Η εργασία υλοποιήθηκε από τους
Περικλής Μπαρμπάκος: 1115201300111
Αλέξανδρος Ράντος-Χαρισόπουλος: 1115201300147

Εργαστήκαμε σε jupyter notebook με μορφή αρχείου .ipynb Μέσα στο φακέλο μας υπάρχει ένα pdf που το περιέχει σε screenshot.

Οπτικοποίηση Δεδομένων :

Αρχικά φτιάχνουμε ένα dataframe για όσα είναι good και ένα για όσα είναι bad.Έπειτα ξεφορτωνάμαστε όλα τα numerical attributes γιατί αρχικά θα δουλέψουμε με τα categorical.
Με τη βοήθεα του DictVectorizer φτιάχνουμε dataframes που έχουν σαν στήλες κάθε πιθανή τιμή attribute με τιμές 1 ή 0 για κάθε σειρά που δηλώνει ύπαρξη ή μη.
Έπειτα ξεκινάμε με μια for loop να ερευνούμε ξεχωριστά κάθε feature φτιάχνοντας ένα counter κι εξετάζοντας τις στήλες που αφορούν αυτό(Attributei,Ai…)
Χρησιμοποιώντας από collections τους counter μετράμε πόσες φορές εμφανίζεται κάθε attribute και φτιάχνουμε τα bar plots για good και bad με άλλα χρώματα.
Ο κώδικας για αυτό το κομμάτι ξεχωριστά υπάρχει
στο αρχείο ‘Categorical Visualization.py’ ενώ τα αποτελέσματα εμφανίζονται και στο ‘Categorical Visualization.pdf’

Ανάλογα εργαζόμαστε για να απομονώσουμε τα numerical attributes έχοντας πάλι ξεχωριστά good και bad και επεξραζόμενοι κατάλληλα τα χρώματα και τα γραφήματα φτιάχνουμε τα boxplots.
‘Numerical Visualization.py’ και Numerical Visualization.pdf’

Classification :
Την μετατροπή των categorical columns του dataset την κάναμε με το df_dummies().Έπειτα το μετατρέπουμε σε πίνακα έτσι ώστε να είναι σε αποδεκτή μορφή από τους classifiers.Μοιράζουμε τα δεδομένα με 10-fold μέθοδο(ShuffleSplit ώστε να μοιράσουμε τυχαία τα δεδομένα και όχι με τη σειρά που είναι στο dataframe) και στο training και στο test dataset.Κάνουμε fit την εκάστοτε μέθοδο και έπειτα predict στο κομμάτι που τεστάρουμε(1/10 του συνολικού).

Όπως φαίνεται και στα αρχεία εξόδου η καλύτερη προσέγγιση γινόταν με τον RandomForest.Επομένως μαυτόν κάναμε και τις προβλέψεις στο αρχείο testSet_categories.csv

Κώδικας στο αρχείο Classification.py ενώ τα αρχεία είναι το παραπάνω και το EvaluationMetric_10fold.csv.

Information gain : Αρχικά ορίζουμε μια συνάρτηση η οποία υπολογίζει την εντροπία ενός δοθέντος dataframe για την περίπτωση μας στην ουσία με τον γνωστό τύπο.Μετράμε πιθανότητα του good και του bad και εφαρμόζουμε τη θεωρία.
Έπειτα φτιάχνουμε μια συνάρτηση info_sum_attribute η οποία στον τύπο I(T,a)=H(T)-H(T/a) υπολογίζει στην ουσία την ποσότητα H(T/a).Δηλαδή το άθροισμα που δίνει το

$$-\sum_{v \in vals(a)} \frac{|\{\mathbf{x} \in T | x_a = v\}|}{|T|} \cdot H(\{\mathbf{x} \in T | x_a = v\})$$

Αρχικά απομονώνουμε το attribute μαζί με το Label και με τη βοήθεια του value counts λαμβάνουμε τις τιμές των attributes και τον αριθμό των εμφανίσεων τους.Για κάθε πιθανή τιμή μετράμε την εντροπία του dataframe που εμφανίζεται μόνο αυτό το attribute και με το κλάσμα των εμφανίσεων του προς το μέγεθος του dataset λαμβάνουμε την ποσότητα που πρέπει να προσθέσουμε για αυτή την τιμή του attribute.Ακολουθούμε την ίδια διαδικασία για κάθε πιθανή τιμή που μας έδωσε η value_counts και έξω από την συνάρτηση στο κύριο κομμάτι απλά αφαιρούμε τον αριθμό που επιστράφηκε από την συνολική εντροπία για να πάρουμε το info_gain και να το κάνουμε append στη λίστα που τα κρατάμε.

Αντίστοιχα φτιάχνουμε άλλη συνάρτηση(info_num_sum) για τα numerical attributes που υπολογίζει την ίδια ποσότητα.Εδώ εκμεταλλευόμαστε την numpy.histogram η οποία μας επιστρέφει μια λίστα που κόβει σε 5 bins την ποσότητα στην οποία διανέμονται οι numerical τιμές του column αλλά και πόσες εμφανίσεις τιμών έχουμε στο κάθε διάστημα.Αυτός είναι και ο αριθμός που χρειαζόμαστε για τον τύπο, ενώ για να βρούμε τις εντροπίες καθενός dataset που έχει attributes ανάμεσα στο εκάστοτε διάστημα ελέγχουμε οι τιμές να ανήκουν ανάμεσα σε αυτές της λίστας bins του np.histogram. Στη συνέχεια η διαδικασία είναι η ίδια.
Κάποιες πληροφορίες σε σχέση με τα παραπάνω βρίσκονται στο ‘InfoGain+Bins.png’ που είναι μια εκτύπωση μας για τη συνολική εντροπία του dataset,τη λίστα που μας έδωσε η np.histogram ώστε να χωρίσουμε σε 5 bins τα στοιχεία,και τέλος η λίστα με τους αριθμούς information gain καθενός feature.

Classification με την αφαίρεση feature

Αρχικά γίνεται ταξινόμηση της λίστας infogain από το μικρότερο στο μεγαλύτερο ώστε να αφαιρούμε κάθε φορά το πιο μικρό για να τρέξουμε τον classifier.Έπειτα ακολουθούμε πάλι τη διαδικασία του 10-Kfold Validation με το ShuffleSplit κάνοντας κάθε φορά drop ένα feature και κρατώντας το εκάστοτε mean accuracy σε μια λίστα.Με τη βοήθεια αυτής κάνουμε plot το αποτέλεσμα που δείχνει τη μεταβολή του accuracy μετά την αφαίρεση κάθε feature.

Παράγουμε ένα αρχείο ‘FeatureSelection+InfoGain.csv' που βγάζει σε πινακάκι ποιο feature αφαιρέσαμε κατά σειρά και το Information Gain του αντίστοιχα.
Το plot βρίσκεται στο αρχείο ‘FeatureSelection.png’

Τέλος,έχοντας κρατήσει το accuracy του καλύτερου ταξινομητή,το συγκρίνουμε με το μεγαλύτερο που βγήκε στην παραπάνω διαδικασία της αφαίρεσης χαρακτηριστικών. Σε περίπτωση που έχουμε βρει κάπου μεγαλύτερο,ξανακάνουμε τις προβλέψεις για το test.tsv κάνοντας drop στα ίδια χαρακτηριστικά που είχαμε κάνει και σε εκείνο το σημείο.

Για ασφάλεια φτιάχνουμε και 2ο αρχείο με τα testSet categories που αφορά την παραπάνω διαδικασία με το πρόθεμα Drop_Features(...)