SUPERVISED AND UNSUPERVISED DATA MINING

TABLE OF CONTENTS

1.0. Introduction

This report covers the steps carried out in the analysis and cleaning of an unstructured Car review dataset. The dataset is made up of reviews of 10,678 customers for four different car models (Toyota, Hyundai, Ford, and Kia), with their respective car recommendation choices to other people, over a period. The aim is to develop a suitable model that will utilize the information in the dataset and classify new/future customer reviews appropriately. This information would help the car manufacturers/industry's to make informed decisions based on predicted positive and negative car recommendation reviews to others.

The dataset used in this report is stored in a csv file titled, ***"Car_Reviews.csv".*** *Relevant information about the* ***"Car_Reviews.csv"*** *dataset can be found in the appendix of this report.*

2.0. Data Cleaning/Pre-Processing

The Car reviews dataset is populated with text data in an unstructured format and must be converted to a structured format before algorithmic patterns can be extracted from it. This dataset differs from structured and semi-structured dataset because it does not hold a tabular structure nor contain markers or tags showing hierarchy or semantics. The reason for this conversion is because the algorithm needed to train the car reviews data can only be loaded with structured numerical data in organized rows and columns. Unlike numerical data, text data/categorical data is quite complex and must undergo a cleaning process before conversion can be made to a structured format. The following steps have further been taken to clean the car reviews dataset:

a) The unstructured dataset was imported into the integrated development environment as a dataframe [df] using the pandas function (*pd.read_csv*), useful for reading csv files. The pandas set option was further used to ensure all columns and column values were visible, by setting the cells to the maximum column width.

b) The column of interest ('Recommend'), showing whether a customer is willing to recommend cars for purchase or not, was further converted to a numerical feature using the map method of conversion to ensure compliance with the numeric algorithm requirement. It is worthy of note that the 'Yes' textual label in the 'Recommend' column was assigned to the positive label '1', and the 'No' textual label was assigned to the negative label '0'.

c) A cleaner function was defined for the application of different data cleaning operations on the unstructured customer reviews column. The importance of the cleaner function is to accommodate all data cleaning steps and apply the operations in its entirety to the 'Review' column, thereby creating a new dataframe holding the cleaned customer reviews.

d) The 'Review' column is passed through the *BeautifulSoup* function imported from a python package called bs4 (beautifulsoup4). This function helps to remove all html encodings from the car reviews dataset to ensure the absence of unnecessary symbols that can inhibit the smooth running of the algorithm. This function has an inbuilt html parser called *'lmxl'* used to identify and remove all html entities present in the dataset.

e) All texts are further extracted from the dataset without the html entities using a *'get_text()*' function and redefined with a new variable name.

f) The redefined data was passed through a specialized language called regular expressions (re), used to compare/match strings in a text data. The substitution method of regular expression was used to replace all URL's, hyperlinks, non-alphabetic characters such as numbers, punctuations etc., occurring multiple times within the dataset, with a whitespace. This is done to ensure the removal of non-textual characters in preparation for the data structure conversion process.

g) Upon removal of all symbols, numbers and punctuations from the dataset, all texts were further converted to lower case using the '*t.lower()'* inbuilt python function. This is to ensure uniformity and standardize all texts in the dataset. However, for these texts to be converted seamlessly, they need to be tokenized (i.e., broken down into individual words/tokens). The lowercase function will not convert sentences or string of words until they are broken to tokens. The cleaned data was passed into a natural language toolkit tokenizer *(nltk.word_tokenize())* before running it through the lower case function. The natural language toolkit also uses a sentence tokenizer called *'punkt'* in the tokenization process to identify where words or sentences end for the creation of more accurate tokens.

h) The tokenized data was further filtered to remove all stop words (i.e., common words that have a less useful meaning). The stop words were first imported in English language because the dataset is in English, and then filtered using *'lambda'* function. The '*lambda'* function returns all words except the stop words defined in the list of documented stop words downloaded using the natural language toolkit (nltk).

i) The cleaned data was passed through a process called lemmatization (removal of synonyms, plurals, and some parts of speech). This is important to ensure the algorithm treats similar words with different suffixes as one; thereby reducing word repetition and complexity. A wordnet document containing the synonyms and plural forms of different words was downloaded from natural language toolkit and used by the *'wordnetlemmatizer()'* function to compare and remove all plural forms, synonyms, and parts of speech leaving only root words (i.e., noun form of words).

j) All steps embedded in the cleaner function was applied to the entire 'Review' column and stored in a dataframe with a new variable name ('cleaned_review').

k) The null/empty rows in the cleaned dataset were further removed using the *'map(len)'* function to retain only rows with values greater than zero. This is to ensure that all empty rows resulting from substitution with whitespaces or total elimination is discarded to reduce the sparseness of the dataset.

l) The target label and features label were further defined. The target label or label of interest is the 'Recommend' column, showing whether a customer is willing to recommend a car to other people or not. The cleaned review column ('cleaned_review') was set as the features label.

3.0. TF-IDF Matrix

The Term Frequency-Inverse Document Frequency (TF-IDF) Matrix is the structured format that the cleaned car review dataset was converted to, for the purpose of model development. The dynamics of this matrix takes the product of the frequency of token values (TF) in a document and the degree of rarity of the same token (IDF) in the entire corpus. This structured format is preferred because it combines both token importance in a document and rarity of tokens in a corpus to determine token significance. The steps carried out in the creation of the TF-IDF matrix are as follows:

a) The tokens in the cleaned data were first joined together to form a sentence before passing it to the matrix creation function (*TfidfVectorizer*). The *'TfidfVectorizer'* function is a function imported from scikit learn python library used for the creation of the TF-IDF matrix. The reason for joining all individual tokens in the data is because the *'TfidfVectorizer'* does not accept stand-alone words/tokens but works excellently well with sentences of words.

b) The *'TfidfVectorizer'* is further defined with two hyperparameters needed to regulate the dataset. The '*ngram_range'* is used to specify the type of variables to be created in the TF-IDF matrix. This parameter was set to create unigrams (single words), bigrams (double words) and trigrams (triple words). This wide range of words is to ensure the accommodation of different word types within the matrix. Another parameter defined within the *'TfidfVectorizer'* function is the minimum document frequency *(min_df)* used to remove words that are least sensible or words with lower predicting power in the dataset. The purpose of this is to regulate the number of variables created in the matrix to build a simple or less complex model. The number of '*min_df*' (0.00281) defined, implies that only sensible words within 30 documents of the 10,678 documents in the dataset would be considered as a variable.

c) All sensible unigrams, bigrams and trigrams that met the '*min_df'* condition was then extracted using the *'tfidf.fit()'* function to create variables for the TF-IDF matrix.

d) The TF-IDF matrix was finally created using the *'tfidf.transform()'* function and further populated with the created TF-IDF values into a sparse matrix datatype. A new name is assigned to the matrix representing the structured data which has been converted from its unstructured format.

4.0. Hyperparameter Tuning

Two hyperparameters (*min_df* and *ngram_range)* were identified in the creation of the TF-IDF matrix and was tuned using the '*TdidfVectorizer'* function from the scikit learn library.

a) Minimum document frequency (*min_df*): This is an unknown value selected to represent a proportion of the dataset. This is used because an average unstructured dataset is made up of diverse tokens, all of which may not be useful in the creation of the TF-IDF matrix and the classification model development. Upon repetitive trials, the optimal value with sensible unigrams, bigrams and trigrams was 0.00281. This represents a strict condition of 30 out of 10,678 documents (i.e., 0.00281*10678=30), meaning all unigrams, bigrams and trigrams must be present in at least 30 documents to be considered a variable suitable for

the matrix. This hyperparameter needs to be tuned to detect the optimum number of documents containing a reasonable level of comprehensive or sensible variables. This parameter when tuned appropriately, simplifies the model, and discards irrelevant words in a document.

b) N-grams (*ngram_range*): N-gram is a sequence of words or token present in a document, where n represents a positive number from 1 to infinity. N-grams is a bag-of-words approach for creating/extracting words from a text data for the purpose of modelling. This hyperparameter is used to specify the kind of variables to create. By defining (*ngram_range*=(1,3)), the hyperparameter was tuned to create only unigrams, bigrams (sequence of two words) and trigrams (sequence of three words) in the documents. This was to allow for more options of words within the cleaned dataset.

5.0. Supervised Learning

These are learning algorithms loaded with known input and output data to learn patterns in the data, explain the relationship between the input and output data and make predictions for future data.

5.1. Model Creation, Evaluation and Selection

Two supervised learning classification models were implemented on the car review dataset. These two models were selected because they perform well on a text dataset compared to other classification models. Critical analysis of both models is further evaluated, and a preferred model is selected for the car recommendation business application.

5.1.1. Support Vector Classifier (SVC)

This classifier model was implemented because it performs excellently well on text datasets, due to its dimensionality ignorance property. It is also suitable for datasets with discrete output, which is a perfect fit for the car review dataset which has just two output classes. SVC is also preferred due to the speed employed in running the algorithm to completion. Due to decreased emphasis placed on dimensionality (number of attributes in a dataset), the model development process is not slowed down by the high dimensional nature of text data. Generally, kernel tuning is the only process that slows down the running of SVC model on a regular dataset; but for implementation on a text dataset, the kernel is not tuned. This is because the text data is already in a high dimensional form and does not need to be transformed to a higher dimension to be linearly separable as in the case of a low dimensional data.

The kernel was however set to a linear default with a low regularization parameter (C=1), permitting a lot of slack data points on the other side of the linearly separable margin for the purpose of generalization. SVC is also great at avoiding overfitting through the implementation of cross validation. A 10-split cross validation was used because of the large size of the car review dataset and for a better evaluation accuracy as evaluation on a single test set is a poor practice for model development. This split trains the data and evaluates the performance of the model on different test sets to detect whether overfitting is occurring or not. The *'stratifiedkfold'* function from scikit learn library was used to run the cross validation for the car review data by ensuring that the ratio between the two classes (training/test data

and original data) is consistently maintained; in this case, to ensure that the training set is balanced in the same ratio as the original balanced dataset. This method was used in this scenario as opposed to grid search because there is no hyperparameter tuning to be carried out under SVC. Therefore, a for loop function is used to iterate over the defined train and test set indices.

The model was trained by passing the training sets into a *'fit()'* function and then predicted using the *'predict()'* function. The predicted output was further compared with the actual output by estimating an accuracy score for each split. The criteria for selecting and minimizing the accuracy prediction error is due to the state of the car review dataset being a balanced dataset and for the purpose of maximizing prediction accuracy. The accuracy score for each split was displayed, and the mean for all 10 scores were estimated to give the final model performance which was 0.91447.

5.1.2. Naïve Bayes Classifier (NBC)

This model was implemented because it is equally suitable for a text classification problem but will perform poorly on a regular dataset and result in overfitting. It assumes that all variables in a dataset are independent of each other which is in line with the objective of developing a TF-IDF matrix, where tokens are independent of one another. This strong but naive assumption is usually not applicable in a regular dataset, as variables in these datasets are mostly related, resulting in an overall poor performance when applied. NBC is also suitable for a high dimensional data because it runs fast by using conditional probability estimations to make fast predictions. The multinomial naïve bayes function was used because it uses a multinomial distribution to represent the frequency of each token and favourable for discrete features.

Unlike SVC, NBC considers dimensionality, but the model speed is not affected by the number of attributes in a data; Hence, it performs at a reasonably high speed. NBC has no hyperparameter to tune but a 10-split cross-validation process similar to the training and testing sampling performed in the development of the SVC model was carried out. The evaluation metric and prediction error selected to be minimized in the NBC model was accuracy. This is because of the balanced dataset used in the development of this model as well as to maximize prediction accuracy. An iteration function was used for the defined test and train data and the accuracy scores for all 10 splits were computed and stored in an empty list. The mean of these scores was taken to give an average score of 0.89405.

5.1.3. Model Selection and Interpretation

Both SVC and NBC models performed well on the car review dataset with similar performance accuracy scores of 0.914 and 0.894 respectively. However, these models differ slightly with respect to certain properties. Although, they both run fast and are not affected by large number of variables, NBC is a simpler model as predictions are made using the bayes theorem probabilistic formular. The use of this formular simplifies the model engineering process needed to train the data compared to SVC. Also, NBC has a probabilistic interpretation based on its naive assumption of independence i.e., the probability of a document belonging to a class over another can be derived. This interpretability is achieved because each feature is treated independently towards the class prediction; therefore, the

extent to which each feature contributes to the class prediction in the conditional probability estimation can be explained in this model.

SVC on the other hand equally performed well but is more complex and more computationally technical to train than NBC model. SVC is a relatively non-interpretable model; therefore, inferences cannot be drawn from the data and how they contribute to the overall performance with respect to business applications.

Although, SVC has a higher performance score by a 0.02 margin more than the NBC model performance score, NBC model is the recommended model to deploy in this case, because the NBC model is simpler, performs well on multi-class problems and sentiment analysis application like the car review problem, highly scalable, not sensitive to irrelevant features and provides the car manufacturers with the option to gain insights from the data through its relative model interpretability.

5.2. Model Deployment

A limitation of deploying NBC model on new car reviews dataset is that if the new reviews obtained are not in a categorical text format i.e., a regular dataset, the independence assumption may not hold through, thereby leading to a poor performance. Also, NBC has a zero-frequency issue, where zero probability is assigned to a categorical feature in a test set that is absent in a training data. Another weakness of the NBC model is that it assumes that predictor variables especially the continuous variables are normally distributed. The implication, however, is that if the new car reviews are continuous and are not normally distributed, then the performance will plunge to a low. A general limitation of text mining on new car reviews dataset is the syntactics issue i.e., position of car reviews tokens in the data and semantics issue i.e., meaning of car reviews token in the dataset. Although, this limitation can be handled by technical tokenization processes such as deep learning etc, the engineering process becomes more complex.

6.0. Unsupervised Learning

This is a learning method where the algorithms are not provided with output, but only provided with input variables. The major function of an unsupervised learning model is data exploration and pattern extraction. Prediction is not done with these types of models as data points are not labelled.

6.1. Uniform Manifold Approximation and Projection (UMAP)

The unsupervised model used in the visualization of clusters in the car reviews dataset is the UMAP technique. This visualization technique was chosen because the car review dataset is a high dimensional text data with reasonably large number of variables. This technique is preferred to avoid overcrowding when the data is transformed and represented on a two-dimensional scatter plot compared to other visualization techniques like PCA etc. The methodology of overcrowding prevention is by the use of a fuzzy radius projected from each data point, covering ‘n’ number of data points within the radius. The probability of data points being neighbours is derived when the radii overlap each other. A fuzzy radius is preferred to a fixed or variable radius because it reduces the isolation of data points and increases the accuracy of datapoints being probable neighbours.

The UMAP visualization technique ensures that the distance between data points in its high dimensional form is accurately represented as the same distance between data points in its low-dimensional form. Therefore, data points closest to each other will have a stronger outline indicating a higher probability of being neighbours while farther points will be fuzzy with a faded outlook, showing a lesser probability of being neighbours. UMAP initially estimates probabilities of data points randomly placed on a two-dimensional space using fuzzy radii i.e., random embedding, and then compares it with the probability estimation in the high dimensional space. An optimization algorithm (stochastic gradient descent) is also run to minimize the differences between the two probabilities, which synchronizes both probabilities to have the same position represented in a scatter plot visualization.

UMAP has two hyperparameters which are *'n_neighbors'* representing the number of neighbour extensions of a radius i.e., nearest neighbours and *'min_dist'* representing the minimum distance between similar datapoints in a low-dimensional space.

a) An optimal value of 170 was set for the *'n_neighbors'* to accommodate a reasonable number of datapoints within the fuzzy radius, for the identification of clusters with similar car recommendations. By setting this *'n_neighbors'* value, the fuzzy radius was able to extend its radar to cover 170 data points leading to the probability estimation of stronger or closer datapoints. This value also resulted in the creation of a descent scatter plot visualization that was easy to read and interpret.

   A low value of 0.4 was set for the *'min_dist'* to ensure a close distance between datapoints and avoid overly spaced datapoints within the scatter plot visualization, thereby creating clusters of similar datapoints. This helps to better achieve the clustering function of the UMAP visualization technique and provides a solution to the car review problem by clustering similar recommendations together.

   The *'n_components'* was set to a value of 2 because the data is to be visualized on a two-dimensional plot having X and Y axis. The data was further transformed into two variables using the *'u.fit_transform()'* function. All three columns (Review, Recommend and Vehicle Title) were defined in a list before passing them into the *'text'* function, to ensure that all three values are displayed when data points are hovered on in the scatter plot. The scatter plot was created using plotly, with a layout width and height of 1500. These respective layout dimensions were chosen because it provided a clearer and readable visualization. All properties were passed into the *'go.Figure()'* function for display.

   Upon analysis of the scatter plot, two clusters were identified. The positive recommendation class with good customer reviews (red data points) constituting the upper half of the scatter plot and the negative recommendation class with poor customer reviews (purple data points), clustered at the lower part of the scatter plot diagram. This shows that UMAP was able to successfully cluster datapoints with similar recommendation information together. This further helps the relevant actors of the business to visualize reviews with respect to recommendation choice and make improved business decisions.

b) One sub-cluster was also identified, embedded in both the positive and negative car recommendation labels. Sub-clusters are simply subdivisions of an existing cluster, it is a clustering term that is used to describe grouping of datapoints within a large cluster.

Similar vehicle types were however found around different parts of the scatter plot clustered together. This sub cluster was identified because the vehicle type column was passed into the text function for more insights on the dataset. This however implies that selected vehicle types and models were relevant in recording similar reviews within the dataset. These sub-clusters were made possible because UMAP successfully identified similar vehicle types within each recommendation class. This information is useful to establish links between vehicle types, car reviews and possible recommendations in the future.

7.0. Conclusion

Upon critical analysis of the problem domain, the car review dataset was better suited to be trained using SVC and NBC after proper cleaning and conversion from its unstructured format to a structured format. Excellent accuracy scores were recorded from both models reflecting accurate classification of future car reviews. NBC is however recommended for the business application in this scenario, as it is simpler and a great performing model. The developed scatter plot was also able to successfully classify positive and negative customer recommendations into clusters and subclusters of vehicle type for easy visualization and interpretation.

8.0. References

livebook.manning.com. (n.d.). Chapter 6. Classifying with naive Bayes and support vector machines · Machine Learning with R, the tidyverse, and mlr. [online] Available at: https://livebook.manning.com/book/machine-learning-for-mortals-mere-and-otherwise/chapter-6/9 [Accessed 29 Mar. 2022].

Molnar, C. (n.d.). 5.7 Other Interpretable Models | Interpretable Machine Learning. [online] christophm.github.io. Available at: https://christophm.github.io/interpretable-ml-book/other-interpretable.html.

Simplilearn.com. (n.d.). Understanding Naive Bayes Classifier. [online] Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/naive-bayes-classifier.

SearchEnterpriseAI. (n.d.). What is Unsupervised Learning? [online] Available at: https://www.techtarget.com/searchenterpriseai/definition/unsupervised-learning.

upGrad blog. (2020). Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2020. [online] Available at: https://www.upgrad.com/blog/naive-bayes-explained/.

9.0. Appendix

Number of Instances = 10,678

Number of Columns = 3

S/N	Column Name	Label
1	Vehicle_Title	Ford, Hyundai, Kia, and Toyota Car Models
2	Review	Customer Reviews
3	Recommend	Customer Recommendation (Yes, No)