Clustering, Classification and Dimension Reduction Techniques on real dataset

Lakshay Talwar

26/09/2022

School of Mathematics,
Cardiff University

A dissertation submitted in partial fulfilment of the
requirements for **Data Science and Analytics**
by taught programme, supervised by **Dr. Andreas Artemiou**

Acknowledgments

I would like to thank my supervisor, Professor Dr. Andreas Artemiou, for his timely help and for pointing me towards the right direction throughout this project. I am grateful to Ms Joanna Emery for providing me with an opportunity to pursue this project. Also, I would like to acknowledge the advice and moral support provided by both of my professor.

EXECUTIVE SUMMARY

Businesses seek to engage with their customers in a way that is not constrained by time, the availability of human resources, or language. To keep their customers happy and consequently lower customer churn, they must anticipate their demands and satisfaction. Cars are an integral element of our daily life. At a point when an individual considers of buying a car, there are many aspects that could impact his/her interest. There are different selection criteria for buying a car such as price, maintenance, comfort, and safety precautions, etc. In this paper, it was applied with various data classification models to the car evaluation dataset. The model created with the training dataset has been evaluated with the standard metrics such as accuracy, precision and recall. In this project I have applied seven different classification algorithms which helps in predicting that in which condition the customer is willing to purchase the car which is also predicted by all the different classifications. Here, it is visible that the supervised algorithms like Decision Tree, Random Forest, Support Vector Machine and K-Nearest Neighbours works better. On the other hand, Naïve bayes algorithms Like Bernoulli and Gaussian Naïve Bayes algorithm only performed mediocre but Artificial Neural Network performed a little better than Naïve Bayes algorithms. My experimental results shows that Support Vector machine is the most suitable kind of algorithm for the car evaluation dataset.

1. INTRODUCTION

Cars are an integral element of our daily life. The automobile industry produces a variety of vehicles from various manufacturers. When it comes to choosing an automobile, customers have many alternatives. Customers will primarily consider pricing, safety, comfort, and how luxurious the vehicle appears when making decisions (Guo *et al.*, 2021). As far as model and manufacturer preferences go, cars have a wide range of features. When buyers are making their selections, three crucial factors—price, safety and luxury—are taken into account. These elements drastically reduce the likelihood of accidents happening. When purchasing cars, some basic equipment is also crucial to take into account. This comprises amenities, safety equipment, and performance-enhancing devices in automobiles. As was previously stated, one of the key considerations in purchasing a car is safety. The same is true for convenience, which includes features like maintenance, door, and luggage boot. Cost consideration is essential to ensuring that an automobile purchase is worthwhile for the owner. Owning a car entails financial responsibilities as well because it needs to be maintained for convenience.(Rehman *et al.*, 2018). These elements are more essential for lowering accidents. Finding appropriate algorithms that will produce the greatest novel automobile quality prediction in performance evaluation requires the employment of a variety of classification techniques (K., 2020). The car evaluation dataset in this work was subjected to a number of different data mining classification techniques.(Edgar, no date). The algorithm applied on the training dataset was assessed using industry-standard measures like accuracy, recall, and precision. A trustworthy evaluation technique benefits both consumers and producers. It lessens the load on businesses and increases sales. Additionally, it provides customers with a higher level of service in a highly competitive industry(Jain and Kr Vishwakarma, 2017). A considerable field of study now focuses on accurate analysis of the product.

2. LITERATURE REVIEW

2.1. DATASET AND MACHINE LEARNING

2.1.1. Type of Dataset:

Before making a decision, evaluating the state of the car is essential. It takes a lot of time and labour to manually distinguish between a car in good or acceptable condition and one in bad condition. Because machine learning has been demonstrating promising results in handling classification-related problems, which can use ML approaches to create an automated system for evaluating cars.(*Car Evaluation Analysis Using Decision Tree Classifier | by Mohammad Masum, PhD | Towards Data Science*, 2019). Essentially, cars are a part of our daily lives. The customers must make a choice because different manufacturers build different types of cars. When a person thinks about buying a car, there are several factors that could affect his or her decision regarding the type of car they are interested in. (*Evaluating a Car's Condition with Machine Learning - Hypi*, 2019). The car evaluation dataset is collected from UCI Machine Learning Repository and the **data source (creator)** was Marko Bohanec . It contains 1728 car sample information with 7 attributes, including one class feature that tells whether the car is in acceptable conditions (*Car Evaluation Analysis Using Decision Tree Classifier | by Mohammad Masum, PhD | Towards Data Science*, 2019). Here, the task is to predict whether a car is in an unacceptable, acceptable, good or

very good condition based on car characteristics such as the price of the car, maintenance cost, safety features, luggage space, seating capacity, and the number of doors(*Evaluating a Car's Condition with Machine Learning - Hypi*, 2019).

- **Buying price:** which corresponds to the price of the car. There are four possible values for this attribute: Very High: vhigh, High: high, Low: low, Medium: med.
- **Maintenance cost:** which stands for the maintenance cost. It can also have the same four possible values as for the buying attribute: Very High: vhigh, High: high, Low: low, Medium: med.
- **Number of doors:** corresponds to the number of doors of a car. The possible values are 2, 3, 4, 5more.
- **Number of persons**: refers to the seating capacity of a car. The possible values are 2, 4 or more.
- **Luggage boot space:** contains information about the luggage compartment, and can have small, med, and big as the possible values.
- **Safety:** corresponds to the safety rating of the car. The possible values are low, med, and high.
- **Class:** describes the manual assessment of the condition of the vehicle. Unacceptable, Acceptable, Good, and Very Good conditions are all possible for automobiles. The values are denoted by the abbreviations unacc, acc, good, and vgood. In order to make the class attribute more readable, it has been renamed to condition.

2.1.2. **Why do we differentiate in machine learning?**

Modern inventions like machine learning have improved many business and professional procedures as well as our daily lives. It's a branch of artificial intelligence (AI) that focuses on creating intelligent computer systems that can learn from databases by employing statistical approaches (*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020). Computer systems can use all the client data by using machine learning (*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020). It follows the pre-programmed instructions while also adapting to new circumstances or changes. Algorithms change as a result of data, exhibiting previously unprogrammed behaviours(*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020).

A digital assistant might scan emails and extract the crucial information if it could read and recognise context. The capacity to forecast future client behaviour is a built-in feature of this learning. This enables you to be more proactive as well as responsive to your clients' needs. A subset of machine learning is deep learning. It is essentially a three-layer artificial neural network. One-layer neural networks are capable of making approximations of predictions(*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020). The inclusion of additional layers can help with improving accuracy and optimization(*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020).

Machine learning is useful in a wide range of sectors and industries and has the potential to advance through time.(*6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog*, 2020).

Internet search engines, spam-filtering email software, websites that offer personalised recommendations, banking software that spots suspicious transactions, and many phone apps like speech recognition all employ machine learning.

There are many more potential uses for the technology, some with higher stakes than others. Future developments will have a big impact on society and could help the UK economy. Machine learning, for instance, could give us readily available "personal assistants" to help us manage our lives, it could greatly enhance the transportation system by using autonomous vehicles, and it could greatly enhance the healthcare system by enhancing disease diagnosis or personalising treatment. In security applications, machine learning might be used to examine online activity or email traffic. The ramifications of these and other technological applications must be thought through right away, and steps must be taken to assure that usage will be advantageous to the society(Stevanovic, Vlajic and An, 2013).

2.2. WHY COMPARISON IS IMPORTANT IN MACHINE LEARNING?

In applied machine learning, contrasting machine learning approaches and choosing a final model are frequent activities. Common resampling techniques for model evaluation include k-fold cross-validation, from which mean skill scores can be derived and directly compared. Although straightforward, this method may be deceptive because it is difficult to determine if the difference in mean skill scores is a true difference or a statistical fluke(Widyananda *et al.*, 2022).

In order to address this issue and estimate the possibility of the skill score samples being seen on the presumption that they were taken from the same distribution, statistical significance tests were developed. It is implied that the difference in skill scores is statistically significant if the null hypothesis, or starting point, is rejected. Statistical hypothesis testing can increase your confidence in the interpretation and the presentation of results during model selection, even though it is not full proof. (Pai and Potdar, 2017).

2.3.DIFFERENT CLASSIFICATIONS ADOPTED IN THE STUDY

On the basis of training data, the Classification algorithm is a Supervised Learning technique that is used to categorise new observations. In classification, a programme makes use of the dataset or observations that are provided to learn how to categorise fresh observations into various classes or groups. For instance, cat or dog, yes or no, 0 or 1, spam or not spam, etc. Targets, labels, or categories can all be used to describe classes. In contrast to regression, classification's output variable is a category rather than a value, such as "Green or Blue," "fruit or animal," etc. The Classification algorithm uses labelled input data because it is a supervised learning technique, therefore it comprises input and output information.(*Classification Algorithm in Machine Learning - Javatpoint*, no date)

In classification algorithm, a discrete output function(y) is mapped to input variable(x).

- y=f(x), where y = categorical output

The best example of an ML classification algorithm is **Email Spam Detector**.

The primary objective of a classification algorithm is to determine the category of a given dataset, and these algorithms are primarily employed to forecast the results for categorical data.

Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.(*Classification Algorithm in Machine Learning - Javatpoint*, 2018)

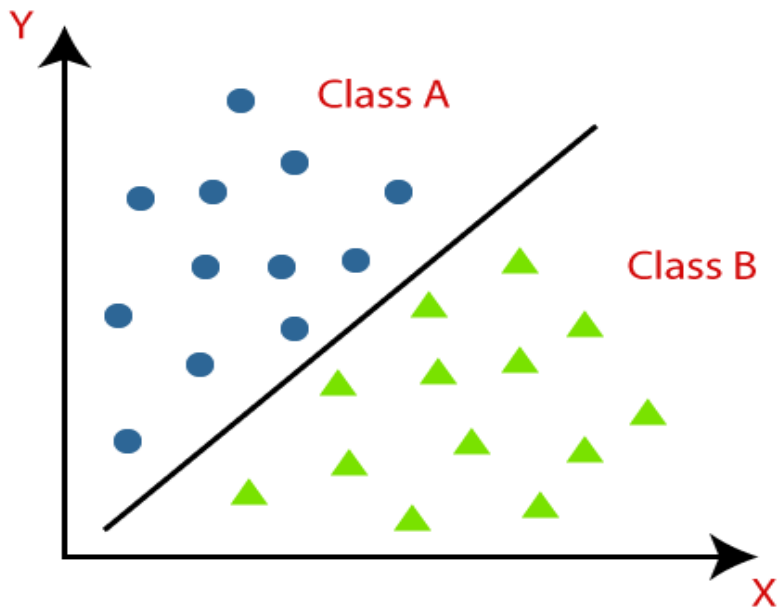

Figure 1: Classification Process. Source: accessed from Javapoint.com on 20.09.2022

2.3.1. Decision Tree

The supervised learning algorithms family includes the decision tree algorithm. The decision tree technique, in contrast to other supervised learning methods, is capable of handling both classification and regression issues. By learning straightforward decision rules derived from previous data, a Decision Tree is used to build a training model that may be used to predict the class or value of the target variable (training data)(Ardandy *et al.*, 2021).

In decision trees, it begin at the tree's root when anticipating a record's class label. It is in contrast to the root attribute's values with that of the attribute on the record. On the basis of comparison, it is followed by the branch corresponding to that value and jump to the next node.(Ghazvini, Abu Bakar and Awwalu, 2014)

Important Terminology related to Decision Trees

- **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
- **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
- **Pruning:** When sub-nodes are removed from the sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
- **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.(Ghazvini, Abu Bakar and Awwalu, 2014)

2.3.2. Random Forest

A random forest is a machine learning method for tackling classification and regression issues. It makes use of ensemble learning, a method for solving complicated issues by combining a number of classifiers. In a random forest algorithm, there are many different decision trees. The random forest algorithm creates a "forest" that is trained via bagging or bootstrap aggregation. The accuracy of machine learning algorithms is increased by bagging, an ensemble meta-algorithm(Bouckaert and Frank, 2004). The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

The decision tree algorithm's shortcomings are eliminated with a random forest. It improves precision and lowers dataset overfitting. Without requiring numerous configurations in packages, it generates predictions (like scikit-learn).(*Introduction to Random Forest in Machine Learning | Engineering Education (EngEd) Program | Section*, 2019)

2.3.3. Support Vector Machine(SVM)

A supervised machine learning approach called the Support Vector Machine (SVM) can be applied to classification or regression problems. However, classification issues are where it is most frequently utilised. When using the SVM algorithm, each data point is represented as a point in n-dimensional space (where n is the number of features you have), with each feature's value being the value of a certain coordinate.(Ayma *et al.*, 2015) Next, the algorithm performs classification by identifying the hyper-plane that effectively distinguishes the two classes (look at the below snapshot).

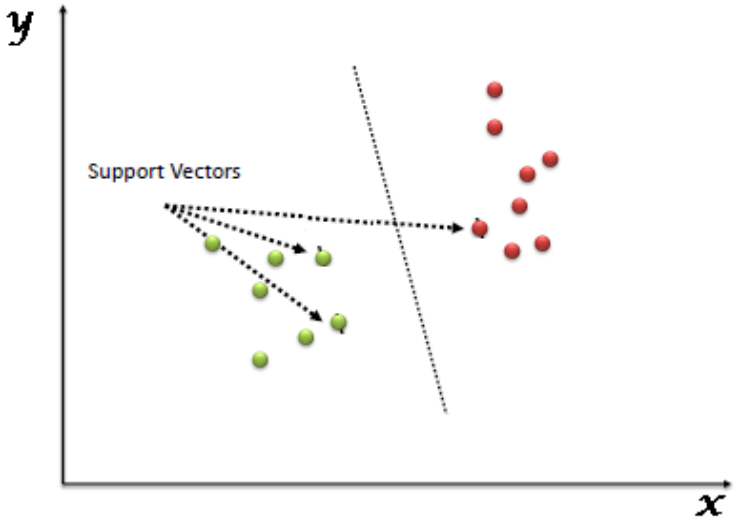

Figure 2: How Support Vector Works. Source: analyticsvidhya.com accessed 20/09/2022

Simply put, support vectors are an individual observation's coordinates. A frontier that best separates the two classes (hyper-plane/line) is the SVM classifier.(*SVM | Support Vector Machine Algorithm in Machine Learning*, 2017)

2.3.4. K-Nearest Neighbours

The k-nearest neighbours algorithm, sometimes referred to as KNN or k-NN, is a supervised learning classifier that employs proximity to produce classifications or predictions about the grouping of a single data point. Although it can be applied to classification or regression issues, it is commonly employed as a classification algorithm because it relies on the idea that is the comparable points can be discovered close to one another.(Yang and Liu, 1999). A class label is chosen for classification problems based on a majority vote, meaning that the label that is most commonly expressed around a particular data point is adopted. Despite the fact that this is officially "plurality voting," literature more frequently refers to "majority vote." The difference between both terms is that "majority voting" informally calls for a majority of more than 50%, which typically only applies when there are only two options. You don't absolutely need 50% of the vote to draw a conclusion about a class when there are many classes, such as four categories; you might assign a class label with a vote of more than 25%.(Beleites and Salzer, 2008).

2.3.5. Bernoulli Naïve Bayesian

The Naive Bayes Classifier is frequently used in machine learning because to its effectiveness and simplicity. By selecting one or more class labels from a pre-defined list of fixed classes, the data is categorised. Naive Bayes employs two event modes: the multinomial event model (Kibriya *et al.*, 2004) for multi-class classification problems and the Bernoulli model for binary classification. Hadoop Map Reduce for text categorization suffers from a processing speed issue that increases latency and makes use of a lot of code. Text classification in the existing Hadoop Map Reduce system uses KMeans. K-Means is a restriction when there are too many data points and when processing speed is a concern due to its complicated time complexity (Pakhira, 2014)

The Naive Bayes Classifier is frequently used in machine learning because to its effectiveness and simplicity. By selecting one or more class labels from a pre-defined list of fixed classes, the data is categorised. Naive Bayes employs two event modes: the multinomial event model (Kibriya *et al.*, 2004)for multi-class classification problems and the Bernoulli model for binary classification. Hadoop Map Reduce for text categorization suffers from a processing speed issue that increases latency and makes use of a lot of code. Text classification in the existing Hadoop Map Reduce system uses KMeans. K-Means is a restriction when there are too many data points and when processing speed is a concern due to its complicated time complexity (Pakhira, 2014).(Venkatesh and Ranjitha, 2019).

2.3.6. Gaussian Naïve Bayesian

Gaussian naive Bayes classification is a case of naive Bayes method with an assumption of having a Gaussian distribution on attribute values given the class label. For example, suppose that ith attribute is continuous and its mean and variance are represented by μc,iandσ2c,i, respectively, given the class label c. Hence, the probability of observing the value xi in ith attribute given the class label c, is computed by Equ.(1) that is also called as normal distribution.

$$p(xi|c)=1\sqrt{2\pi\sigma 2c,i}\exp(-(xi-\mu c,i)22\sigma 2c,i) \quad (1)$$

Even with Gaussian estimation, this method suffers the weakness of conditional independence of attributes. (Jahromi and Taheri, 2018).

2.3.7. Artificial Neural Network

Neural networks or artificial neural networks (ANN) are computer techniques. It aimed to mimic the behaviour of "neurons-based" biological systems. ANNs are computer models that draw inspiration from the central nervous systems of animals. Both pattern recognition and machine learning are possible with it. These were portrayed as networked networks of "neurons" that could compute values from inputs. An oriented graph is a neural network. It is made up of nodes connected by arcs that, using a biological analogy, represent neurons.(Ul Rehman *et al.*, 2018) Dendrites and synapses are corresponding to it. While at each node, each arc had a weight assigned to it. Apply the data the node has received as input and define the activation function along the incoming arcs, with the arc weights taken into account.

A machine learning technique called a neural network is based on the design of a human neuron. There are millions of neurons in the human brain. Electrical and chemical impulses are sent and processed by it. These neurons are linked together by synapses, a unique structure(Baker *et al.*, 2005). Neurons can transmit messages through synapses. Neural networks emerge from enormous numbers of simulated neurons. An information processing method is an artificial neural network. It functions similarly to how the human brain handles information. A huge number of interconnected processing units make up an ANN, which collaborate to process data. They also get useful outcomes from it.(Sharma Sheetal, 2017)

3. METHODOLOGY

3.1. Overview

In order to provide the best analysis of a car evaluation dataset it was used with multiple classification algorithms and compare all of them against each other to see which provides the best accuracy ratio. This is to help a customer/ prospective buyer to have an idea that which kind of car suits the best for them. To the best of my knowledge the analysis of the car to this extent is not done till now and I saw that supervised classification algorithm is performing better in training and testing aspect. In this analysis it is to firstly load the dataset.(Pai and Potdar, 2017) Then it is cleaned the dataset and check the type of dataset whether it is numerical type, categorical type, time series dataset or it is a text dataset. In broader aspect it could be summarised that this dataset is either a numerical dataset or a bivariate dataset. For this dataset it falls under bivariate dataset. Here, I uploaded the dataset and checked for its data value. After this I assigned the column names of the dataset and clarify that these are the column names of the said values. Since the classification algorithms cannot understand bivariate data which includes character in it. So, it is encoded in the different data value which is specific numerical values which makes it

possible for the algorithm to understand the values and give the answer based on it. However, I use it to train the algorithms using the encoded values but only using half of the dataset to do so. The other half of the dataset is then used for testing the algorithms. Which gives either the predicted values or gives the accuracy of the algorithms. (Bouckaert and Frank, 2004)

3.2. Tools and Kit

Colaboratory, sometimes known as "Colab," is a Google Research product. Colab is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser. Technically speaking, Colab is a hosted Jupyter notebook service that offers free access to computer resources, including GPUs, and requires no setup to use. Users can build and run Python code in their web browsers using Google Colaboratory, a freemium product provided by Google Research. The Jupyter open source is actually the foundation of Colab, which enables you to generate and share calculation files without having to download or install anything.(*Google Colab*, 2017).

3.3. Exploratory Data Analysis

Exploratory data analysis is a crucial procedure that entails performing early investigations on data in order to find patterns, identify anomalies, test hypotheses, and validate assumptions with the aid of summary statistics and graphical representations.(Patil Prasad, 2018). Here, it is kept the following column names in x-axis and decision column on y-axis so as to compare the relationship between all the columns and confirm our own hypothesis.

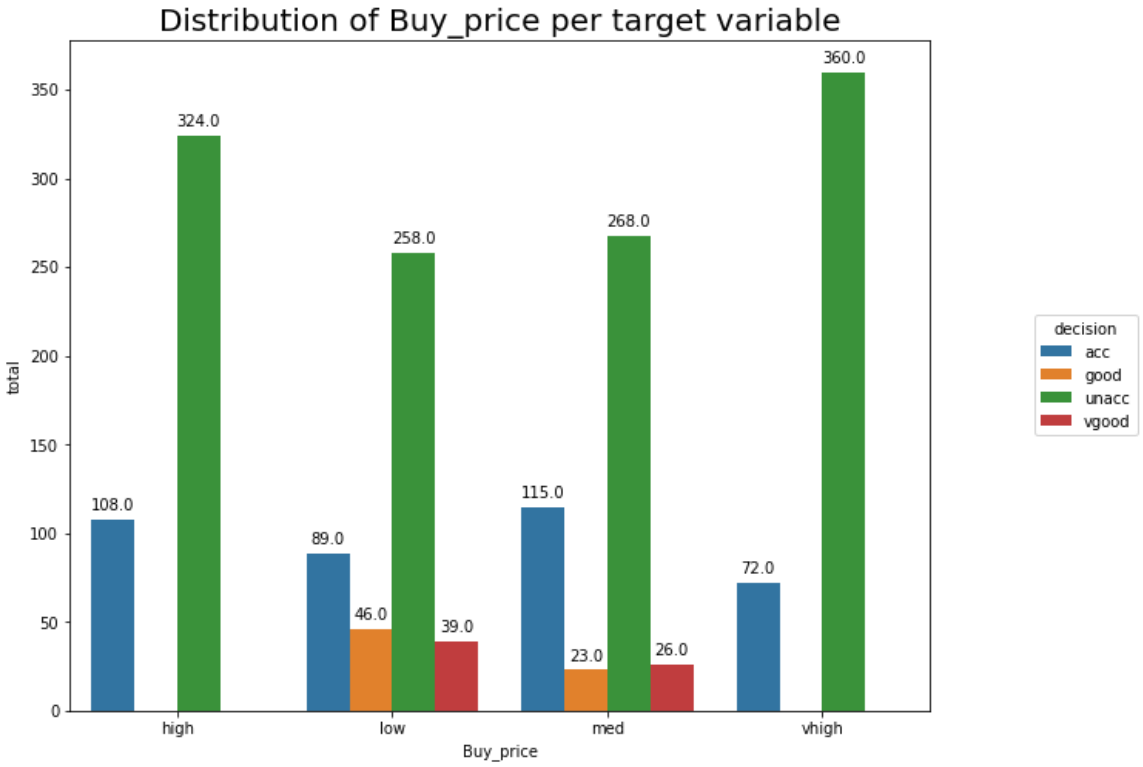

Figure 3: Buying price to decision relation

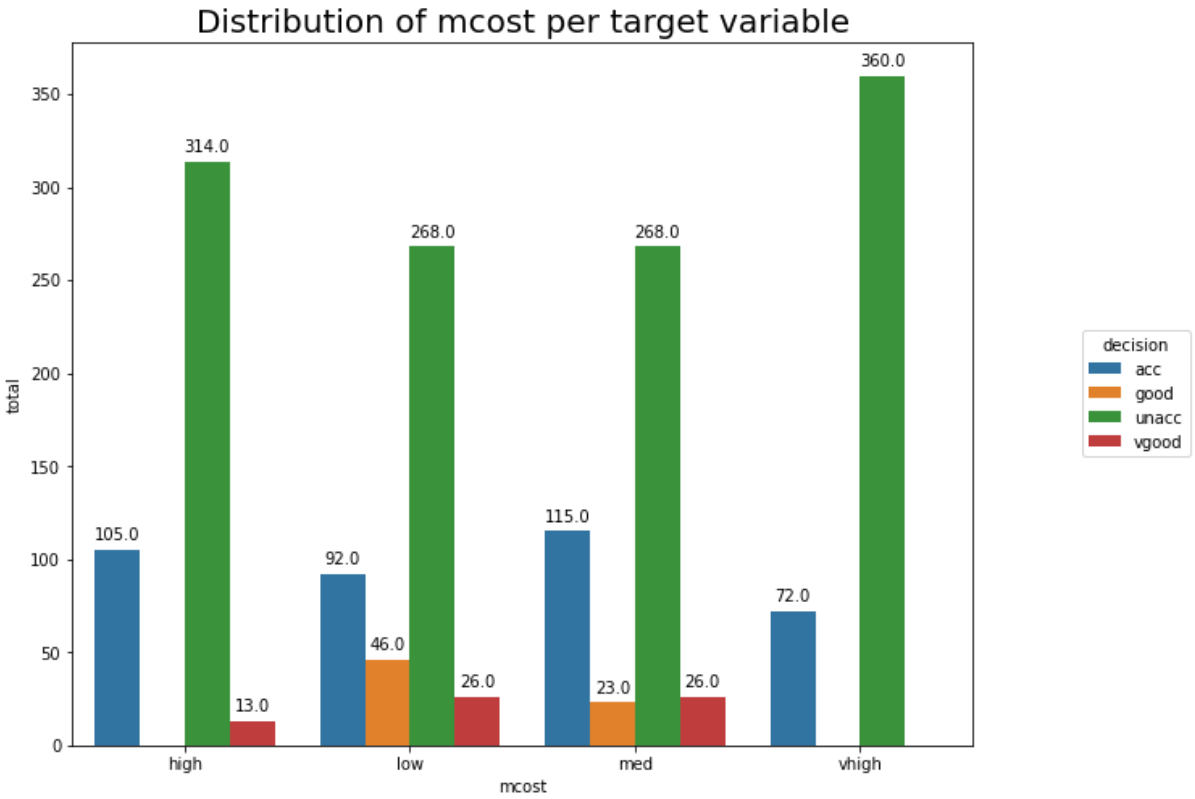

*Figure 4:*Maintenance cost to decision

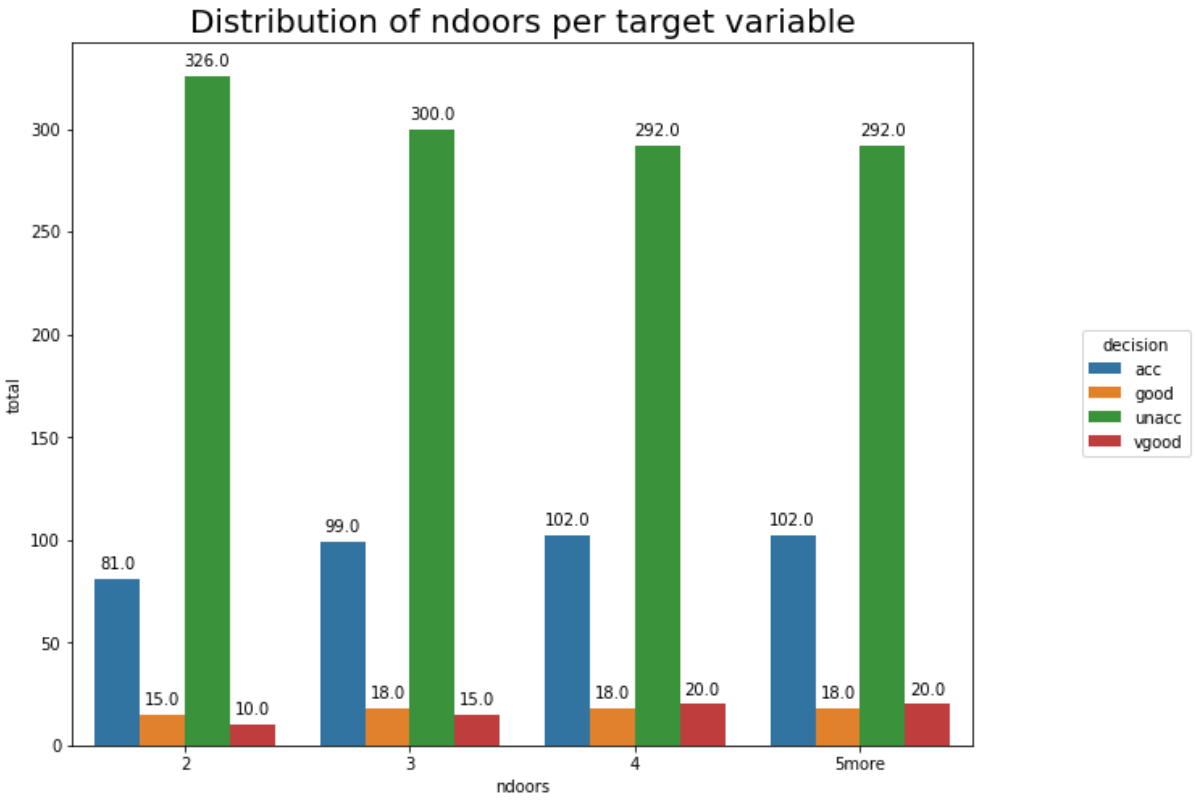

Figure 5: Number of doors to decision relation

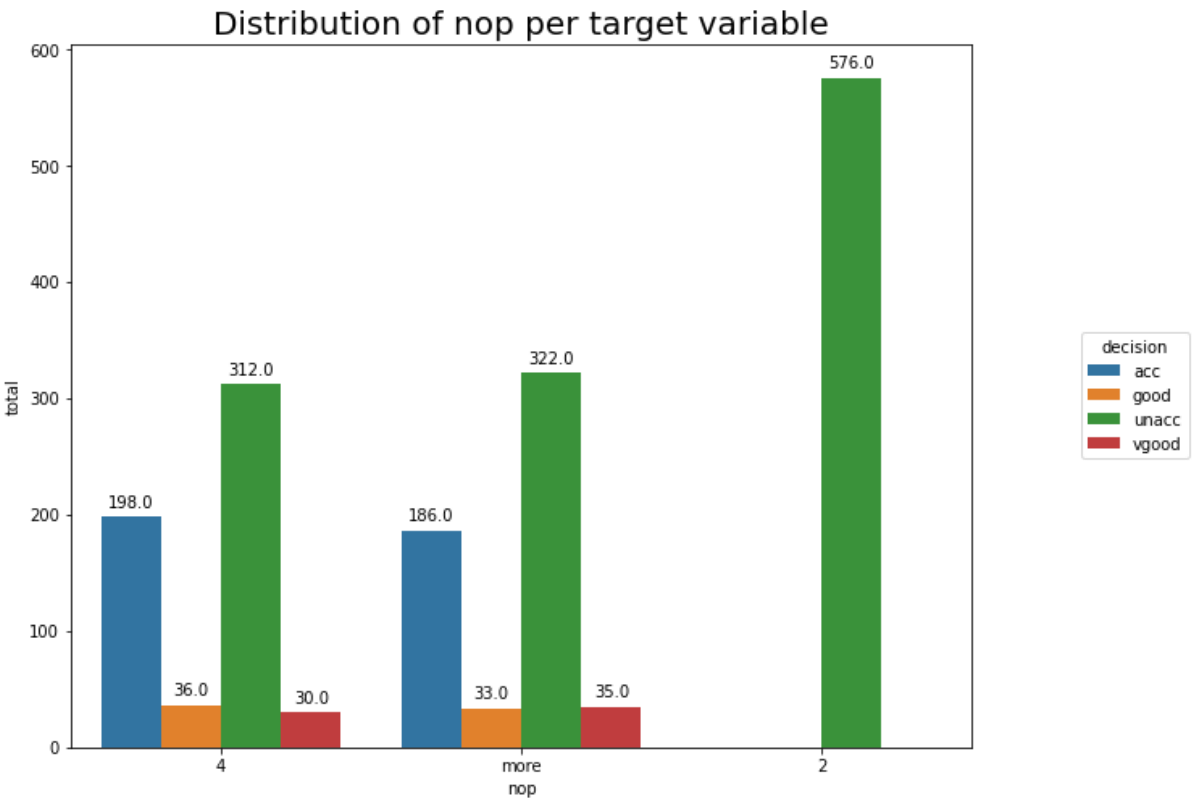

Figure 6: Number of people to decision relation

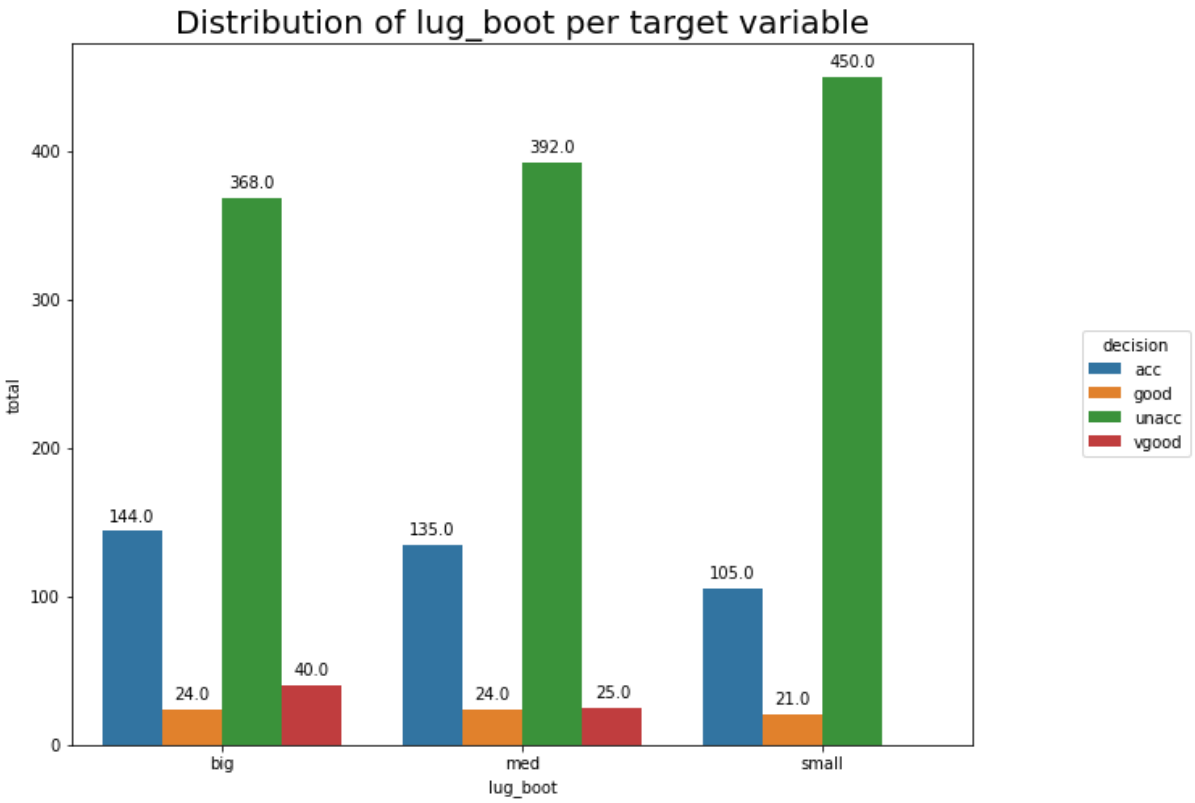

Figure 7: Luggage boot space to decision relation

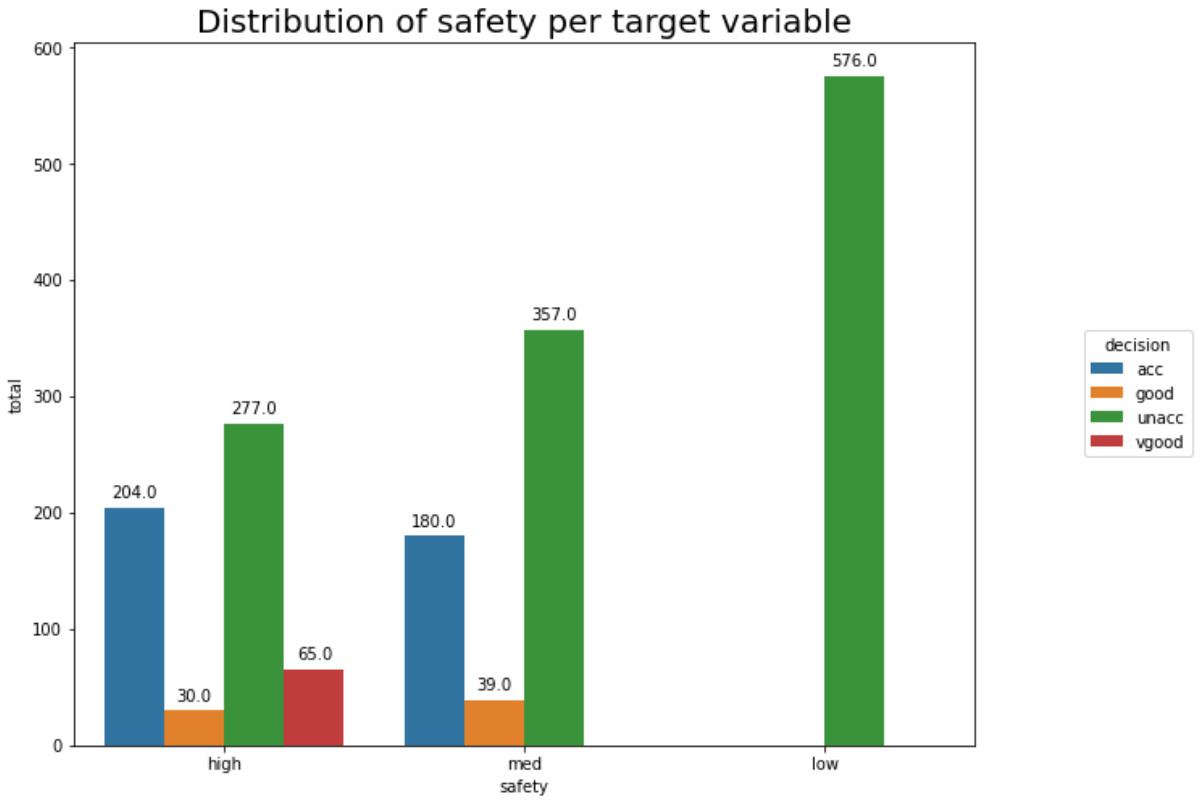

Figure 8: Safety decision relation

3.4. Experiments

3.4.1. Data Cleaning

The practise of correcting or deleting inaccurate, damaged, improperly formatted, duplicate, or incomplete data from a dataset is known as data cleaning. There are numerous ways for data to be duplicated or incorrectly categorised when merging multiple data sources. Even if results and algorithms appear to be correct, they are unreliable if the data is inaccurate. Because the procedures will differ from dataset to dataset, there is no one definitive way to specify the precise phases in the data cleaning process. But it is essential to create a template for your data cleaning procedure so you can be sure you are carrying it out correctly each time.(Company Sales force, 2019)

3.4.2. Data transformation

The process of altering the format, structure, or values of data is known as data transformation. Data can be modified at two points in the data pipeline for initiatives including data analytics. Data transformation serves as the middle phase in an ETL (extract, transform, load) process, which is commonly used by businesses with on-premises data warehouses. Today, the majority of businesses employ cloud-based data warehouses that expand compute and storage resources with latency measured in seconds or minutes. Organizations can load raw data into the data warehouse without preload transformations thanks to the cloud platform's scalability; this is known as the ELT model (extract, load, transform). (A Talend Product Stitch, 2015)

Processes such as data integration, data migration, data warehousing, and data wrangling all may involve data transformation.Data transformation can be aesthetic (such as standardising salutations or street names), destructive (such as eliminating fields and records), or constructive (adding, copying, and replicating data) (renaming, moving, and combining columns in a database). (Yuan, Cong and Thalmann, 2012).

3.4.3. Data set split

When data is divided into two or more subgroups, this is known as data splitting. A two-part split often involves testing or evaluating the data in one part and training the model in the other. Data splitting is a crucial component in data science, especially when building models from data. This method aids in ensuring the accuracy of data model construction and the processes that use data models, such as machine learning. (Gillis, 2019).

3.5. Algorithms: Training and Testing

A method for assessing a machine learning algorithm's performance is the train-test split. It can be applied to issues involving classification or regression as well as any supervised learning algorithm. The process entails splitting the dataset into two subsets. The training dataset is the first subset, which is used to fit the model. The model is not trained using the second subset; rather, it is given the input element of the dataset, and its predictions are then made and contrasted with the expected values.(Ramanathan and Sharma, 2017) The test dataset is the second dataset in question.

- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model. (Pai and Potdar, 2017). It is anticipated that applying the model in this way. In other words, to fit it to the data that is currently available with inputs and outputs that are known, then to predict on new examples in the future where do not have the anticipated output or target values.(Brownlee, 2020)

Coding sample for training and testing all the algorithm

```
from sklearn.model_selection import train_test_split,GridSearchCV
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_sta
   te=1)

from sklearn import tree,svm,naive_bayes,neighbors,ensemble
from sklearn.metrics import confusion_matrix,accuracy_score,classification_re
   port

#decision tree

from sklearn.model_selection import GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 6, 9, 12],
    'max_features': [2, 3],
    'min_samples_leaf': [2, 3, 4, 5]
}
dt_gs = GridSearchCV(tree.DecisionTreeClassifier(), param_grid = param_grid)
dt_gs.fit(X_train, y_train)
print("####### DECISION TREE #######")
print(dt_gs.best_params_)

#output
####### DECISION TREE #######
#{'criterion': 'entropy', 'max_depth': 12, 'max_features': 3, 'min_samples_le
   af': 2}

# random forest

# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
rf_gs = GridSearchCV(ensemble.RandomForestClassifier(), param_grid = param_gr
   id,
                          cv = 3, n_jobs = -1, verbose = 2)
rf_gs.fit(X_train, y_train)
print(" ####### RANDOM FOREST #######")
```

```
print(rf_gs.best_params_)

# output
 ####### RANDOM FOREST #######
# {'bootstrap': True, 'max_depth': 80, 'max_features': 3, 'min_samples_leaf':
    3, 'min_samples_split': 12, 'n_estimators': 100}

# support vector machines

param_grid2 = {'C': [0.1, 1, 10, 100, 1000],
               'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
               'kernel': ['rbf','linear'],
               'decision_function_shape':['ovo','ovr']}
svc_gs = GridSearchCV(svm.SVC(), param_grid2, refit = True, verbose = 3)
svc_gs.fit(X_train, y_train)
print(" ####### SVM #######")
print(svc_gs.best_params_)

#output
####### SVM #######
#{'C': 100, 'decision_function_shape': 'ovo', 'gamma': 0.1, 'kernel': 'rbf'}



# knn
k_range = list(range(1, 31))
print(k_range)
param_grid = dict(n_neighbors=k_range)
print(param_grid)
knn_gs = GridSearchCV(estimator=neighbors.KNeighborsClassifier(),param_grid=p
   aram_grid,cv=5,return_train_score=True) # Turn on cv train scores
knn_gs.fit(X_train, y_train)
print("####### KNN ######")
print(knn_gs.best_params_)

#output
# 6 neighbours

# Artificial Neural Network
# Split Data to Train and Test
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size=0.2)
```

3.6. Comparison among different algorithms

3.6.1. Decision tree v/s Random Forest

Table 1: Decision Tree VS Random Forest Source(K., 2020)

Decision Tree	Random Forest®
A decision tree is a tree-like model of decisions along with possible outcomes in a diagram.	A classification algorithm consisting of many decision trees combined to get a more accurate result as compared to a single tree.
There is always a scope for overfitting, caused due to the presence of variance.	Random forest algorithm avoids and prevents overfitting by using multiple trees.
The results are not accurate.	This gives accurate and precise results.
Decision trees require low computation, thus reducing time to implement and carrying low accuracy.	This consumes more computation. The process of generation and analyzing is time-consuming.
It is easy to visualize. The only task is to fit the decision tree model.	This has complex visualization as it determines the pattern behind the data.

Code Sample for decision tree and random forest:

```
#DECISION TREE
y_pred = dt_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in decision tree is : {}".format(cas)
   )
print("Confusion matrix for decision tree is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'Decision Tree GS'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = dt_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in decision tree is : {}".format(cas)
   )
print("Confusion matrix for decision tree is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for decision tree is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
```

```
#RANDOM FOREST
y_pred = rf_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Random Forest is : {}".format(cas)
   )
print("Confusion matrix for Random Forest is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'Random Forest'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = rf_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Random Forest is : {}".format(cas)
   )
print("Confusion matrix for Random Forest is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for Random Forest is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
```

3.6.2. SVM v/s KNN

Support Vector Machine (SVM) is a Supervised Machine Learning algorithm that is used for regression and/or classification. Although it is sometimes very helpful for regression, classification is where it is most often used. In essence, SVM identifies a hyper-plane that establishes a distinction between the various types of data. This hyper-plane is just a line in two-dimensional space. Each dataset item is plotted in an N-dimensional space using SVM, where N is the total number of features and attributes in the dataset.(Yang and Liu, 1999). The best hyperplane should then be found to divide the data. You must have realised by now that SVM can only perform binary classification naturally (i.e., choose between two classes). For multi-class problems, there are numerous approaches to apply. The two results of each classifier will be: The data point belongs to that class, or the data point does not belong to that class.(Ayma *et al.*, 2015)

In contrast, for non-linearly separable data, we employ kernelized SVM. Let's say we have some data in one dimension that is non-linearly separable. This data can be transformed into two dimensions, where it will become linearly separable. Each 1-D data point is mapped to a

corresponding 2-D ordered pair to do this. Therefore, we can simply translate the data to a higher dimension and then make it linearly separable for any non-linearly separable data in any dimension. This shift is profound and wide-ranging. A kernel is nothing more than a comparison of data points' similarity. Given two data points in the original feature space, the kernel function in a kernelized SVM reveals the degree of similarity between the points in the newly converted feature space. There are various kernel functions available, but two are very popular:

Radial Basis Function Kernel (RBF): The similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below. RBF is the default kernel used in SVM. Equation (2)

$$\mathbf{K(x,x') = exp(-\gamma||x-x'||^2)}\mathbf{\ldots\ldots\ldots\ldots\ldots\ldots\ldots eq}(2)$$

Polynomial Kernel: The Polynomial kernel takes an additional parameter, 'degree' that controls the model's complexity and computational cost of the transformation.(*What is the k-nearest neighbors algorithm? | IBM*, 2019)

Code Sample for SVM and KNN

```
#Support Vector Machine
y_pred = svc_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in SVM is : {}".format(cas))
print("Confusion matrix for SVM is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'SVM'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = svc_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in SVM is : {}".format(cas))
print("Confusion matrix for SVM is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for SVM is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1

#K-Nearest Neighbours
y_pred = knn_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in KNN is : {}".format(cas))
```

```
print("Confusion matrix for KNN is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'KNN'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = knn_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in KNN is : {}".format(cas))
print("Confusion matrix for KNN is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for KNN is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
```

3.6.3. Bernoulli V/s Gaussian V/s ANN

The naive Bayesian classifier is a very appealing classifier that has shown to be successful in a variety of real-world settings, such as text classification, medical diagnosis, and system performance management. The Naive Bayes model was used by (Ghazvini, Abu Bakar and Awwalu, 2014)to predict the author of Arabic texts. The Nave Bayes model has also been utilised for automatic classification of web pages from a huge data network, defect diagnostics in steam turbines, and sales forecasting(Ul Rehman *et al.*, 2018). The naive Bayesian classifier is a very appealing classifier that has shown to be successful in a variety of real-world settings, such as text classification, medical diagnosis, and system performance management. The Naive Bayes model was used by to predict the author of Arabic texts. The Nave Bayes model has also been utilised for automatic classification of web pages from a huge data network, defect diagnostics in steam turbines, and sales forecasting. (Simfukwep, Kundap and Chembep, 2015).

Code sample for Bernoulli Naïve Bayes, Gaussian Naïve Bayes and Artificial Neural Network

```
#BERNOULLI NB
cl1.fit(X_train,y_train)
y_pred = cl1.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Bernoulli NB is : {}".format(cas))
print("Confusion matrix for Bernoulli NB is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'BernoulliNB'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = cl1.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Bernoulli NB is : {}".format(cas))
print("Confusion matrix for Bernoulli NB is : ")
```

```
print(confusion_matrix(y_pred,y_test))
print("Classification report for Bernoulli NB is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1

#GAUSSIAN NB
cl2.fit(X_train,y_train)
y_pred = cl2.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Gaussian NB is : {}".format(cas))
print("Confusion matrix for Gaussian NB is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'GaussianNB'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100

y_pred = cl2.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Gaussian NB is : {}".format(cas))
print("Confusion matrix for Gaussian NB is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for Bernoulli NB is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1


# Artificial Neural Network

# create model
model = Sequential()
model.add(Dense(25, input_dim=6, kernel_initializer='uniform', activation='re
   lu'))
model.add(Dense(30, kernel_initializer='uniform', activation='relu', kernel_c
   onstraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(10, kernel_initializer='uniform', activation='relu'))
model.add(Dense(1, kernel_initializer='uniform', activation='relu'))

# Compile model
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

# Fit the model
model.fit(X_Train, Y_Train, epochs=600, batch_size=10)
```

```
# Evaluate the model
scores = model.evaluate(X_Test, Y_Test)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

4. Results and Discussion

1.1.DATA CLEANING

Data cleansing is used in classification challenges to get better outcomes. The primary goal of data cleaning is to eliminate all inconsistencies from training data. (Beleites and Salzer, 2008). As unclean data affect the accuracy of the results acquired, this gives us adequate justification to employ reliable data while creating an effective classification model. The dataset used in this study has also been cleansed to guarantee its suitability for model construction.(Pai and Potdar, 2017). Nominal qualities have been changed to numerical attributes. This conversion is necessary in order to enable data normalisation. The conversion is displayed in Table 2

Table 2: Nominal to numeric conversion

Attribute	**Nominal**	**New Numeric Value**
Buying	Vhigh	1
	High	2
	Med	2
	Low	4
Maintenance	Vhigh	1
	High	2
	Med	3
	Low	4
Number of doors	2	1
	4	2
	5 more	3
Number of People	2	1
	4	2
	more	3
Luggage boot space	Small	1
	Med	2
	Big	3
Safety	Low	1
	Med	2
	High	3
Classes/ Decision	Unacc	1

	Acc	2
	Good	3
	Vgood	4

1.2.DATA TRANSFORMATION

Raw input data which was pre-processed after the dataset was selected to avoid having a detrimental impact on the outcomes. It is absolutely essential to the effectiveness of the neural network and supervised algorithms.(A Talend Product Stitch, 2015) The two fundamental pre-processing methods are normalisation and data transformation. While normalisation tends to distribute data uniformly by scaling it over an appropriate range, transformation manipulates raw data inputs to create a single input. This can aid the learning process of the network by improving its capacity to comprehend the relationship between inputs and outputs produced. Instead of using z-score methods, min-max normalisation is being used for this particular investigation. Generally speaking, min-max normalisation will always produce results in the 0 to 1 range.(Ramanathan and Sharma, 2017).

1.3.DATA SPLIT SET

In order to use one half of the pre-processed dataset as training data and the other as testing or validation network, the dataset is divided into two shares of variable sizes. The approach used for data splitting can have a significant impact on how well a model performs. Incorrect data splitting can lead to inaccurate and highly variable performance. Data from training sets are used to train the classification algorithm. By contrasting the attributes of the dataset with the class or label, the training model is created. The model is evaluated on test data, which is the other half of the split dataset, after training. Four splits are tested in this research study. In the data set split, a 10-fold cross validation is also checked. The fitting operation will be carried out ten times, with each fit consisting of a training set at 90% and a testing set at 10%. This is referred to as 10-folds.(Bouckaert and Frank, 2004).

1.3.1. DECISION TREE VS RANDOM FOREST

Using Decision tree algorithm we get the train accuracy of 93.8% and testing accuracy of 86.3%. However, when it was applied Random forest algorithm we get the trained accuracy of 98.9% ~ 99% where as for the test accuracy we get 95.9% ~ 96%. By utilising the random forest algorithm which avoids and prevents overfitting. The outcomes are not reliable. Results are exact and accurate. Decision trees are quick to implement and have low accuracy because they need little calculation.

1.3.2. SVM VS KNN

Support Vector Machine training accuracy IT WAS ACHIEVED achieve 99.9% where we can see that this is the highest of all the algorithms we have applied so far but for the testing accuracy it only performed 96.9% ~ 97%. On the other hand KNN or K- nearest neighbour algorithm performed only 96.1% on training dataset however on the testing of the algorithm we can see that it was only 91.9% ~ 92% only. While both algorithms yield positive results regarding the accuracy

in which they classify the dataset, the SVM provides significantly better classification accuracy and classification speed than the KNN.

1.3.3. BERNOULLI V/S GAUSSIAN V/S ANN

we applied Bernoulli naïve Bayes algorithm which predicted trained accuracy of 69.6% which is the lowest of all the algorithms so far. This Algorithm when tested performed gave a better result of 70.9% ~ 71%. Also the Gaussian Naïve Bayes algorithm performed almost equivalent to Bernoulli Naïve Bayes algorithm which predicted the training accuracy of 70.3% but the testing of this algorithm gave 70.9% ~ 71% accuracy. The last algorithm that was applied to this dataset was Artificial Algorithm. Which predicted the accuracy score of 71.1% when tested on the dataset.

1.4. Key Findings

This project shows the comparison of seven different classification algorithm which depicts the relation between the condition of a car to purchase it. There are relations in the dataset which depicts that higher the accuracy of the algorithm predicts there is better chance of getting the prediction correct which is buyer purchasing on the prediction by the algorithm. Hence, according to the applied classification algorithm Support Vector Machine predicts with the accuracy of 97%. Which is the highest of all the algorithms the way SVM works is by finding the best/optimal decision boundary that will help us separate the classes most accurately. First off, THE dataset's somewhat skewed classes could be used by the technique because only a small number of vectors (data points) are needed to determine the categorization border. Therefore, the algorithm only needed a few examples of each class to generate a reasonably accurate decision boundary. Secondly, if we think about our issue, we need to assess the car's acceptability, and the characteristics that are offered suggest that the final choice may be influenced by minor adjustments to a few elements. Since support vectors are the most helpful data points since they are the ones most likely to be erroneously classified, let's imagine that the car's maintenance cost is high but the rest of the attributes are reasonable. This high maintenance cost may have a big impact on the final selection.

With an accuracy of 97% , It was found that the Support Vector Machine algorithm outperforms all the algorithm. The performance of Random Forest, KNN, Decision Tree, SVM, Gaussian NB, Bernoulli NB and Artificial Neural Network in classifying automotive performance quality using a dataset taken from the UCI repository is investigated in this research. Different attributes are used to define performance in these datasets. In comparison to all the method, the suggested work shows that SVM performs the best performs better.

The datasets considered in this report; the Naïve Bayes and Artificial Neural Network algorithm can detect the performance of cars with moderate accuracy. This study observed that, in order to get the higher accuracy the dataset needs to be much higher in data values. The proposed and existing works differ by number of attributes, samples and amount of data and implemented decision tree algorithm, KNN and random forest with an accuracy of 86.3%,91.9% ~ 92% and 95.9% ~ 96% respectively proposed a support vector machine classifier for classification purposes and achieved an accuracy of 96.9% ~ 97%. For data mining classification on automobile data sets, artificial neural networks (ANN) and naive Bayesian classifiers (NB) were used. The accuracy of these two algorithms is 70.9% ~ 71% and 71.1% respectively . The performance of the algorithms varies depending on the amount of data, balanced or unbalanced data, as

shown in the report above. In my work, I used a total of 1728 samples of values which are accurate, inaccurate, good and very good quality of data. It is observed that, all the algorithms are found to be able to handle the imbalance data. Support vector machine algorithm achieves a better accuracy of 97%.

2. Conclusion

Accidents are reduced by accurate identification of car quality evaluation. When the performance of all algorithms are compared, the suggested Support Vector Machine approach outperforms the all the algorithm applied. The suggested algorithm solves the difficulties in this datasets by enhancing accuracy by 97% for Support Vector Machine algorithm and outperforms all the applied algorithms for this report. Future study can utilise more sophisticated techniques to increase accuracy and address other issues, such as choosing the type of emotion, assembling the testing dataset, and looking at more auto evolution as there are a large number of flexible cars on the market. We can conduct the same research for other things outside only compact brands.

References

1. *6 Real-World Examples of Machine Learning - Salesforce Blog - Salesforce EMEA Blog* (no date). Available at: https://www.salesforce.com/eu/blog/2020/06/real-world-examples-of-machine-learning.html (Accessed: 26 September 2022).
2. A Talend Product Stitch (2015) *What is data transformation: definition, benefits, and uses | Stitch*. Available at: https://www.stitchdata.com/resources/data-transformation/ (Accessed: 26 September 2022).
3. Ardandy, F. A. *et al.* (2021) 'Perbandingan Algoritma Naive Bayes dan Linear Discriminant Analysis dengan Dataset Car Evaluation', *Jurnal Rekayasa Elektro Sriwijaya*, 3(1), pp. 213–217. doi: 10.36706/JRES.V3I1.45.
4. Ayma, V. A. *et al.* (2015) 'Classification algorithms for big data analysis, a map reduce approach', *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 40(3W2), pp. 17–21. doi: 10.5194/ISPRSARCHIVES-XL-3-W2-17-2015.
5. Baker, G. *et al.* (2005) 'Cochlea Modelling: Clinical Challenges and Tubular Extraction BT - AI 2004: Advances in Artificial Intelligence', *Australasian Joint Conference on Artificial Intelligence*. Edited by G. I. Webb and X. Yu, pp. 74–85.
6. Beleites, C. and Salzer, R. (2008) 'Assessing and improving the stability of chemometric models in small sample size situations', *Analytical and Bioanalytical Chemistry*, 390(5), pp. 1261–1271. doi: 10.1007/S00216-007-1818-6.
7. Bouckaert, R. R. and Frank, E. (2004) 'Evaluating the replicability of significance tests for comparing learning algorithms', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3056, pp. 3–12. doi: 10.1007/978-3-540-24775-3_3.
8. Brownlee, J. (2020) *Train-Test Split for Evaluating Machine Learning Algorithms*. Available at: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/ (Accessed: 26 September 2022).

9. *Car Evaluation Analysis Using Decision Tree Classifier | by Mohammad Masum, PhD | Towards Data Science* (no date) *towards data science*. Available at: https://towardsdatascience.com/car-evaluation-analysis-using-decision-tree-classifier-61a8ff12bf6f (Accessed: 26 September 2022).
10. *Classification Algorithm in Machine Learning - Javatpoint* (no date). Available at: https://www.javatpoint.com/classification-algorithm-in-machine-learning (Accessed: 26 September 2022).
11. Company Sales force (no date) *Data Cleaning: Definition, Benefits, And How-To | Tableau*. Available at: https://www.tableau.com/learn/articles/what-is-data-cleaning (Accessed: 26 September 2022).
12. Edgar, J. (no date) 'Optimising car performance modifications : simple methods of measuring engine, suspension, brakes. and aerodynamic performance gains', p. 72.
13. *Evaluating a Car's Condition with Machine Learning - Hypi* (no date). Available at: https://hypi.io/2019/10/15/evaluating-a-cars-condition-with-machine-learning/ (Accessed: 26 September 2022).
14. Ghazvini, A., Abu Bakar, A. and Awwalu, J. (2014) 'Performance Comparison of Data Mining Algorithms: A Case Study on Car Evaluation Dataset', *International Journal of Computer Trends and Technology*, 13(2). doi: 10.14445/22312803/IJCTT-V13P117.
15. Gillis, A. S. (no date) *What is data splitting and why is it important?* Available at: https://www.techtarget.com/searchenterpriseai/definition/data-splitting (Accessed: 26 September 2022).
16. *Google Colab* (no date). Available at: https://research.google.com/colaboratory/faq.html (Accessed: 26 September 2022).
17. Guo, D. *et al.* (2021) 'Forecast of passenger car market structure and environmental impact analysis in China', *Science of The Total Environment*, 772, p. 144950. doi: 10.1016/J.SCITOTENV.2021.144950.
18. *Introduction to Random Forest in Machine Learning | Engineering Education (EngEd) Program | Section* (no date). Available at:

https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/ (Accessed: 26 September 2022).
19. Jahromi, A. H. and Taheri, M. (2018) ‘A non-parametric mixture of Gaussian naive Bayes classifiers based on local independent features’, *19th CSI International Symposium on Artificial Intelligence and Signal Processing, AISP 2017*, 2018-January, pp. 209–212. doi: 10.1109/AISP.2017.8324083.
20. Jain, P. and Kr Vishwakarma, S. (2017) ‘A Case Study on Car Evaluation and Prediction: Comparative Analysis using Data Mining Models’, *International Journal of Computer Applications*, 172(9), pp. 975–8887.
21. K., V. (2020) ‘A Random Forest-based Classification Method for Prediction of Car Price’, *International Journal of Psychosocial Rehabilitation*, 24(3), pp. 2639–2648. doi: 10.37200/IJPR/V24I3/PR2020298.
22. Kibriya, A. M. *et al.* (2004) ‘Multinomial naive bayes for text categorization revisited’, *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, 3339, pp. 488–499. doi: 10.1007/978-3-540-30549-1_43/COVER.
23. Pai, C. and Potdar, K. (2017) ‘A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers’, *Article in International Journal of Computer Applications*, 175(4), pp. 975–8887. doi: 10.5120/ijca2017915495.
24. Pakhira, M. K. (2014) ‘A linear time-complexity k-Means algorithm using cluster shifting’, *Proceedings - 2014 6th International Conference on Computational Intelligence and Communication Networks, CICN 2014*, pp. 1047–1051. doi: 10.1109/CICN.2014.220.
25. Patil Prasad (2018) *What is Exploratory Data Analysis? | by Prasad Patil | Towards Data Science*. Available at: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15 (Accessed: 26 September 2022).
26. Ramanathan, T. T. and Sharma, D. (2017) ‘Multiple Classification Using SVM Based Multi Knowledge Based System’, *Procedia Computer Science*, 115, pp. 307–311. doi: 10.1016/J.PROCS.2017.09.139.

27. Rehman, Z. U. *et al.* (2018) 'Performance evaluation of MLPNN and NB: A Comparative Study on Car Evaluation Dataset', *IJCSNS International Journal of Computer Science and Network Security*, 18(9), p. 144.
28. Sharma Sheetal (2017) *Artificial Neural Network (ANN) in Machine Learning - DataScienceCentral.com*. Available at: https://www.datasciencecentral.com/artificial-neural-network-ann-in-machine-learning/ (Accessed: 26 September 2022).
29. Simfukwep, M., Kundap, D. and Chembep, C. (2015) 'Comparing Naive Bayes Method and Artificial Neural Network for Semen Quality Categorization', *IJISET-International Journal of Innovative Science, Engineering & Technology*, 2(7). Available at: www.ijiset.com (Accessed: 26 September 2022).
30. Stevanovic, D., Vlajic, N. and An, A. (2013) 'Detection of malicious and non-malicious website visitors using unsupervised neural network learning', *Applied Soft Computing Journal*, 13(1), pp. 698–708. doi: 10.1016/J.ASOC.2012.08.028.
31. *SVM | Support Vector Machine Algorithm in Machine Learning* (no date). Available at: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ (Accessed: 26 September 2022).
32. Ul Rehman, Z. *et al.* (2018) *Performance evaluation of MLPNN and NB: A Comparative Study on Car Evaluation Dataset*, *IJCSNS International Journal of Computer Science and Network Security*. Available at: https://www.researchgate.net/publication/332465875.
33. Venkatesh and Ranjitha, K. V. (2019) 'Classification and Optimization Scheme for Text Data using Machine Learning Naïve Bayes Classifier', *2018 IEEE World Symposium on Communication Engineering, WSCE 2018*, pp. 33–36. doi: 10.1109/WSCE.2018.8690536.
34. *What is the k-nearest neighbors algorithm? | IBM* (2019). Available at: https://www.ibm.com/uk-en/topics/knn (Accessed: 26 September 2022).
35. Widyananda, W. *et al.* (2022) 'DATASET MISSING VALUE HANDLING AND CLASSIFICATION USING DECISION TREE

C5.0 AND K-NN IMPUTATION: STUDY CASE CAR EVALUATION DATASET', *Journal of Theoretical and Applied Information Technology*, 30(12). Available at: www.jatit.org (Accessed: 17 September 2022).

36. Yang, Y. and Liu, X. (1999) 'A re-examination of text categorization methods', *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 1999*, pp. 42–49. doi: 10.1145/312624.312647.
37. Yuan, Q., Cong, G. and Thalmann, N. M. (2012) 'Enhancing Naive Bayes with various smoothing methods for short text classification', *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, pp. 645–646. doi: 10.1145/2187980.2188169.

Appendix

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
 list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) tha
t gets preserved as output when you create a version using "Save & Run All
"
# You can also write temporary files to /kaggle/temp/, but they won't be s
aved outside of the current session

# importing datasets
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
# Reading Data Set
from google.colab import drive
drive.mount("/content/gdrive")


# Importing the dataset (Link - https://drive.google.com/file/d/1GBP8p2PML
yo4KK_9CjnTF0g0pDiq85_m/view?usp=sharing)
df = pd.read_csv('/content/gdrive/MyDrive/Dissertation dataset/car_evaluat
ion (version 1).csv', header = 0)
df.describe()
columns = ["Buy_price","mcost","ndoors","nop","lug_boot","safety","decisio
n"]
df.columns = columns
df.head()
df.head()

X_list=["Buy_price","mcost","ndoors","nop","lug_boot","safety"]

X=df[X_list]
y=df.decision
y.value_counts()
```

```
df.shape #tells the number of rows and columns of a given DataFrame.
X.Buy_price.unique()
X.mcost.unique()
X.ndoors.unique()
X.nop.unique()
X.lug_boot.unique()
X.safety.unique()
y.unique()
#defining function for bar graph
def grp_brplt(col1):

    df1 = df.groupby(['decision',col1]).size().to_frame('total').reset_ind
ex()

    plt.figure(figsize=(10,8))
    ax=plt.subplot()
    ax = sns.barplot(data=df1, x=df1[col1], y=df1["total"], hue=df1["decis
ion"])

    for p in ax.patches:
        ax.annotate(format(p.get_height(), '.1f'),
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha = 'center', va = 'center',
                       xytext = (0, 9),
                       textcoords = 'offset points')

    ax.set_title('Distribution of ' +col1+ ' per target variable', fontsiz
e=20)
    ax.legend(loc='center right', bbox_to_anchor=(1.25, 0.5), ncol=1, titl
e='decision')
    return ax
grp_brplt("Buy_price")#buying price
grp_brplt("mcost")#maintenance cost
grp_brplt("ndoors")#number of doors
grp_brplt("nop")#number of people
grp_brplt("lug_boot")#luggage boot space
grp_brplt("safety")#car safety
!pip install category_encoders
# importing necessary package for encoding our categorial features
import category_encoders as ce

encoder_X = ce.OrdinalEncoder(cols=["Buy_price","mcost","ndoors","nop","lu
g_boot","safety"])
X= encoder_X.fit_transform(X)
```

```
encoder_Y = ce.OrdinalEncoder()
y=np.ravel(encoder_Y.fit_transform(y))
from sklearn.model_selection import train_test_split,GridSearchCV
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_
state=1)
from sklearn import tree,svm,naive_bayes,neighbors,ensemble
from sklearn.metrics import confusion_matrix,accuracy_score,classification
_report
cl1 = naive_bayes.BernoulliNB()
cl2 = naive_bayes.GaussianNB()
#decision tree

from sklearn.model_selection import GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 6, 9, 12],
    'max_features': [2, 3],
    'min_samples_leaf': [2, 3, 4, 5]
}
dt_gs = GridSearchCV(tree.DecisionTreeClassifier(), param_grid = param_gri
d)
dt_gs.fit(X_train, y_train)
print("####### DECISION TREE #######")
print(dt_gs.best_params_)

#output
####### DECISION TREE #######
#{'criterion': 'entropy', 'max_depth': 12, 'max_features': 3, 'min_samples
_leaf': 2}

# random forest

# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12],
    'n_estimators': [100, 200, 300, 1000]
}
rf_gs = GridSearchCV(ensemble.RandomForestClassifier(), param_grid = param
_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
rf_gs.fit(X_train, y_train)
```

```
print(" ####### RANDOM FOREST #######")
print(rf_gs.best_params_)

# output
 ####### RANDOM FOREST #######
# {'bootstrap': True, 'max_depth': 80, 'max_features': 3, 'min_samples_lea
f': 3, 'min_samples_split': 12, 'n_estimators': 100}

# support vector machines

param_grid2 = {'C': [0.1, 1, 10, 100, 1000],
               'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
               'kernel': ['rbf','linear'],
               'decision_function_shape':['ovo','ovr']}
svc_gs = GridSearchCV(svm.SVC(), param_grid2, refit = True, verbose = 3)
svc_gs.fit(X_train, y_train)
print(" ####### SVM #######")
print(svc_gs.best_params_)

#output
####### SVM #######
#{'C': 100, 'decision_function_shape': 'ovo', 'gamma': 0.1, 'kernel': 'rbf
'}

# knn

k_range = list(range(1, 31))
print(k_range)
param_grid = dict(n_neighbors=k_range)
print(param_grid)
knn_gs = GridSearchCV(estimator=neighbors.KNeighborsClassifier(),param_gri
d=param_grid,cv=5,return_train_score=True) # Turn on cv train scores
knn_gs.fit(X_train, y_train)
print("####### KNN ######")
print(knn_gs.best_params_)

#output
# 6 neighbours
import pandas as pd
row_num = 0
compiler_compare = pd.DataFrame()
#DECISION TREE
y_pred = dt_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
```

```
print("Accuracy score for training data in decision tree is : {}".format(c
as))
print("Confusion matrix for decision tree is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'Decision Tree GS'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = dt_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in decision tree is : {}".format(c
as))
print("Confusion matrix for decision tree is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for decision tree is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#RANDOM FOREST
y_pred = rf_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Random Forest is : {}".format(c
as))
print("Confusion matrix for Random Forest is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'Random Forest'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = rf_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Random Forest is : {}".format(c
as))
print("Confusion matrix for Random Forest is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for Random Forest is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#Support Vector Machine
y_pred = svc_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in SVM is : {}".format(cas))
print("Confusion matrix for SVM is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'SVM'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = svc_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
```

```
print("Accuracy score for training data in SVM is : {}".format(cas))
print("Confusion matrix for SVM is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for SVM is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#K-Nearest Neighbours
y_pred = knn_gs.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in KNN is : {}".format(cas))
print("Confusion matrix for KNN is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'KNN'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = knn_gs.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in KNN is : {}".format(cas))
print("Confusion matrix for KNN is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for KNN is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#BERNOULLI NB
cl1.fit(X_train,y_train)
y_pred = cl1.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Bernoulli NB is : {}".format(ca
s))
print("Confusion matrix for Bernoulli NB is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'BernoulliNB'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = cl1.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Bernoulli NB is : {}".format(ca
s))
print("Confusion matrix for Bernoulli NB is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for Bernoulli NB is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#GAUSSIAN NB
```

```
cl2.fit(X_train,y_train)
y_pred = cl2.predict(X_train)
cas = accuracy_score(y_train,y_pred)
print("Accuracy score for training data in Gaussian NB is : {}".format(cas
))
print("Confusion matrix for Gaussian NB is : ")
print(confusion_matrix(y_pred,y_train))
compiler_compare.loc[row_num, 'Name'] = 'GaussianNB'
compiler_compare.loc[row_num, 'Train Accuracy Score'] = cas*100
y_pred = cl2.predict(X_test)
cas = accuracy_score(y_test,y_pred)
print("Accuracy score for training data in Gaussian NB is : {}".format(cas
))
print("Confusion matrix for Gaussian NB is : ")
print(confusion_matrix(y_pred,y_test))
print("Classification report for Bernoulli NB is : ")
print(classification_report(y_pred,y_test))
compiler_compare.loc[row_num, 'Test Accuracy Score'] = cas*100
row_num+=1
#Accuracy Table

compiler_compare
import seaborn as sns
sns.set_style("dark")
ax = compiler_compare.plot.barh(x='Name', rot=0, figsize=(10, 10))
plt.yticks(size =16)
plt.legend(loc='lower left')
plt.ylabel("Compiler" , size =20)
plt.title("Compiler Accuracy Comparision", size = 20)
for index, value in enumerate(compiler_compare['Test Accuracy Score']):
    plt.text(value+10, index+.12, str(value),size=16)
for index, value in enumerate(compiler_compare['Train Accuracy Score']):
    plt.text(value+10, index-.17, str(value),size=16)
import numpy
import pandas
from keras.utils import np_utils
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical



from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
#from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.constraints import maxnorm



# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load dataset
#dataframe = pandas.read_csv(r"../input/car_evaluation.csv")

# Reading Data Set
from google.colab import drive
drive.mount("/content/gdrive")


# Importing the dataset (Link - https://drive.google.com/file/d/1GBP8p2PML
yo4KK_9CjnTF0g0pDiq85_m/view?usp=sharing)
dataframe = pandas.read_csv('/content/gdrive/MyDrive/Dissertation dataset/
car_evaluation (version 1).csv', header = 0)


# Assign names to Columns
dataframe.columns = ['buying','maint','doors','persons','lug_boot','safety
','classes']

# Encode Data
dataframe.buying.replace(('vhigh','high','med','low'),(1,2,3,4), inplace=T
rue)
dataframe.maint.replace(('vhigh','high','med','low'),(1,2,3,4), inplace=Tr
ue)
dataframe.doors.replace(('2','3','4','5more'),(1,2,3,4), inplace=True)
dataframe.persons.replace(('2','4','more'),(1,2,3), inplace=True)
dataframe.lug_boot.replace(('small','med','big'),(1,2,3), inplace=True)
dataframe.safety.replace(('low','med','high'),(1,2,3), inplace=True)
dataframe.classes.replace(('unacc','acc','good','vgood'),(1,2,3,4), inplac
e=True)
print("dataframe.head: ", dataframe.head())
print("dataframe.describe: ", dataframe.describe())
list = []
for i in dataframe.classes:
```

```
  if pandas.isna(i):
    continue
  else:
    list.append(i)
plt.hist((list))
dataframe.hist()
dataframe
dataset = dataframe.values


X = dataset[:,0:6]
Y = numpy.asarray(dataset[:,6], dtype="int64")

print(X)
print(Y)
# Split Data to Train and Test
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X, Y, test_size=0.2)

# create model
model = Sequential()
model.add(Dense(25, input_dim=6, kernel_initializer='uniform', activation=
'relu'))
model.add(Dense(30, kernel_initializer='uniform', activation='relu', kerne
l_constraint=maxnorm(3)))
model.add(Dropout(0.2))
model.add(Dense(10, kernel_initializer='uniform', activation='relu'))
model.add(Dense(1, kernel_initializer='uniform', activation='relu'))

# Compile model
model.compile(loss='mse', optimizer='adam', metrics=['accuracy'])

# Fit the model
model.fit(X_Train, Y_Train, epochs=600, batch_size=10)

# Evaluate the model
scores = model.evaluate(X_Test, Y_Test)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```