E2, E3, E4). Για να γίνει ο υπολογισμός αυτός θα πρέπει να χρησιμοποιηθούν εκ νέου οι μέθοδοι για την εξαγωγή αποτελεσμάτων απ' το Σχ. 15 σε *Dataframes* που περιέχουν μόνο την εκάστοτε πληροφορία πλαισίου. Γι' αυτό το λόγο τροποποιήσαμε τα *Dataframes* εισόδου με τις επιλογές φιλτραρίσματος που παρέχει η βιβλιοθήκη *pandas*. Επίσης, χρησιμοποιήθηκε η μέθοδος *reset_index* καθώς το φιλτράρισμα άφησε δείκτες διάσπαρτους στο αρχείο με αποτέλεσμα η μέθοδος predict να δίνει σφάλμα εκτέλεσης (*key_error*). Η συγκεκριμένη υλοποίηση φαίνεται στο Σχ. 16.

```
for i in evidence:
    new_test_df = test_df[test_df["Evidence"] == i]
    new_x_test = new_test_df.drop(df_columns[1], axis=1).reset_index(drop=True)
    new_y_test = new_test_df[df_columns[1]]

    predictions = predict(new_x_test, trained_matrix)

    print('Bayesian probabilistic predictor for evidence ', i)
    print("Classification report")
    print(classification_report(new_y_test, predictions))
    print("Confusion matrix")
    print(confusion_matrix(new_y_test, predictions))
```

Σχήμα 16: Κώδικας εξαγωγής αποτελεσμάτων ανά κατηγορία πληροφορίας πλαισίου

Τα αποτελέσματα ανά κατηγορία πλαισίου φαίνονται στον πίνακα 9.

Πίνακας 8: Classification report ανά κατηγορία πλαισίου

	precision				recall				f1-score				support			
Evidence / Next State	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4	E1	E2	E3	E4
S1	0.64	0.58	0.84	1	0.82	0.64	0.76	1	0.72	0.61	0.8	1	500	533	1186	844
S4	0	0.49	0.85	0	0	0.79	0.93	0	0	0.6	0.89	0	154	542	1157	0
S5	0.49	0.36	0	0	0.87	0.52	0	0	0.63	0.42	0	0	286	228	44	0
S6	0	0	0.47	0	0	0	1	0	0	0	0.63	0	12	70	226	0
S7	0.4	0	0	0	0.13	0	0	0	0.19	0	0	0	135	116	154	0
S8	0	0	0	1	0	0	0	1	0	0	0	1	41	84	39	279
S9	0.58	0	0	0	0.34	0	0	0	0.43	0	0	0	141	212	17	0
accuracy								1	0.57	0.49	0.78	1	1269	1785	2823	1123
macro avg	0.3	0.2	0.31	1	0.31	0.28	0.38	1	0.28	0.23	0.33	1	1269	1785	2823	1123
weighted avg	0.47	0.37	0.74	1	0.57	0.49	0.78	1	0.49	0.42	0.75	1	1269	1785	2823	1123

Από τα παραπάνω αποτελέσματα παρατηρούμε ότι το μοντέλο προβλέπει με απόλυτη ακρίβεια τις περιπτώσεις όπου υπάρχει πληροφορία πλαισίου E4 (δουλείες οικίας). Όπως βλέπουμε σ' αυτήν την περίπτωση ο χρήστης μπορεί να μεταβεί μόνο σε δύο καταστάσεις S1 και S8. Αυτό είναι ο λόγος που προβλέπεται η συγκεκριμένη κατηγορία καλύτερα αφού υπάρχουν λίγες επιλογές και είναι πολύ καλά ορισμένη.

Στην περίπτωση του απλού υπαλλήλου (E3) το μοντέλο επίσης εμφανίζει μεγαλύτερη ακρίβεια απ' την συνολική του συστήματος (78% έναντι 70%) καθώς έχει καλή απόδοση στις κύριες καταστάσεις που μεταβαίνουν αυτοί οι χρήστες. Το μοντέλο αποτυγχάνει να προβλέψει τελείως τις καταστάσεις S5, S7, S8, S9 το οποίο δεν έχει μεγάλη επιρροή στην ακρίβεια του μοντέλου αφού υπάρχουν ελάχιστα δείγματα. Παρόλα αυτά δείχνει την αδυναμία του να προβλέψει μια διαφορετική μετακίνηση πέρα απ' το σπίτι, χώρος εργασίας, γεύμα στο γραφείο για έναν απλό υπάλληλο.

Στην περίπτωση του εταιρικού στελέχους (E1) και του υπαλλήλου marketing (E2) η ακρίβεια είναι χαμηλότερη απ' τη συνολική (57% και 49% αντίστοιχα έναντι 70%). Αυτό είναι αναμενόμενο με βάση τα προηγούμενα αποτελέσματα ώστε να προκύψει το συνολικό αποτέλεσμα απ' τις τέσσερις κατηγορίες. Στην πρώτη περίπτωση (E1) προβλέπονται μόνο οι καταστάσεις S1, S5,S7, S9 και στην δεύτερη (E2) μόνο οι S1, S4, S5. Ποιοτικά αυτό που μπορεί να εξαχθεί είναι ότι για τις δύο αυτές περιπτώσεις υπάρχουν όλες οι πιθανές καταστάσεις σαν αποτέλεσμα κάτι που κάνει πιο δύσκολη την πρόβλεψη τους. Με βάση το σετ δεδομένων οι ρουτίνες αυτών των χρηστών είναι πιο ευρείς και δύσκολο να προβλεφθούν. Όπως αναφέρθηκε και προηγουμένως ο μικρός αριθμός δειγμάτων στο σετ εκμάθησης κάνει την πρόβλεψη εξαιρετικά δύσκολη για οποιοδήποτε μοντέλο. Αυτό που θα μπορούσε να γίνει για αυτές τις περιπτώσεις είναι η εισαγωγή επιπλέον πληροφοριών πλαισίου στα δεδομένα ώστε να οριστεί καλύτερα κάτω από ποιες συνθήκες λαμβάνουν χώρα αυτές οι μετακινήσεις ώστε να γίνει και η αντίστοιχη πρόβλεψη.

4.2 Σύγκριση αποτελεσμάτων με πληροφορία πλαισίου και χωρίς

Στη συνέχεια θα εξάγουμε τα αποτελέσματα του μοντέλου αν αυτό βασίζεται μόνο στις πιθανότητες μετάβασης του πίνακα 3, δηλαδή χωρίς την ύπαρξη της στήλης *Evidence* στο σετ δεδομένων μας. Η υλοποίηση της μεθόδου *probs* (Σχ. 12) έχει γίνει με ευέλικτο τρόπο για να είναι χρηστική με διαφορετικό αριθμό δεδομένων εισόδου (*parent classes*) όπως παρουσιάστηκε στην ενότητα 3.3. Θα χρειαστούν κάποιες μικρές τροποποιήσεις στις μεθόδους *fit* και *predict* ώστε ο χρήστης να μπορεί απρόσκοπτα να αφαιρεί αν το επιθυμεί τις πληροφορίες πλαισίου. Αυτές ουσιαστικά είναι η αλλαγή των ορισμάτων εισόδου της probs στην συνάρτηση fit και κάποιες μικρές τροποποιήσεις και έλεγχοι στα παραγόμενα *dictionaries* ώστε να αντιμετωπίζουν σωστά και τις δύο περιπτώσεις. Οι νέες μέθοδοι

φαίνονται στα Σχ. 17 και 18, όπου έχουν υπογραμμιστεί οι αλλαγές σε σχέση με τα Σχ. 13 και 14

```
def fit(train, child, parent1, parent2=None) -> dict:
    """This function provides a dictionary with the predicted state for all data combination
     based on a train dataset. The algorithm logic is to simply select the state with the
     maximum probability from possible states in the dataset. If parent 2 is set to None then
     the fit function will only account for current state column data"""
    transition_pred = {}
    parent1_list = train[parent1].unique().tolist()
    for i in parent1_list:
        if parent2 is not None:
            parent2_list = train[parent2].unique().tolist()
            for j in parent2_list:
                transition_prob = probs(train, child, parent1_list, parent1, [i], parent2, [j])
                if sum(transition_prob.values()) != 0:
                    transition_pred.update(
                        {f"{i},{j}": list(transition_prob.keys())[list(transition_prob.values()).index(
                            max(transition_prob.values()))][12:14]})
                else:
                    transition_pred.update({f"{i},{j}": None})
        else:
            # this branch calculates probs without evidence column data
            transition_prob = probs(train, df_columns[1], states, df_columns[0], [i])
            if sum(transition_prob.values()) != 0:
                transition_pred.update(
                    {f"{i}": list(transition_prob.keys())[list(transition_prob.values()).index(
                        max(transition_prob.values()))][12:14]})
            else:
                transition_pred.update({f"{i}": None})
    return transition_pred
```

Σχήμα 17: Τροποποίηση fit function για χρήση χωρίς πληροφορία πλαισίου

```
def predict(test: pd.DataFrame, fit_matrix: dict) -> list:
    """This function provides a list with the predicted next state for each
    row of the given dataset. It uses the fit function results, that is trained on
    a different dataset, to provide predictions """
    columns = test.columns.tolist()
    predictions = []
    for i in range(len(test)):
        if len(list(fit_matrix.keys())[0]) > 2:
            key = f"{test[columns[0]][i]},{test[columns[1]][i]}"
        else:
            # runs when evidence is not accounted
            key = f"{test[columns[0]][i]}"
        predictions.append(fit_matrix.get(key))
    return predictions
```

Σχήμα 18: Τροποποίηση predict function για χρήση χωρίς πληροφορία πλαισίου

Με τις αλλαγές αυτές αρκεί μόνο η αλλαγή στα ορίσματα της μεθόδου fit για την εξαγωγή αποτελεσμάτων χωρίς την στήλη evidence. Αυτό γίνεται με την εντολή:
trained_matrix = fit(data, df_columns[1], df_columns[0], None)
όπου ο parent2 έχεε οριστεί ως τίποτα (*None*).
Χωρίς τη χρήση της στήλης evidence η έξοδος τηw μεθόδου fit γίνεται στην παρακάτω μορφή όπου κάθε αρχική κατάσταση αντιστοιχίζεται πάντα με μόνο μία επόμενη κατάσταση αφού έχουμε μόνο μία στήλη εισόδου.

```
{'S1': 'S4',
 'S4': 'S6',
 'S5': 'S1',
 'S6': 'S4',
 'S7': 'S1',
 'S8': 'S1',
```

'S9': 'S1'}
Στην έξοδο της ίδιας μεθόδου που παρατέθηκε στην ενότητα 3.3 είχαμε 30 διαφορετικούς συνδυασμούς (4*7=32–2= 30 →δύο συνδυασμοί δεν είναι δυνατοί απ' τον ορισμό του προβλήματος), το οποίο δίνει πολύ περισσότερες επιλογές στο μοντέλο πρόβλεψης.
Στους πίνακες 9 και 10 παρουσιάζεται το *classification report* και ο *confusion matrix* του μοντέλου χωρίς να λαμβάνεται υπόψη η πληροφορία πλαισίου απ' το αρχείο.

Πίνακας 9: Classification report για δεδομένα χωρίς πληροφορία πλαισίου

Predicted state	precision	recall	f1-score	support
S1	0.69	0.9	0.78	3063
S4	0.69	0.75	0.72	1853
S5	0	0	0	558
S6	0.31	1	0.47	308
S7	0	0	0	405
S8	0	0	0	443
S9	0	0	0	370
accuracy			0.63	7000
macro avg	0.24	0.38	0.28	7000
weighted avg	0.5	0.63	0.55	7000

Πίνακας 10: Confusion matrix για δεδομένα χωρίς πληροφορία πλαισίου

Actual state \ Predicted state	S1	S4	S5	S6	S7	S8	S9
S1	2748	0	0	315	0	0	0
S4	468	1385	0	0	0	0	0
S5	265	90	0	203	0	0	0
S6	0	0	0	308	0	0	0
S7	372	33	0	0	0	0	0
S8	93	261	0	89	0	0	0
S9	54	231	0	85	0	0	0

Όπως ήταν αντιληπτό απ' την έξοδο της μεθόδου *fit* και φαίνεται και στους παραπάνω πίνακες, το μοντέλο προβλέπει μόνο τις καταστάσεις S1, S4, S6. Η συνολική ακρίβεια είναι 63% η οποία δεν είναι πολύ χαμηλότερη απ' όταν χρησιμοποιείται η πληροφορία πλαισίου. Αυτό συμβαίνει γιατί οι υπόλοιπες καταστάσεις έχουν πολύ μικρό αριθμό δειγμάτων στο σετ και δεν επηρεάζουν σημαντικά την ακρίβεια. Σ' αυτή την περίπτωση γίνεται φανερό ότι η ακρίβεια δεν είναι αξιόπιστο μέτρο αξιολόγησης σε πολλές περιπτώσεις ιδιαίτερα όταν οι κλάσεις πρόβλεψης ενός προβλήματος δεν είναι ισορροπημένες σε αριθμό δειγμάτων, όπως είχε αναφερθεί και στην ενότητα 2 της βιβλιογραφικής επισκόπησης. Το μοντέλο αδυνατεί να προβλέψει εντελώς πάνω απ' τις μισές καταστάσεις του προβλήματος χωρίς αυτό να επηρεάζει σημαντικά τη συνολική ακρίβεια.

4.3 Απόδοση μοντέλου με ισορροπημένες κλάσεις πρόβλεψης

Όπως έγινε αντιληπτό απ' το τελευταίο παράδειγμα είναι σημαντικό να διερευνηθεί η απόδοση του μοντέλου όταν οι διαφορετικές καταστάσεις πρόβλεψης υποστηρίζονται από όμοιο αριθμό δειγμάτων. Σ' αυτή την περίπτωση η ακρίβεια σαν μέτρο αξιολόγησης, μπορεί να δείξει καλύτερα την απόδοση του μοντέλου σε όλες τις καταστάσεις του προβλήματος

ταξινόμησης. Εδώ, να σημειωθεί ότι και σε κανονικά σετ δεδομένων δεν αναμένεται οι καταστάσεις πρόβλεψης να είναι ισορροπημένες αφού η πλειοψηφία των τοποθεσιών χρηστών είναι συνήθως στην οικία και τον χώρο εργασίες με άλλες τοποθεσίες να έχουν μικρότερο αριθμό δειγμάτων. Η ανάλυση αυτή θα γίνει για να έχουμε με καλύτερο τρόπο την απόδοση του μοντέλου με έναν αριθμό, που θα δείχνει αυτά που εξάγονται με βαθύτερη ανάλυση απ' τους πίνακες *classification report* και *confusion matrix* (πίνακες 6,9 και 7,10 αντίστοιχα).

Το πρώτο βήμα είναι η δημιουργία ενός σετ δεδομένων όπου σε αυτή την περίπτωση ο αριθμός δειγμάτων της στήλης *Next state* που είναι η κλάση πρόβλεψης να είναι ίδιος για τις 7 καταστάσεις του προβλήματος (Πίνακας 2). Τα προηγούμενα σετ δεδομένων είχαν ίδιο αριθμό δειγμάτων για την στήλη *Current state*. Όμοια με προηγουμένως θα δημιουργηθεί ένα αρχείο με 1000 δείγματα για κάθε κατάσταση πρόβλεψης. Για να γίνει αυτό, δημιουργείται με τον ίδιο τρόπο που αναλύθηκε στην ενότητα 3.2 ένα σετ δεδομένων με 5000 δείγματα για κάθε κατάσταση της στήλης *Current state*. Αυτό έγινε για να εξασφαλιστεί ότι σε αυτό το σετ θα υπάρχουν τουλάχιστον 1000 δείγματα για κάθε κατάσταση πρόβλεψης. Όπως φαίνεται απ' την υποστήριξη δειγμάτων στον πίνακα 9 χρειάζονται παραπάνω από τριπλάσια δείγματα σε σχέση με την αρχική δημιουργία για να εξασφαλιστεί ότι η κατάσταση S6 που έχει την μικρότερη υποστήριξη θα έχει πάνω από 1000 δείγματα. Το νέο σετ δεδομένων εξάχθηκε στο αρχείο *test_data_5000.csv*.

Για την εξαγωγή του αρχείου με ισορροπημένες καταστάσεις προβλέψεων δημιουργήθηκε το αρχείο *shuffle_csv.py* όπου θα γίνει η επεξεργασία του προηγούμενο αρχείου για την εξαγωγή του αποτελέσματος, όπου χρησιμοποιούνται βασικές λειτουργίες της βιβλιοθήκης *pandas* (*iloc* [28] και *sample* [26]). Η υλοποίηση φαίνεται στο Σχ. 19 και το αποτέλεσμα εξάγεται στο αρχείο *balanced_test_1000.csv*.

```
import pandas as pd

df = pd.read_csv("test_data_5000.csv")
sample = df.sample(frac=1, random_state=1)
states = sample["Next state"].unique().tolist()
# print(states)
new_dfs = []
for i in states:
    bal = sample[sample["Next state"] == i].iloc[:1000]
    new_dfs.append(bal)

balanced = pd.concat(new_dfs, axis=0)
balanced = balanced.sample(frac=1, random_state=1)
print(balanced.describe)
print(balanced.head)

balanced.to_csv("balanced_test_1000.csv", index=False)
```

Σχήμα 19: Κώδικας δημιουργίας αρχείου με ισορροπημένες κλάσεις πρόβλεψης

Για την εξαγωγή προβλέψεων και αποτελεσμάτων με βάση αυτό το αρχείο το μόνο που απαιτείται είναι η αντικατάσταση του αρχείου που σώζεται στην μεταβλητή *test_df* στο Σχ. 15. Εδώ να τονιστεί ότι η εκμάθηση του μοντέλου εξακολουθεί να γίνεται με το ίδιο αρχείο (*generated_data_1000.csv*) που έχει δημιουργηθεί σύμφωνα με τις πιθανότητες των πινάκων 3 και 5. Το νέο αρχείο έχεις άλλες συχνότητες μετάβασης απ' τον τρόπο κατασκευής του και δεν πρέπει να χρησιμοποιηθεί για εκμάθηση του μοντέλου. Τα αντίστοιχα με την ενότητα 4.1 αποτελέσματα του μοντέλου με ισορροπημένες κλάσεις πρόβλεψης φαίνονται στους πίνακες 11 και 12.

Πίνακας 11: Classification report για δεδομένα με ισορροπημένες κλάσεις πρόβλεψης

Predicted state	precision	recall	f1-score	support
S1	0.45	0.84	0.59	1000
S4	0.44	0.81	0.57	1000
S5	0.45	0.69	0.54	1000
S6	0.86	0.72	0.79	1000
S7	0.84	0.05	0.1	1000
S8	1	0.67	0.8	1000
S9	0.78	0.14	0.24	1000
accuracy			0.56	7000
macro avg	0.69	0.56	0.52	7000
weighted avg	0.69	0.56	0.52	7000

Πίνακας 12: Confusion matrix για δεδομένα με ισορροπημένες κλάσεις πρόβλεψης

Actual state \ Predicted state	S1	S4	S5	S6	S7	S8	S9
S1	839	60	33	58	10	0	0
S4	144	815	31	0	0	0	10
S5	106	161	692	41	0	0	0
S6	0	0	278	722	0	0	0
S7	560	267	119	0	54	0	0
S8	75	165	61	0	0	669	30
S9	120	383	339	15	0	0	143

Όπως φαίνεται η ακρίβεια έπεσε στο 56% απ' το 70%. Αυτό αναμενόταν καθώς οι καταστάσεις που δεν αποδίδει καλά τα μοντέλο όπως κυρίως οι S7, S9 (αλλά και οι S5, S6) έχουν πολύ μεγαλύτερη εκπροσώπηση στο σετ δεδομένων. Για τον ίδιο λόγο έχει μειωθεί το *precision* στις καταστάσεις S1, S4 αφού υπάρχουν πολλές λανθασμένες προβλέψεις (*false positives*) λόγω του αυξημένου αριθμού δειγμάτων στις άλλες καταστάσεις. Αντίθετα το *recall* παραμένει υψηλό στις δύο αυτές καταστάσεις αφού το μοντέλο δεν ανιχνεύει σε πολλές περιπτώσεις διαφορετικές καταστάσεις ενώ στην πραγματικότητα είναι αυτές.

Με τον ισορροπημένο αριθμό δειγμάτων στις διαφορετικές καταστάσεις η ακρίβεια δείχνει την πιθανότητα του μοντέλου να προβλέψει σωστά αν επιλέξουμε τυχαία μια αρχική κατάσταση. Γι' αυτό και μπορεί να θεωρηθεί πιο αξιόπιστο σαν μετρικό μιας τιμής σ' αυτήν την περίπτωση.

Στη συνέχεια εξάγουμε τα αποτελέσματα χωρίς να λαμβάνουμε υπόψη την στήλη με την πληροφορία πλαισίου. Στον πίνακα 13 παρατίθεται συγκεντρωτικά στοιχεία για την ακρίβεια του μοντέλου με ή χωρίς πληροφορία πλαισίου και με ή χωρίς ισορροπημένο αριθμό δειγμάτων καταστάσεων.

Παρατηρούμε ότι η ακρίβεια χωρίς την πληροφορία πλαισίου έπεσε στο 38% με ισορροπημένο αριθμό δειγμάτων, μια πτώση 25% σε σχέση με το αρχικό σετ δεδομένων, η οποία είναι πολύ μεγαλύτερη απ' την αντίστοιχη πτώση στην ακρίβεια με την χρήση της πληροφορίας πλαισίου. Αυτό ήταν αναμενόμενο απ' την ανάλυση που έγινε στην ενότητα 4.2 αφού το μοντέλο μπορεί να προβλέψει μόνο τρεις καταστάσεις και αυξήθηκε ο αριθμός δειγμάτων σε όλες τις υπόλοιπες, στις οποίες το μοντέλο έχει πλήρη αδυναμία.

Πίνακας 13: Συγκεντρωτικός πίνακας ακρίβειας μοντέλου στις περιπτώσεις που αναλύθηκαν

Test data	Non-balanced classes		Balanced classes	
Evidence	With	Without	With	Without
Accuracy	0.7	0.63	0.56	0.38

5. Σύγκριση μοντέλου πρόβλεψης με άλλα κοινά χρησιμοποιούμενα μοντέλα

Όπως αναφέρθηκε στην ενότητα 2 στα προβλήματα ταξινόμησης χρησιμοποιούνται πολλά διαφορετικά μοντέλα πρόβλεψης και είναι στόχος σε κάθε έρευνα να βρεθεί αυτό με την καλύτερη απόδοση για το συγκεκριμένο πρόβλημα καθώς αυτή ποικίλει ανάλογα με την δομή των δεδομένων εισόδου και της κλάσης πρόβλεψης. Γι' αυτό το λόγο, θα γίνει στην ενότητα αυτή σύγκριση του μοντέλου που αναπτύχθηκε σ' αυτήν την εργασία που βασίζεται στα Bayesian networks και είναι ένα στατιστικό μοντέλο μάθησης με άλλα κοινά χρησιμοποιούμενα μοντέλα. Σαν σετ δεδομένων εκμάθησης θα χρησιμοποιήσουμε το ίδιο αρχείο *generated_data_1000.csv*. Σαν μετρικό αξιολόγησης θα χρησιμοποιήσουμε την ακρίβεια και γι' αυτό το λόγο το σετ δεδομένων δοκιμής θα είναι αυτό με τον ισορροπημένο αριθμό δειγμάτων στην κλάση πρόβλεψης (*balanced_test_1000.csv)*, για του λόγους που αναφέρθηκαν στην προηγούμενη ενότητα.

Στην ενότητα 2.2 αναφέρθηκαν διάφορες κατηγορίες μοντέλων πρόβλεψης για προβλήματα ταξινόμησης. Για την υλοποίηση των μοντέλων σύγκρισης θα χρησιμοποιηθούν μοντέλα που παρέχονται στη βιβλιοθήκη *scikit-learn* η οποία έχει ήδη εγκατασταθεί και χρησιμοποιήθηκαν απ' αυτήν οι μέθοδοι *classification_report* και *confusion_matrix* για την εξαγωγή αποτελεσμάτων αξιολόγησης. Αυτές θα χρησιμοποιηθούν εκ νέου για την αξιολόγηση των νέων μοντέλων. Τα μοντέλα που θα υλοποιηθούν για να γίνει η σύγκριση παρουσιάζονται συνοπτικά παρακάτω.

Αρχικά θα χρησιμοποιηθεί ένα μοντέλο *Logistic Regression*, το οποίο είναι μια συνάρτηση ταξινόμησης με ένα μοναδικό πολυωνυμικό μοντέλο. Βρίσκει τα όρια μεταξύ των διαφορετικών καταστάσεων και υπολογίζει την πιθανότητα μιας κατάστασης ανάλογα με την απόσταση απ' τα όρια [9]. Έχει χρησιμοποιηθεί ευρέως σε προβλήματα ταξινόμησης και είναι ένα γραμμικό μοντέλο. Η βιβλιοθήκη scikit-learn παρέχει ένα τέτοιο μοντέλο [29] το οποίο εύκολα μπορεί να εισαχθεί στο πρόγραμμά μας με την εντολή:
from sklearn.linear_model import LogisticRegression

Στη συνέχεια θα υλοποιηθεί και ένα μοντέλο Decision Trees, το οποίο είναι ένα δενδρικό μοντέλο, όπου κάθε κόμβος αναπαριστά μια κατάσταση προς ταξινόμηση και τα κλαδιά τις τιμές που μπορεί να λάβει. Η ταξινόμηση ξεκινά απ' τον κεντρικό κόμβο και γίνεται ανάλογα με τις τιμές των χαρακτηριστικών των κόμβων [9]. Όμοια με προηγουμένως εισάγεται το αντίστοιχο μοντέλο απ' την βιβλιοθήκη *scikit-learn* [30]με την εντολή:
from sklearn.tree import DecisionTreeClassifier

Θα χρησιμοποιηθεί επιπλέον ένα μοντέλο Kernel-nearest neighbors το οποίο αναθέτει ένα άγνωστο δείγμα εισόδου σε κ κοντινούς γείτονες με βάση αποθηκευμένα δεδομένα εκμάθησης, κοιτάει τα κ δείγματα στο σετ αναφοράς που είναι ποιο κοντά στο άγνωστο δείγμα και παίρνει απόφαση με ψηφοφορία [31]. Σ' αυτό το μοντέλο θα πρέπει να οριστεί ο αριθμός κ των γειτόνων. Εισάγεται με την εντολή:
from sklearn.neighbors import KNeighborsClassifier

Τέλος, θα χρησιμοποιηθεί και ένα μοντέλο Naïve Bayes, το οποίο είναι μια απλοποιημένη εκδοχή των *Bayesian Networks* με ένα μητρικό κόμβο και πολλά κόμβους παιδιά με την υπόθεση της ανεξαρτησίας μεταξύ των παιδιών σε σχέση με τους μητρικούς κόμβους [9]. Εισάγεται με την εντολή:
from sklearn.naive_bayes import CategoricalNB

5.1 Υλοποίηση των μοντέλων σύγκρισης

Η υλοποίηση των μοντέλων σύγκρισης έγινε στο αρχείο *other_predictors.py*. Η βιβλιοθήκη *scikit-learn* παρέχει ένα πολύ απλό τρόπο υλοποίησης για τα μοντέλα πρόβλεψης της. Αρκεί η αρχικοποίηση του αντικειμένου του μοντέλου σε μια μεταβλητή, και στη συνέχεια η

εφαρμογή των μεθόδων *fit* και *predict* στο αντικείμενο αυτό με το σετ δεδομένων εκμάθησης και δοκιμής αντίστοιχα. Η διαδικασία αυτή είναι όμοια με αυτή που υλοποιήθηκε στο μοντέλο μας και η εξαγωγή των δεδομένων απόδοσης γίνεται με όμοιο με την ενότητα 4 τρόπο. Τα παραπάνω φαίνονται στο Σχ. 20 για το μοντέλο *DecissionTreeClassifier*.

```
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
predictions = dtree.predict(X_test)

print('DecisionTreeClassifier predictor')
print("Classification report")
print(classification_report(y_test, predictions))
print("Confusion matrix")
print(confusion_matrix(y_test, predictions))
```

Σχήμα 20: Υλοποίηση και εξαγωγή αποτελεσμάτων μοντέλου πρόβλεψης απ' την βιβλιοθήκη scikit-learn

Αυτό που απομένει είναι η δημιουργία των κατάλληλων σετ δεδομένων, που θα χρησιμοποιηθούν στις παραπάνω μεθόδους. Όπως αναφέρθηκε προηγουμένως θα χρησιμοποιηθούν τα ίδια σετ δεδομένων με την ενότητα 4.3. Τα δεδομένα μας περιέχουν στις στήλες εισόδου (input features) δεδομένα με διαφορετικές κατηγορίες-καταστάσεις σε μορφή *string*. Οι αλγόριθμοι της βιβλιοθήκης scikit-learn δεν μπορούν να λειτουργήσουν με μη αριθμητικά δεδομένα στις μεταβλητές εισόδου στις περισσότερες περιπτώσεις. Γι' αυτό το λόγο πρέπει να επεξεργαστούν σε τέτοια μορφή. Μια μεταβλητή με δύο καταστάσεις εύκολα μπορεί να αναπαρασταθεί με μία δυαδική μεταβλητή όπου το 0 απεικονίζει τη μία κατάσταση και το 1 την άλλη. Αυτό μπορεί να επεκταθεί και σε περισσότερες καταστάσεις προσθέτοντας επιπλέον μεταβλητές. Η βιβλιοθήκη *pandas* παρέχει την μέθοδο get_dummies [32] η οποία κάνει την παραπάνω μετατροπή και θα χρησιμοποιηθεί για τις στήλες εισόδου *Current state* και *Evidence*. Αυτό δεν είναι απαραίτητο για τη στήλη πρόβλεψης καθώς τα μοντέλα δοκιμάστηκαν και δεν είχαν κάποιο πρόβλημα στην υλοποίηση. Αυτή η διαδικασία για την εξαγωγή των κατάλληλων *Dataframes X_train, y_train, X_test* και *y_test* που χρησιμοποιούνται στο Σχ. 20 φαίνεται στο Σχ. 21

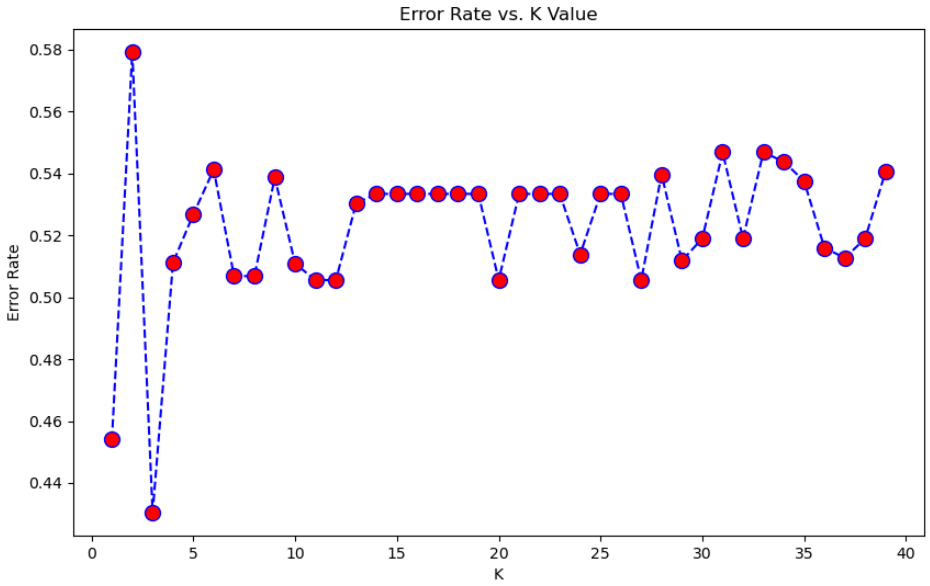
```
data = pd.read_csv("generated_data_1000.csv")
current = pd.get_dummies(data["Current state"], drop_first=True)
# print(current)
evidence = pd.get_dummies(data["Evidence"], drop_first=True)
# print(evidence)
X_train = pd.concat([current, evidence], axis=1)

y_train = data["Next state"]

test = pd.read_csv("balanced_test_1000.csv")
cur_test = pd.get_dummies(test["Current state"], drop_first=True)
ev_test = pd.get_dummies(test["Evidence"], drop_first=True)
X_test = pd.concat([cur_test, ev_test], axis=1)

y_test = test["Next state"]
```

Σχήμα 21: Μορφοποίηση δεδομένων για χρήση στα μοντέλα της scikit-learn

Με αυτόν τον τρόπο που φαίνεται στα Σχ.21 και 20 εξάγονται τα αποτελέσματα για τα μοντέλα *LogisticRegression*, *DecisionTreeClassifier* και *CategoricalNB (Naive Bayes)*. Για το μοντέλο *KNeighborsClassifier* απαιτείτε ο προσδιορισμός της μεταβλητής κ του αριθμού γειτόνων. Όσο αυξάνεται ο αριθμός αυτός αυξάνεται η πολυπλοκότητα του μοντέλου γι' αυτό πρέπει να υπολογίσουμε τον μικρότερο αριθμό που βελτιώνει σημαντικά την απόδοση του μοντέλου μας. Μια μικρή βελτίωση με μεγάλη αύξηση του αριθμού των γειτόνων είναι υπολογιστικά ασύμφορη. Γι' αυτό θα εξάγουμε τις προβλέψεις του μοντέλου για αριθμό από 1 έως 40 γείτονες και σε κάθε αριθμό θα υπολογίσουμε το μέσο σφάλμα στο σετ δεδομένων δηλαδή το ποσοστό των προβλέψεων που δεν είναι σωστές αφού έχουμε ένα πρόβλημα ταξινόμησης. Στη συνέχεια θα απεικονίσουμε το σφάλμα αυτό σε μια γραφική παράσταση χρησιμοποιώντας την βιβλιοθήκη matplotlib [23] και θα επιλέξουμε την κατάλληλη τιμή για την μεταβλητή του αριθμού γειτόνων. Η υλοποίηση αυτή φαίνεται στο Σχ. 22 και στο Σχ. 23 είναι η γραφική παράσταση του σφάλματος σε σχέση τον αριθμό γειτόνων.

```
error_rate = []

# Will take some time
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))

plt.figure(figsize=(10,6))
plt.plot(range(1, 40), error_rate, color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
plt.show()
```

Σχήμα 22: Κώδικας προσδιορισμού αριθμού γειτόνων για το μοντέλο KNeighborsClassifier



Σχήμα 23: Γραφική παράσταση μέσου σφάλματος με αριθμό γειτόνων μοντέλου KNeighborsClassifier

Απ' το Σχ.23 επιλέγουμε την τιμή κ=3 όπου έχουμε το ελάχιστο σφάλμα.

5.2 Σύγκριση αποτελεσμάτων των διαφορετικών μοντέλων

Με την διαδικασία που αναλύθηκε προηγουμένως εξάγουμε τα αποτελέσματα για τα τέσσερα μοντέλα. Στον πίνακα 14 παρουσιάζεται συγκεντρωτικά η ακρίβεια του κάθε μοντέλου συμπεριλαμβανομένους του δικού μας μοντέλου που παρουσιάστηκε στην ενότητα 3. Εδώ να σημειωθεί ότι στην περίπτωση του μοντέλου logistic regression παρόλο που έγινε ο υπολογισμός έβγαλε ένα σφάλμα ότι δεν έγινε πλήρη σύγκλιση του μοντέλου στον προκαθορισμένο αριθμό επαναλήψεων που ορίζεται. Γι' αυτό το λόγο ακολουθώντας τις οδηγίες [29] ανεβάσαμε την παράμετρο max_iter του μοντέλου από 100 (default) σε 200 και το σφάλμα λύθηκε. Η αλλαγή αυτή έγινε στην εντολή αρχικοποίησης του μοντέλου και είναι η ακόλουθη: *log_reg = LogisticRegression(max_iter=200)* . Προηγουμένως δεν υπήρχε όρισμα και χρησιμοποιούταν η προκαθορισμένη τιμή (100).

Πίνακας 14: Σύγκριση ακρίβειας διαφορετικών μοντέλων

Prediction Model	Bayesian Network	Naïve Bayes	Logistic Regression	Decision trees	K-nearest Neighbors
Accuracy	0.56	0.49	0.55	0.56	0.57

Παρατηρούμε ότι τα μοντέλα Naïve Bayes και Logistic Regression έχουν χειρότερη ακρίβεια σε σχέση με την δική μας υλοποίηση. Όπως ειπώθηκε παραπάνω η λογική των αλγορίθμων αυτών είναι σχετικά απλοϊκή και το πρώτο είναι απλοποίηση των Bayesian Network, όπου κάποιες υποθέσεις δεν ισχύουν στο πρόβλημά μας. Έτσι αναμενόταν, το *Naïve Bayes* μοντέλο να έχει χειρότερη απόδοση. Παρόλο την απλή λογική που διέπει το μοντέλο *Logistic Regression* έχει ακρίβεια πολύ κοντινή στην υλοποίηση μας. Το συγκεκριμένο πρόβλημα περιέχει μόνο δύο κλάσεις εισόδου και μία κλάση πρόβλεψης, οπότε έχει απλή σχετικά δομή και μπορούν πολλοί αλγόριθμοι να έχουν παραπλήσια απόδοση. Καθώς τα δύο μοντέλα αυτά έχουν χειρότερη απόδοση δεν θα διερευνηθούν περαιτέρω.

Τα μοντέλα Decision Tree και K-nearest Neighbors εμφανίζουν ίδια ή και ελαφρώς καλύτερη ακρίβεια γι' αυτό το λόγο θα διερευνηθούν περαιτέρω.

5.2.1 Μοντέλο K-nearest Neighbors

Αρχικά παρουσιάζονται στους πίνακες 15 και 16 το *classification report* και *confusion matrix* του μοντέλου *K-nearest Neighbors (n_neighbors =3)* όμοια με προηγουμένως.

Πίνακας 15: Classification report του K-nearest Neighbors (k=3)

Predicted state	precision	recall	f1-score	support
S1	0.49	0.82	0.62	1000
S4	0.44	0.79	0.57	1000
S5	0.6	0.3	0.4	1000
S6	0.86	0.72	0.79	1000
S7	0.84	0.05	0.1	1000
S8	1	0.67	0.8	1000
S9	0.43	0.64	0.51	1000
accuracy			0.57	7000
macro avg	0.67	0.57	0.54	7000

weighted avg	0.67	0.57	0.54	7000

Πίνακας 16: Confusion matrix του K-nearest Neighbors (k=3)

Actual state \ Predicted state	S1	S4	S5	S6	S7	S8	S9
S1	823	51	37	58	10	0	21
S4	155	785	13	0	0	0	47
S5	140	180	296	41	0	0	343
S6	0	0	0	722	0	0	278
S7	455	415	76	0	54	0	0
S8	42	116	0	0	0	669	173
S9	49	227	71	15	0	0	638

Από τα παραπάνω αποτελέσματα παρατηρείται ότι το μοντέλο αυτό αποδίδει καλύτερα για την κατάσταση S9 όπου υπάρχει σημαντική βελτίωση (σε σχέση με τον Πίνακα 12), ενώ αποδίδει χειρότερα κυρίως στην κατάσταση S5 αλλά και ελάχιστα στις καταστάσεις S1 και S4. Στις υπόλοιπες καταστάσεις έχουν ακριβώς τα ίδια αποτελέσματα. Αν εξάγουμε τα αποτελέσματα με το ίδιο μοντέλο με το αρχείο δοκιμής με τον μη ισορροπημένο αριθμό δειγμάτων (test_data_1000.csv) βλέπου με ότι η ακρίβεια είναι 68% έναντι 70% του Bayesian Network μοντέλου. Αυτό αναμένεται καθώς οι καταστάσεις που αποδίδει χειρότερα έχουν μεγαλύτερη εκπροσώπηση σε δείγματα, σε σχέση με την κατάσταση S9 που αποδίδει καλύτερα.

Στη συνέχεια αν παρατηρήσουμε το Σχ. 23 και τον τρόπο που επιλέχθηκε η τιμή του αριθμού γειτόνων βλέπουμε ότι το μοντέλο αυξάνοντας τον αριθμό συγκλίνει σε μια τιμή σφάλματος 50-51%, που σημαίνει ακρίβεια αντίστοιχα 49-50%. Αυτή η ακρίβεια είναι σημαντικά μικρότερη απ' την δική μας υλοποίηση. Τώρα για την τιμή 3 η οποία εμφανίζει το ελάχιστο σφάλμα βλέπουμε ότι καθώς δεν συγκλίνει εκεί η ακολουθία και υπάρχει μεγάλη μεταβολή απ' τον προηγούμενο και επόμενο αριθμό έχουμε μια παραλλαγή του φαινομένου *Overfitting* στα δεδομένα δοκιμής. Το *Overfitting* είναι το φαινόμενο όπου η προσπάθεια να γίνει όσο καλύτερη σύγκλιση στα δεδομένα εκμάθησης ενός προβλήματος υπάρχει το ρίσκο να γίνει προσαρμογή του μοντέλου σε θόρυβο απ' τα δεδομένα, δηλαδή η προσαρμογή σε ιδιαιτερότητες του συγκεκριμένου σετ και όχι σε γενικά μοτίβα [33]. Στην δική μας περίπτωση, εξετάζουμε το σφάλμα του μοντέλου στα δεδομένα δοκιμής για διαφορετικούς αριθμούς γειτόνων του αλγορίθμου και επιλέγουμε τον αριθμό με τον μικρότερο. Απ' την μορφή της γραφικής παράστασης του Σχ. 23 μπορούμε να βγάλουμε το συμπέρασμα ότι το μοντέλο με τον συγκεκριμένο αριθμό κάνει *overfit* στα συγκεκριμένα δεδομένα δοκιμής.

Για τους παραπάνω λόγους το συμπέρασμα είναι ότι το μοντέλο *K-nearest Neighbors* δεν μπορεί να θεωρηθεί ότι αποδίδει καλύτερα σε σχέση με το *Bayesian Network* στο συγκεκριμένο πρόβλημα αλλά και γενικότερα σε προβλήματα πρόβλεψης κινητικότητας, αφού εν γένει παρουσιάζει μικρότερη ακρίβεια και απαιτεί παραμετροποίηση (αριθμός γειτόνων) πάνω σε συγκεκριμένα τεστ δοκιμής τα οποία μπορεί να είναι διαφορετικά από τα πραγματικά δεδομένα.

5.2.2 Μοντέλο Decision Trees

Στους πίνακες 17 και 18 παρουσιάζονται το *classification report* και *confusion matrix* του μοντέλου *Decision Trees* όμοια με προηγουμένως. Παρατηρούμε άμεσα ότι οι πίνακες αυτοί είναι πανομοιότυποι με τους πίνακες 11, 12 των αποτελεσμάτων του μοντέλου Bayesian Network που υλοποιήθηκε σ' αυτήν την εργασία. Συγκρίνοντας και με το αρχείο δοκιμής με τον μη ισορροπημένο αριθμό δειγμάτων (test_data_1000.csv) παίρνουμε πάλι πανομοιότυπα αποτελέσματα. Συμπεραίνουμε ότι το παρών μοντέλο έχει ίδια ακριβώς απόδοση με τη δική μας υλοποίηση. Αυτό μπορεί να δικαιολογηθεί απ' την ομοιότητα του δεντρικού μοντέλου με κλαδιά με τον ακυκλικό γράφο που είναι τα *Bayesian Networks*.

Πίνακας 17: Classification report του μοντέλου Decision Trees

Predicted state	precision	recall	f1-score	support
S1	0.45	0.84	0.59	1000
S4	0.44	0.81	0.57	1000
S5	0.45	0.69	0.54	1000
S6	0.86	0.72	0.79	1000
S7	0.84	0.05	0.1	1000
S8	1	0.67	0.8	1000
S9	0.78	0.14	0.24	1000
accuracy			0.56	7000
macro avg	0.69	0.56	0.52	7000
weighted avg	0.69	0.56	0.52	7000

Πίνακας 18: Confusion matrix του μοντέλου Decision Trees

Actual state \ Predicted state	S1	S4	S5	S6	S7	S8	S9
S1	2497	169	142	230	25	0	0
S4	260	1499	72	0	0	0	22
S5	63	102	367	26	0	0	0
S6	0	0	82	226	0	0	0
S7	229	120	39	0	17	0	0
S8	39	88	24	0	0	279	13
S9	54	154	110	4	0	0	48

6. Συμπεράσματα

Στην εργασία αυτή αρχικά έγινε η επιλογή δημιουργίας συνθετικών δεδομένων καθώς τα δεδομένα της εργασίας αναφοράς [2] δεν ήταν διαθέσιμα και δεν βρέθηκε κάποιο άλλο δημόσια διαθέσιμο σετ δεδομένων από πραγματικές μετρήσεις από έξυπνα κινητά που είναι και ο πιο συνηθισμένος τρόπος για την συγκέντρωσή τους σύμφωνα με την βιβλιογραφία. Παρόλα αυτά δείξαμε παρά τον απλοποιημένη μορφή τους, τα συνθετικά δεδομένα που παράχθηκαν θα μπορούσαν να προκύψουν από επεξεργασία ενός πραγματικού σετ δεδομένων, οπότε μπορούν να χρησιμοποιηθούν σαν βάση ανάπτυξης ενός μοντέλου πρόβλεψης. Ένας περιορισμός των συγκεκριμένων συνθετικών δεδομένων είναι η απουσία του χρόνου σε κάθε μέτρηση, στοιχείο που είναι εύκολα διαθέσιμο στα πραγματικά σετ δεδομένων. Η προσθήκη του χρόνου θα έδινε άλλη μία μεταβλητή εισόδου στο σύστημα σε προσθήκη της τωρινής κατάστασης και της πληροφορίας πλαισίου ώστε να μπορούν να εξαχθούν περισσότερες διαφορετικές καταστάσεις και να βελτιωθεί η ακρίβεια πρόβλεψης του μοντέλου.

Υλοποιήθηκε ένα μοντέλο που βασίζεται στη λογική των *Bayesian Networks* για την πρόβλεψη της επόμενης κατάστασης του χρήστη βασιζόμενο στις πληροφορίες εισόδου. Η λογική του αλγορίθμου βασίζεται στον υπολογισμό των πιθανοτήτων μετάβασης στην κλάση στόχο του προβλήματος με βάση τις διαφορετικές κατηγορίες των δεδομένων εισόδου από τα δεδομένα εκμάθησης. Σαν πρόβλεψη επιλέγεται η κατάσταση που έχει τη μέγιστη πιθανότητα για τις εκάστοτε πληροφορίες εισόδου, δηλαδή είναι ένα μοντέλο στατιστικής μάθησης. Η υλοποίηση στην παρούσα φάση υποστηρίζει μέχρι δύο καταστάσεις εισόδου άλλα εύκολα μπορεί να επεκταθεί σε τρεις ή παραπάνω με μικρές αλλαγές στις βασικές μεθόδους *probs, fit, predict* του μοντέλου, ώστε να επιτρέψει την εισαγωγή του χρόνου ή και άλλων πληροφοριών πλαισίου. Εδώ να αναφέρουμε ότι τα μοντέλα αυτής της μορφής αλλά και οι κρυφές ακολουθίες Markov δεν επιτρέπουν τον υπολογισμό βασιζόμενα στην πρότερη ιστορία των μετακινήσεων. Η παρουσία του χρόνου στα δεδομένα χρηστών θα επέτρεπε με κατάλληλη επεξεργασία να φτιαχτούν αλληλουχίες καταστάσεων ανά χρήστη μέχρι να επαναληφθεί μια κατάσταση. Δηλαδή, αντί να έχουμε μόνο τις διακριτές καταστάσεις (π.χ. S1, S4, S5, S6, S7, S8, S9) του προβλήματος μας να έχουμε και συνδυασμούς με βάση προηγούμενες μεταβάσεις όπως S1→S4→S6 με στόχο πάντα να είναι η πρόβλεψη της επόμενης κατάστασης. Το μοντέλο μας μπορεί να λειτουργήσει σε μια τέτοια περίπτωση αφού δεν αλλάζει η φύση του προβλήματος, απλά αυξάνει ο αριθμός των διαφορετικών καταστάσεων με την προσθήκη των πολλαπλών μεταβάσεων σε αυτές. Η δυνατότητα να ληφθεί υπόψη η ιστορία πρότερων καταστάσεων θα βελτιώσει την ακρίβεια του μοντέλου αφού θα ορίζονται με περισσότερες λεπτομέρειες τα μοτίβα, με την προϋπόθεση ότι η

αλληλουχίες καταστάσεων υπάρχουν στο σετ εκμάθησης. Μια άλλη βελτίωση θα ήταν η προσαρμογή του μοντέλου ώστε να ανανεώνετε σε ροή από πραγματικά δεδομένα. Σε ένα σύστημα σε πραγματικό περιβάλλον υπάρχει η ανάγκη προσαρμογής σε νέα δεδομένα που συγκεντρώνονται απ' τους χρήστες διαρκώς. Έτσι, θα πρέπει να επανεκπαιδεύεται το μοντέλο με βάση νέα δεδομένα και πολλά παλιότερα να διαγράφονται, αφού εύλογα αντιλαμβανόμαστε ότι τα μοτίβα μετακίνησης των ανθρώπων αλλάζουν και τα παλαιότερα δεν είναι σχετικά μετά από κάποιο σημείο. Μια ακόμα επέκταση θα ήταν η δημιουργία προφίλ χρηστών και η εξαγωγή πρόβλεψης με βάση ομοιότητα σε κάποιον χρήστη εφόσον το μοντέλο αδυνατεί στον συγκεκριμένο όπως έγινε στην εργασία του Qiao, Yuanyuan, et al. [7].

Η απόδοση του μοντέλου αξιολογήθηκε σε βάθος συνολικά αλλά και για κάθε κατάσταση ξεχωριστά. Έγινε σύγκριση των αποτελεσμάτων με τις δύο στήλες εισόδου των δεδομένων, χωρίς την στήλη πληροφορίας πλαισίου και με σετ δεδομένων δοκιμής με ισορροπημένο αριθμό δειγμάτων των διαφορετικών καταστάσεων στην στήλη εξόδου. Το μοντέλο έχει ακρίβεια 56% και 70% για ισορροπημένο και μη αριθμό δειγμάτων αντίστοιχα. Όπως φαίνεται στον Πίνακα 13 η ακρίβεια του μοντέλου πέφτει σημαντικά χωρίς την στήλη πληροφορίας πλαισίου των χρηστών. Είναι αντιληπτό ότι περισσότερες πληροφορίες εισόδου μπορούν να ορίσουν καλύτερα τις διαφορετικές μεταβάσεις.

Στην ενότητα 5 έγινε σύγκριση της απόδοσης του μοντέλου της εργασίας με τέσσερα διαφορετικά μοντέλα από την βιβλιοθήκη *scikit-learn* πάνω σε κοινά δεδομένα εκμάθησης και δοκιμής. Από την σύγκριση αυτή είδαμε ότι κανένα μοντέλο δεν είχε συνολικά καλύτερη ακρίβεια απ' την υλοποίηση μας και βρέθηκε ότι το μοντέλο που βασίζεται σε αλγόριθμο *Decision Tree* έχει πανομοιότυπη απόδοση με το δικό μας. Επιπλέον, αναλύθηκε γιατί το μοντέλο *K-nearest Neighbors* παρότι φαινομενικά σε μία περίπτωση είχε ελαφρώς καλύτερη ακρίβεια, αυτό οφείλεται στο φαινόμενο *overfitting* και η συνολική του απόδοση είναι χειρότερη απ' την υλοποίηση της εργασίας. Από τα παραπάνω επιβεβαιώθηκε ότι τα Bayesian Networks και τα στατιστικά μοντέλα μάθησης εν γένει εμφανίζουν καλύτερη απόδοση σε προβλήματα πρόβλεψης κινητικότητας γι' αυτό και χρησιμοποιούνται ευρέως απ' τους ερευνητές, όπως φάνηκε απ' την βιβλιογραφική έρευνα.

Ο κώδικας υλοποίησης του μοντέλου είναι διαθέσιμος ολόκληρος στο παράρτημα Α της εργασίας και στο δημόσιο αποθετήριο https://github.com/antonis-adraktas/mobility_predictions , όπου μπορούν να γίνουν προτάσεις για βελτιώσεις και προσθήκες.

7. Βιβλιογραφικές αναφορές

[1] T.-M. Grønli, G. Ghinea and M. Younas, "Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing," *Personal and ubiquitous computing 18.4,* (2014): 883-894.

[2] N. Garg, S. K. Dhurandher, P. Nicopolitidis and J. S. Lather, "Efficient mobility prediction scheme for pervasive networks," *International Journal of Communication Systems 31.6,* (2018): e3520.

[3] . H. H. Rashidi, N. K. Tran, E. V. Betts, L. P. Howell and R. Green, "Artificial intelligence and machine learning in pathology: the present landscape of supervised methods," *Academic pathology 6,* (2019): 2374289519873088.

[4] Y. Roh , G. Heo and S. E. Whang, "A survey on data collection for machine learning: a big data-ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering,* (2019).

[5] V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser and P. Thiran, "Where to go from here? Mobility prediction from instantaneous information," *Pervasive and Mobile Computing 9.6,* (2013): 784-797.

[6] P. Wang, S. Wu, H. Zhang και F. Lu, «Indoor location prediction method for shopping malls based on location sequence similarity,» *ISPRS International Journal of Geo-Information 8,11,* (2019): 517.

[7] Y. Qiao, Z. Si, Y. Zhang, F. B. Abdesslem, X. Zhang and J. Yang, "A hybrid Markov-based model for human mobility prediction," *Neurocomputing 278,* (2018): 99-109.

[8] Q. Lv, Y. Qiao, N. Ansari, J. Liu και J. Yang, «Big data driven hidden Markov model based individual mobility prediction at points of interest,» *IEEE Transactions on Vehicular Technology 66.6,* (2016): 5204-5216.

[9] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT) 48.3,* (2017): 128-138.

[10] J. Petzold, A. Pietzowski, F. Bagci, W. Trumler and T. Ungerer, "Prediction of indoor movements using bayesian networks," *International Symposium on Location-and Context-Awareness. Springer, Berlin, Heidelberg,* 2005.

[11] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Hidden_Markov_model. [Accessed 12 1 2022].

[12] M. Hossin and M. Sulaiman, "A review on evaluation metrics for data classification evaluations," *International Journal of Data Mining & Knowledge Management Process 5.2,* (2015): 1.

[13] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix. [Accessed 13 1 2022].

[14] H. Asri, H. Mousannif, H. Al Moatassime and T. Noel, "Using machine learning algorithms for breast cancer risk prediction and diagnosis," *Procedia Computer Science 83,* (2016): 1064-1069.

[15] "python.org," [Online]. Available: https://www.python.org/downloads/. [Accessed 14 1 2022].

[16] "python.org," [Online]. Available: https://docs.python.org/3/library/venv.html. [Accessed 14 1 2022].

[17] "pandas.pydata.org," [Online]. Available: https://pandas.pydata.org/docs/getting_started/index.html#getting-started. [Accessed 14 1 2022].

[18] "numpy.org," [Online]. Available: https://numpy.org/install/. [Accessed 14 1 2022].

[19] "pandas.pydata.org," [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. [Accessed 14 1 2022].

[20] "numpy.org," [Online]. Available: https://numpy.org/doc/stable/reference/random/generated/numpy.random.choice.html. [Accessed 14 1 2022].

[21] "py-bbn," [Online]. Available: https://py-bbn.readthedocs.io/. [Accessed 14 1 2022].

[22] "networkx.org," [Online]. Available: https://networkx.org/. [Accessed 17 1 2022].

[23] «matplotlib.org,» [Ηλεκτρονικό]. Available: https://matplotlib.org/. [Πρόσβαση 17 1 2022].

[24] "scikit-learn.org," [Online]. Available: https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py. [Accessed 17 1 2022].

[25] J. Tan, J. Yang, S. Wu, G. Chen and J. Zhao, "A critical look at the current train/test split in machine learning," *arXiv preprint arXiv:2106.04525*, (2021).

[26] [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html. [Accessed 19 1 2022].

[27] [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics. [Accessed 17 1 2022].

[28] [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html. [Accessed 19 1 2022].

[29] "scikit-learn logistic regression," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accessed 19 1 2022].

[30] "scikit learn Decision Tree Classifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accessed 19 1 2022].

[31] K. Yu, L. Ji and X. Zhang, "Kernel nearest-neighbor algorithm," *Neural Processing Letters 15.2*, (2002): 147-156.

[32] "pandas get dummies," [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html. [Accessed 19 1 2022].

[33] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM computing surveys (CSUR) 27.3*, (1995): 326-327.

8. Παράρτημα Α: Κώδικας Υλοποίησης

8.1 Αρχείο create_data.py

```
import numpy as np
import pandas as pd


class GenerateData:
    column_names = ["Current state", "Next state"]
    states = ["S1", "S4", "S5", "S6", "S7", "S8", "S9"]
    evidence_dict = {
        "E1": "High rank employee",
        "E2": "Marketing employee",
        "E3": "Other employee",
        "E4": "Household work",
    }
    states_dict = {
        "S1": "Person is at home",
        "S4": "In office (cooperate)",
        "S5": "In head office",
        "S6": "Coming to office for lunch",
        "S7": "Attending some party or get together",
        "S8": "Leaving early from office for outside work",
        "S9": "In meeting outside office"
    }
    transition_probs = {
        "S1": [0, 0.8, 0.1, 0, 0.03, 0, 0.07],
        "S4": [0.3, 0, 0.2, 0.3, 0, 0.10, 0.10],
        "S5": [0.5, 0.2, 0, 0, 0.2, 0.1, 0],
        "S6": [0, 0.6, 0, 0, 0, 0.25, 0.15],
        "S7": [0.85, 0.05, 0.05, 0, 0, 0, 0.05],
        "S8": [0.9, 0, 0, 0, 0.1, 0, 0],
        "S9": [0.55, 0.2, 0.2, 0, 0.05, 0, 0]
    }

    def __init__(self, num_samples: int):
        """This function will initialize a GenerateDara object, which
contains a pandas Dataframe with the data created.
        This is synthetic data for  mobility prediction based on the
states and probabilities presented on paper
        Efficient mobility prediction scheme for pervasive networks.
        num_samples defines the number of samples to be produced for each
state.
        """
        self.num_samples = num_samples

        current_state = []
        next_state = []

        for i in self.states:
            for k in range(num_samples):
                current_state.append(i)
                if i in self.transition_probs.keys():
                    next_state.append(np.random.choice(self.states,
p=self.transition_probs[i]))
                else:
```

```
                    next_state.append(np.nan)

        variables = [current_state, next_state]
        self.df = pd.DataFrame(variables).transpose()
        self.df.columns = self.column_names

        self.column_names.append("Evidence")
        self.df[self.column_names[2]] = None

    def add_evidence(self, state1: str, state2: str, evidence: list,
probability: list):
        """This function adds evidence data on the given transition from
state1 to state2
        based on the evidence and probability lists provided"""
        for i in range(len(self.df)):
            if self.df[self.column_names[0]][i] == state1 and
self.df[self.column_names[1]][i] == state2:
                self.df[self.column_names[2]][i] =
np.random.choice(evidence, p=probability)

    def fill_evidence(self):
        """This function fills the evidence column with synthetic data"""

        self.add_evidence("S1", "S4", ["E1", "E2", "E3"], [0.05, 0.25,
0.7])
        self.add_evidence("S1", "S5", ["E1", "E2", "E3"], [0.65, 0.3,
0.05])
        self.add_evidence("S1", "S7", ["E1", "E2", "E3"], [0.4, 0.4, 0.2])
        self.add_evidence("S1", "S9", ["E1", "E2", "E3"], [0.35, 0.6,
0.05])
        self.add_evidence("S4", "S1", ["E1", "E2", "E3"], [0.03, 0.2,
0.77])
        self.add_evidence("S4", "S5", ["E1", "E2", "E3"], [0.3, 0.6, 0.1])
        self.add_evidence("S4", "S6", ["E1", "E2", "E3"], [0.05, 0.25,
0.7])
        self.add_evidence("S4", "S8", ["E4", "E2", "E1"], [0.7, 0.15,
0.15])
        self.add_evidence("S4", "S9", ["E2", "E1", "E3"], [0.6, 0.35,
0.05])
        self.add_evidence("S5", "S1", ["E1", "E2", "E3"], [0.7, 0.2, 0.1])
        self.add_evidence("S5", "S4", ["E1", "E2", "E3"], [0.3, 0.4, 0.3])
        self.add_evidence("S5", "S7", ["E1", "E2", "E3"], [0.4, 0.3, 0.3])
        self.add_evidence("S5", "S8", ["E4", "E1", "E2"], [0.6, 0.2, 0.2])
        self.add_evidence("S6", "S4", ["E3", "E1", "E2"], [0.8, 0.05,
0.15])
        self.add_evidence("S6", "S8", ["E4", "E1", "E2", "E3"], [0.65,
0.05, 0.15, 0.15])
        self.add_evidence("S6", "S9", ["E1", "E2", "E3"], [0.35, 0.55,
0.1])
        self.add_evidence("S7", "S1", ["E1", "E2", "E3"], [0.1, 0.2, 0.7])
        self.add_evidence("S7", "S4", ["E1", "E2", "E3"], [0.0, 0.4, 0.6])
        self.add_evidence("S7", "S5", ["E1", "E2"], [0.75, 0.25])
        self.add_evidence("S7", "S9", ["E1", "E2"], [0.6, 0.4])
        self.add_evidence("S8", "S1", ["E4", "E3", "E2", "E1"], [0.7,
0.22, 0.05, 0.03])
        self.add_evidence("S8", "S7", ["E3", "E2", "E1"], [0.6, 0.15,
0.25])
        self.add_evidence("S9", "S1", ["E3", "E2", "E1", "E4"], [0.3, 0.2,
0.1, 0.4])
```

```
        self.add_evidence("S9", "S4", ["E3", "E2", "E1"], [0.25, 0.6,
0.15])
        self.add_evidence("S9", "S5", ["E3", "E2", "E1"], [0.05, 0.3,
0.65])
        self.add_evidence("S9", "S7", ["E3", "E2", "E1"], [0.1, 0.4, 0.5])


data = GenerateData(1000)
data.fill_evidence()
data.df.sample(frac=1, random_state=1).to_csv("test_data_1000.csv",
index=False)
```

8.2 Αρχείο shuffle_csv.py

```
import pandas as pd

df = pd.read_csv("test_data_5000.csv")
sample = df.sample(frac=1, random_state=1)
states = sample["Next state"].unique().tolist()
# print(states)
new_dfs = []
for i in states:
    bal = sample[sample["Next state"] == i].iloc[:1000]
    new_dfs.append(bal)

balanced = pd.concat(new_dfs, axis=0)
balanced = balanced.sample(frac=1, random_state=1)
print(balanced.describe)
print(balanced.head)

balanced.to_csv("balanced_test_1000.csv", index=False)
```

8.3 Αρχείο bayesian_predictor.py

```
import pandas as pd
from create_data import GenerateData
from sklearn.metrics import classification_report, confusion_matrix

# generate = GenerateData(1000)
# generate.fill_evidence()
data = pd.read_csv("generated_data_1000.csv")
df_columns = data.columns.tolist()
states = data[df_columns[0]].unique().tolist()
evidence = data[df_columns[2]].unique().tolist()


# Define a function for printing marginal probabilities
def probs(df, child, childbands,
          parent1=None, parent1bands=None,
          parent2=None, parent2bands=None,
          parent3=None, parent3bands=None,
          parent4=None, ):
    """This function provides a dictionary with the marginal probabilities
for a classification
     problem with up to 3 parents. The keys of the dictionary provide info
on what the exact column
```

```
      value is in a {child}: {val4}, {parent1}: {val}, {parent2}: {val2},
{parent3}: {val3} format"""
    # Initialize empty list
    prob = {}
    if parent1 is None:
        # Calculate probabilities
        for val in childbands:
            key = f"{child}: {val}"
            value = df[child].tolist().count(val)
            prob.update({key: value})
    elif parent1 is not None:
        # Check if parent2 exists
        if parent2 is None:
            # Calculate probabilities
            for val in parent1bands:
                for val2 in childbands:
                    key = f"{child}: {val2}, {parent1}: {val}"
                    value = df[df[parent1] ==
val][child].tolist().count(val2)
                    prob.update({key: value})
        elif parent2 is not None:
            # Check if parent3 exists
            if parent3 is None:
                # Calculate probabilities
                for val in parent1bands:
                    for val2 in parent2bands:
                        for val3 in childbands:
                            key = f"{child}: {val3}, {parent1}: {val},
{parent2}: {val2}"
                            value = df[(df[parent1] == val) & (df[parent2]
== val2)][child].tolist().count(val3)
                            prob.update({key: value})
            elif parent3 is not None:
                # Check if parent4 exists
                if parent4 is None:
                    # Calculate probabilities
                    for val in parent1bands:
                        for val2 in parent2bands:
                            for val3 in parent3bands:
                                for val4 in childbands:
                                    key = f"{child}: {val4}, {parent1}:
{val}, {parent2}: {val2}, {parent3}: {val3}"
                                    value = df[(df[parent1] == val) &
(df[parent2] == val2) & (df[parent3] == val3)][child].tolist().count(val4)
                                    prob.update({key: value})
    sum_values = sum(prob.values())
    for i in prob.keys():
        if sum_values != 0:
            norm_value = prob.get(i)/sum_values
            prob.update({i: norm_value})
        else:
            prob.update({i: 0})
    return prob


def fit(train, child, parent1, parent2=None) -> dict:
    """This function provides a dictionary with the predicted state for
all data combination
     based on a train dataset. The algorithm logic is to simply select the
```

```
state with the
     maximum probability from possible states in the dataset. If parent 2
is set to None then
     the fit function will only account for current state column data"""
    transition_pred = {}
    parent1_list = train[parent1].unique().tolist()
    for i in parent1_list:
        if parent2 is not None:
            parent2_list = train[parent2].unique().tolist()
            for j in parent2_list:
                transition_prob = probs(train, child, parent1_list,
parent1, [i], parent2, [j])
                if sum(transition_prob.values()) != 0:
                    transition_pred.update(
                        {f"{i},{j}":
list(transition_prob.keys())[list(transition_prob.values()).index(
                            max(transition_prob.values()))][12:14]})
                else:
                    transition_pred.update({f"{i},{j}": None})
        else:
            # this branch calculates probs without evidence column data
            transition_prob = probs(train, df_columns[1], states,
df_columns[0], [i])
            if sum(transition_prob.values()) != 0:
                transition_pred.update(
                    {f"{i}":
list(transition_prob.keys())[list(transition_prob.values()).index(
                        max(transition_prob.values()))][12:14]})
            else:
                transition_pred.update({f"{i}": None})
    return transition_pred


# print(probs(data, df_columns[1], states, df_columns[0], ["S1"],
df_columns[2], ["E1"]))


def predict(test: pd.DataFrame, fit_matrix: dict) -> list:
    """This function provides a list with the predicted next state for
each
    row of the given dataset. It uses the fit function results, that is
trained on
    a different dataset, to provide predictions """
    columns = test.columns.tolist()
    predictions = []
    for i in range(len(test)):
        if len(list(fit_matrix.keys())[0]) > 2:
            key = f"{test[columns[0]][i]},{test[columns[1]][i]}"
        else:
            # runs when evidence is not accounted
            key = f"{test[columns[0]][i]}"
        predictions.append(fit_matrix.get(key))
    return predictions


trained_matrix = fit(data, df_columns[1], df_columns[0], df_columns[2]) #
with evidence
# trained_matrix = fit(data, df_columns[1], df_columns[0], None)   #
without evidence
```

```
print(trained_matrix)
test_df = pd.read_csv("balanced_test_1000.csv")
# print(test_df.head)
X_test = test_df.drop(df_columns[1], axis=1)
# print(X_test)
y_test = test_df[df_columns[1]]

pred = predict(X_test, trained_matrix)
print('Bayesian probabilistic predictor')
print("Classification report")
print(classification_report(y_test, pred))
print("Confusion matrix")
print(confusion_matrix(y_test, pred))


for i in evidence:
    new_test_df = test_df[test_df["Evidence"] == i]
    new_x_test = new_test_df.drop(df_columns[1],
axis=1).reset_index(drop=True)
    new_y_test = new_test_df[df_columns[1]]

    predictions = predict(new_x_test, trained_matrix)

    print('Bayesian probabilistic predictor for evidence ', i)
    print("Classification report")
    print(classification_report(new_y_test, predictions))
    print("Confusion matrix")
    print(confusion_matrix(new_y_test, predictions))
```

8.4 Αρχείο pybbn_code_appendix.py

```
import pandas as pd
import numpy as np
import networkx as nx # for drawing graphs
import matplotlib.pyplot as plt # for drawing graphs
# for creating Bayesian Belief Networks (BBN)
from pybbn.graph.dag import Bbn
from pybbn.graph.edge import Edge, EdgeType
from pybbn.graph.jointree import EvidenceBuilder
from pybbn.graph.node import BbnNode
from pybbn.graph.variable import Variable
from pybbn.pptc.inferencecontroller import InferenceController

# This file is used for visualization of Bayesian network, see comment for
error at the end of file
data = pd.read_csv("generated_data_1000.csv")
df_columns = data.columns.tolist()
states = data[df_columns[0]].unique().tolist()


def get_state_paths(state) -> list:
    """Returns a sorted list of the possible next states from the current
state"""
    return sorted(data[data[df_columns[0]] ==
state][df_columns[1]].unique().tolist())
```

```
def get_prob_transition_per_node(state) -> list:
    """Returns a list of probabilities transition based on the available
paths for each state"""
    paths = get_state_paths(state)
    transition_freq = []
    for i in paths:
        transition_freq.append(data[(data[df_columns[0]] == state) &
(data[df_columns[1]] == i)][df_columns[0]].count())
    return [float(x)/sum(transition_freq) for x in transition_freq]


bbn = Bbn()
nodes = []


def create_nodes():
    count = 0
    for i in states:
        nodes.append(BbnNode(Variable(count, i, get_state_paths(i)),
get_prob_transition_per_node(i)))
        bbn.add_node(nodes[count])
        count += 1


def add_edges():
    for i in nodes:
        node_values = i.to_dict().get('variable').get('values')
        for j in nodes:
            if j.to_dict().get('variable').get('name') in node_values:
                bbn.add_edge(Edge(i, j, EdgeType.DIRECTED))


create_nodes()
add_edges()
print(bbn.edges.keys())
for i in nodes:
    print(i)


def display_graph():
    # Set node positions
    pos = {0: (-2, 2), 1: (-2, 0), 2: (-0.5, -2), 3: (0, 3), 4: (2, 2), 5:
(1.5, 0), 6: (2, -2)}

    # Set options for graph looks
    options = {
        "font_size": 16,
        "node_size": 3000,
        "node_color": "white",
        "edgecolors": "black",
        "edge_color": "red",
        "linewidths": 4,
        "width": 5, }

    # Generate graph
    n, d = bbn.to_nx_graph()
    nx.draw(n, with_labels=True, labels=d, pos=pos, **options)

    # Update margins and print the graph
```

```
    ax = plt.gca()
    ax.margins(0.10)
    plt.axis("off")
    plt.show()


display_graph()
# # Convert the BBN to a join tree
# This Inference controller function gives an error "list index out of
range".
# It requires all sample combinations to be present which by definition
are not present in our case.
# join_tree = InferenceController.apply(bbn)
```

8.5 Αρχείο other_predictors.py

```
import pandas as pd
import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import CategoricalNB
import matplotlib.pyplot as plt

data = pd.read_csv("generated_data_1000.csv")
current = pd.get_dummies(data["Current state"], drop_first=True)
# print(current)
evidence = pd.get_dummies(data["Evidence"], drop_first=True)
# print(evidence)
X_train = pd.concat([current, evidence], axis=1)

y_train = data["Next state"]

test = pd.read_csv("balanced_test_1000.csv")
cur_test = pd.get_dummies(test["Current state"], drop_first=True)
ev_test = pd.get_dummies(test["Evidence"], drop_first=True)
X_test = pd.concat([cur_test, ev_test], axis=1)

y_test = test["Next state"]

# log_reg = LogisticRegression(max_iter=200)
# log_reg.fit(X_train, y_train)
# predictions = log_reg.predict(X_test)

# naive_bayes = CategoricalNB()
# naive_bayes.fit(X_train, y_train)
# predictions = naive_bayes.predict(X_test)

# knn = KNeighborsClassifier(n_neighbors=3)
# knn.fit(X_train, y_train)
# predictions = knn.predict(X_test)

# error_rate = []
#
# # Will take some time
# for i in range(1, 40):
#     knn = KNeighborsClassifier(n_neighbors=i)
```

```
#     knn.fit(X_train, y_train)
#     pred_i = knn.predict(X_test)
#     error_rate.append(np.mean(pred_i != y_test))
#
# plt.figure(figsize=(10,6))
# plt.plot(range(1, 40), error_rate, color='blue', linestyle='dashed',
marker='o',
#          markerfacecolor='red', markersize=10)
# plt.title('Error Rate vs. K Value')
# plt.xlabel('K')
# plt.ylabel('Error Rate')
# plt.show()

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
predictions = dtree.predict(X_test)

print('DecisionTreeClassifier predictor')
print("Classification report")
print(classification_report(y_test, predictions))
print("Confusion matrix")
print(confusion_matrix(y_test, predictions))
```

ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

HELLENIC OPEN UNIVERSITY
SCHOOL OF SCIENCES AND TECHNOLOGY

WWW.EAP.GR