CAR MODEL ACQUISITION

INDEX:

ABSTRACT

The process of buying a pre-owned car can take a lot of time. We find thousands of cars online and offline and spend hours or days comparing them as the pricing of pre-owned cars is not very transparent. The new approach has worked with the requirement for both the client and the merchant to be better educated about the patterns and examples that decide the worth of a pre-owned vehicle on the lookout. Utilizing Machine Learning Algorithms like Linear Regression, Multiple Regression, we will attempt to foster a factual model which will actually want to anticipate the cost of a pre-owned vehicle, in light of past shopper information and a given arrangement of highlights like Engine condition, number of seats etc. We will likewise be contrasting the forecast precision of these models to decide the ideal one. The application deals with the car's acquisition as the primacy. A data set is provided through which the class of the car should be classified. Classification is being carried out through various Machine Learning algorithms.

In this project four algorithms are used and contrasted with the help of visual analysis. The four algorithms are Logistic Regression, K Nearest Neighbors Algorithm, Support Vector Machine and Decision Tree. Out of all the algorithms, Output from Decision Tree was promising as it showed a higher accuracy and precision. Here "class" attribute has been classified into one of the four categories i.e., unacc, acc, vgood, good which actually tells how good a car is and how acceptable it is. Finally, concluding that Car model acquisition can be a challenging task due to the high number of attributes that should be considered for the accurate prediction. The major step in the prediction process is collection and preprocessing of the data. Although, this model has achieved astonishing performance in car sale prediction problem and our aim for the future research is to test this system to work successfully with various data set

1. INTRODUCTION

With used car demand surging, dealers are getting creative with vehicle acquisitions. Buying vehicles directly from the public, also known as "private-party acquisition," is gaining popularity. It's a great way to find quality inventory at prices that boost your bottom line. But not all private-party acquisitions are created equal. Some dealers sit back and wait for deals to come to them. Others proactively pursue the inventory they need. If we need a vehicle fast, procuring from dealer stock will put a vehicle in your driver's hands much quicker than factory ordering. However, in exchange for that expedited timeframe, we're likely to end up paying more. Dealer markup, documentation fees, transportation fees, retail vs. fleet incentives, all of that extra cost is going to be passed on to us. When procuring from dealer stock, there is also no guarantee that the vehicle we want is even going to be available. If the vehicle that we're looking for is popular or production has halted, supply may be limited, and dealers may be reluctant to sell it to non-retail clients. Conversely, ordering vehicles directly from the factory will take longer, but we are guaranteed to get the exact vehicle that we want at the best possible price. Factory ordering vehicles can take anywhere from 3-6 months on average, depending on the manufacturer. This is because each vehicle is custom-made to your specifications. As opposed to being limited to what a dealer has in stock, we get to select each and every option, color, and accessory that we want included, or not included, in the vehicle.

Factory ordering will also allow us to save money because we're omitting the middle-man. We don't have to worry about a litany of fees and ancillary charges. Ordering directly from the factory also allows us to take advantage of fleet incentives that may not otherwise have been available if we purchased through a dealer. Depending on the manufacturer, that could end up saving us thousands of dollars. Vehicle acquisition involves a complex array of decisions, ranging from what types of new vehicles should be purchased when and from whom, to how that purchase will perform over time. When speaking to companies about their vehicle acquisition process, it was always bound up with the management department. Unfortunately, this methodology towards vehicle acquisition can often cause companies to spend more time and money. Developing a clear and well-thought-out strategy for vehicle acquisition will end up paying huge dividends in the long run. Along the same lines, we also need to ensure that the vehicles being offered are appropriate for the function they're going to serve. If the vehicles are going to be used by people who travel more, you may prefer more fuel-efficient vehicles to gas-guzzling pickups or full-size SUVs. If someone is hauling equipment, then we must make sure that the vehicle is large enough to handle the cargo.

We then need to determine the equipment and options that we want to include in each vehicle. Since different vehicles offer different packages, this can sometimes get tricky. We may be able to add a stand-alone moon roof to one vehicle. However, in another vehicle, it may only be available as part of a premium package, thus increasing the price over your budget. It's also

important to determine which equipment consumers expect to be included in a vehicle. This comes into play at resale. A high-end model vehicle with only base features is going to fetch far less on the resale market than one appropriately equipped. When considering which vehicles to purchase, it's essential to know what each will be used for and the specs required to meet those applications. Understanding how a vehicle helps the buyer do his or her job should guide the choice in essence, it's the shopping list that's turned to when evaluating different models. It is essential to match your vehicle selection to the application or function of each vehicle. Vehicles that are too large or have more power than the job requires will have higher operating expenses, such as lower mpg and higher routine maintenance costs. Conversely, if a vehicle is undersized or powered for its job application, there will be increased wear on the vehicle and, thus, increased repairs outside of regular maintenance may be required. Selecting the right vehicle for the job it is performing can lower MPG and indirect costs like fuel expense, insurance expense, and maintenance.

2. SYSTEM REQUIREMENTS AND TOOLS

2.1. HARDWARE:

Device: A High-Level Laptop
Memory And Disk Space: 8gb RAM + 512gb SSD.
Storage: SSD is crucial as we are working with a big dataset.
Processor: Intel i5, 10^{th} Gen (At least)

2.2. SOFTWARE REQUIREMENTS AND TOOLS:

2.2.1. OPERATING SYSTEM: WINDOWS 10

Windows 10 is outlined because it is Microsoft that works with the actual framework for PCs, tablets, and inserted gadgets etc. Microsoft discharged Windows ten, its follow-up to Windows eight. It has been said in the Gregorian calendar month that the window ten is invigorated rather than discharged, and the framework as a successor. When window ten is chosen or received, it will be updated by inheritance squarely from window seven, eight, or window ten. While not meddling in activities such as the framework design methodology, Running Windows ten assists maintenance customers in exchanging the application on the previous software package and installing Windows 10.Shoppers pick up and fill out or refresh window ten. Window ten will be redesigned with the assistance of a window refresh partner to physically begin an associate degree overhaul for Windows. Windows 10 is used to concentrate on adding capabilities that allow IT offices to use mobile phones with board (MDM) programming to anchor and manage gadgets that help run the operating framework. For given broad programming, as an example, the Microsoft Framework Centre Arrangement Chief. Furthermore, Windows Hi adds biometric

verification to Windows 10, allowing customers to sign on with a unique finger impression or facial recognition.

The framework is employed to include virtualization-based security tools, as an example, secluded shopper Mode, Windows Safeguard Gizmo Watch and Windows shielded Qualification Monitor. While attempting to resolve the issue of any strike, Windows 10 is used to keep the highlights of explicit data, procedures, and customer certifications separated. Windows ten is the newer version of Bit Locker secret writing to substantiate data between clients' gadgets, reposting instrumentation, messages, and cloud administrations. Windows eight came up with the new plan and gave touch-empowered motion-driven UI like those on cell phones and tablets, but there wasn't an abundant interpretation of customary work space and digital computer PCs, significantly in business settings. As an example, in Windows ten, Microsoft ventured to deal with this issue and the totally different behaviour of Windows eight, as an example, the associate degree absence of massive business neighborly highlights. The declaration of Windows ten in Gregorian calendar month 2014 by Microsoft was created and window business executives made that point. There was a discharge from Microsoft to Windows ten by seeing the full population in the Gregorian calendar month of 2015. Then shoppers discovered that Windows ten is more cordial than Windows eight as a result of its having an additional typical interface that echoes the work space partaking format of Windows seven. The Windows 10 Anniversary Refresh, which was released in August 2016, created some modifications to the assignment bar and start menu. It also bestowed program enhancements on Edge and provided clients with access to Cortana on the lock screen.In April 2017, Microsoft released the Windows 10 manufacturers' Refresh that created the Windows Hello facial acknowledgment innovation faster and enabled neighbourly spare tabs in Microsoft Edge to examine later.

The Windows 10 fall manufacturers' refresh appeared in the Gregorian calendar month of 2017, adding the Windows Safeguard journey monitor to secure against zero-day assaults. The update also allowed customers and IT to put applications that are running in the background into power-saving mode to save battery life and improve performance.

2.2.2. PROGRAMMING LANGUAGE: PYTHON

Python is a high-level, object-oriented, high-level artificial language with dynamic linguistics. The high-level is formed in information structures, along with dynamic writing and binding, which makes it extremely engaging for rapid application development. It helps with scripting languages wherever the parts are located. Python is referred to as a terribly straightforward language, and it has a straightforward syntax. It helps in reducing the value. Python supports packages which contain code that inspires program code. The in-depth commonplace library is out there in supply and might be freely distributed.

Often, programmers like committing to writing in Python as a result of the productivity it provides. Written material testing and debugging cycles are extremely quick. Debugging is extremely straightforward in python. Whenever the interpreter finds a slip, it generates an exception. If this doesn't happen, then the interpreter prints a stack trace. A supply level computer program helps in examination of native and international variables, analysis of capricious expressions, setting breakpoints, stepping through the code a line at a time, and so on. The quickest way to rectify a program is by adding a few print statements to the supply.

2.2.2. PLATFORM: ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing like data science, machine learning applications, large-scale data processing, predictive analytics, etc., that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, Inc., which was founded by Peter Wang and Travis Oliphant in 2012. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command-line interface (CLI). Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

The following applications are available by default in Navigator

1) JupyterLab
2) Jupyter Notebook
3) QtConsole[19]
4) Spyder
5) Glue
6) Orange
7) RStudio
8) Visual Studio Code

2.2.4. MODULES

2.2.4.1 Numpy:

NumPy also known as Numerical Python is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems. NumPy users include everyone from beginning coders to experienced researchers doing state-of-the-art scientific and industrial research and development. The NumPy API is used extensively in Pandas, SciPy, Matplotlib, scikit-learn, scikit-image and most other data science and scientific Python packages. The NumPy library contains multidimensional array and matrix data structures.

NumPy provides ndarray, a homogeneous n-dimensional array object, with methods to efficiently operate on it. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices. NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

2.2.4.2 Pandas

Pandas is an open-source library that allows to you perform data manipulation, data operation and analysis on numerical tables and time series in Python. It is built on top of NumPy, means it needs NumPy to operate. It easily handles missing data and also uses series for one-dimensional data structure and data frame for multi-dimensional data structure. Pandas provides an efficient way to slice the data and a flexible way to merge, concatenate or reshape the data. In a nutshell, it is a useful library in data analysis. Pandas makes it simple to do many of the time consuming, repetitive tasks associated with working with data, including:

1) Data cleansing
2) Data fill
3) Data normalization
4) Merges and joins
5) Data visualization
6) Statistical analysis
7) Data inspection
8) Loading and saving data

2.2.4.3 Matplotlib

Matplotlib is one of the most popular Python packages used for data visualization. Matplotlib was originally written by John D. Hunter in 2003. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.
Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In matplotlib.pyplot various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes. Similar to matplotlib, we have few other packages which serve different purpose as shown below.

1) Basemap: It is a map plotting toolkit with various map projections, coastlines and political boundaries.
2) Cartopy: It is a mapping library featuring object-oriented map projection definitions, and arbitrary point, line, polygon and image transformation capabilities.
3) Excel tools: Matplotlib provides utilities for exchanging data with Microsoft Excel.
4) Mplot3d: It is used for 3-D plots.
5) Natgrid: It is an interface to the natgrid library for irregular gridding of the spaced data.

Some of the frequently used plots in data analysis are

1) Bar Graph
2) Histogram
3) Scatter Plot
4) Pie Plot
5) Area Plot

2.2.4.4 Seaborn

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs. Visualization is the central part of

Seaborn which helps in exploration and understanding of data. Seaborn contains a number of patterns and plots for data visualization. It uses fascinating themes. It helps in compiling whole data into a single plot. It also provides distribution of data. Seaborn is more comfortable in handling Pandas data frames. It uses basic sets of methods to provide beautiful graphics in python. It offers the following functionalities

1) Dataset oriented API to determine the relationship between variables.
2) Automatic estimation and plotting of linear regression plots.
3) It supports high-level abstractions for multi-plot grids.
4) Visualizing univariate and bivariate distribution.

Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its dataset-oriented, declarative API lets you focus on what the different elements of your plots mean, rather than on the details of how to draw them. Using Seaborn we can plot wide varieties of plots like:

1) Distribution Plots
2) Pie Chart & Bar Chart
3) Scatter Plots
4) Pair Plots
5) Heat maps

2.2.4.5 Math

The Python Math Library provides us access to some common math functions and constants in Python, which we can use throughout our code for more complex mathematical computations. Math module provides value of various constants like pi, tau. Having such constants saves the time of writing the value of each constant every time we want to use it and that too with great precision. Constants provided by the math module are

1) Euler's Number
2) Pi
3) Tau
4) Infinity
5) Not a Number (NaN)

2.2.4.6 Sklearn (Scikit-learn)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. It was initially developed by David Cournapeau as a Google summer of code project in 2007. Later, in 2010, Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel, from FIRCA (French Institute for Research in Computer Science and Automation), took this project at another level and made the first public release (v0.1 beta) on 1st Feb. 2010.

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows

1) Supervised Learning algorithms: Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
2) Unsupervised Learning algorithms: On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
3) Clustering: This model is used for grouping unlabeled data.
4) Cross Validation: It is used to check the accuracy of supervised models on unseen data.
5) Dimensionality Reduction: It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
6) Ensemble methods: As name suggest, it is used for combining the predictions of multiple supervised models.
7) Feature extraction: It is used to extract the features from data to define the attributes in image and text data.
8) Feature selection: It is used to identify useful attributes to create supervised models.
9) Open Source: It is open source library and also commercially usable under BSD license.

3. WHAT IS CAR MODEL ACQUISITION?

3.1. Problem Statement:

The incentive of this project is to determine whether or not to purchase the car based on the class of the car. A dataset with succinct details about various cars ready to be resold is provided. Using the information in the dataset, the algorithms are to be trained and need to classify each car into either class, i.e., unacc, good, vgood, or acc.

3.2. Objective:

We have businesses that deal with the sale of automobiles, like cars. They bring up the cars which have been already used. Ultimately, alterations on the bought cars are done and they are resold. It is critical for enterprises to know the kind of cars that they must look out for. There are numerous cars with own set of characteristics. Analyzing each and every attribute is a difficult undertaking that takes a lot of time and effort. As a result, an algorithm that can analyze the characteristics of a car and determine whether or not to purchase it is required. Our project satisfies the problem's criteria and allows for a speedy choice.

3.3.Organization:

Initially, the dataset is loaded into the Jupyter notebook on the Anaconda platform and examined to retrieve inferences. In order to facilitate machine learning algorithms, data preprocessing is performed on the data set. Moreover, as some of the tuples present in the data set might have redundant or irrelevant data, we eliminate such tuples and select the appropriate ones accordingly. We also look if there are any missing values, and if so, we eliminate such tuples to facilitate the classifiers. A bunch of algorithms like KNN, SVM, Decision Tree, and Logistic Regression are now trained and tested. A random input is provided to test the classifiers and to predict whether or not to acquire the car.

4. ARCHITECTURE

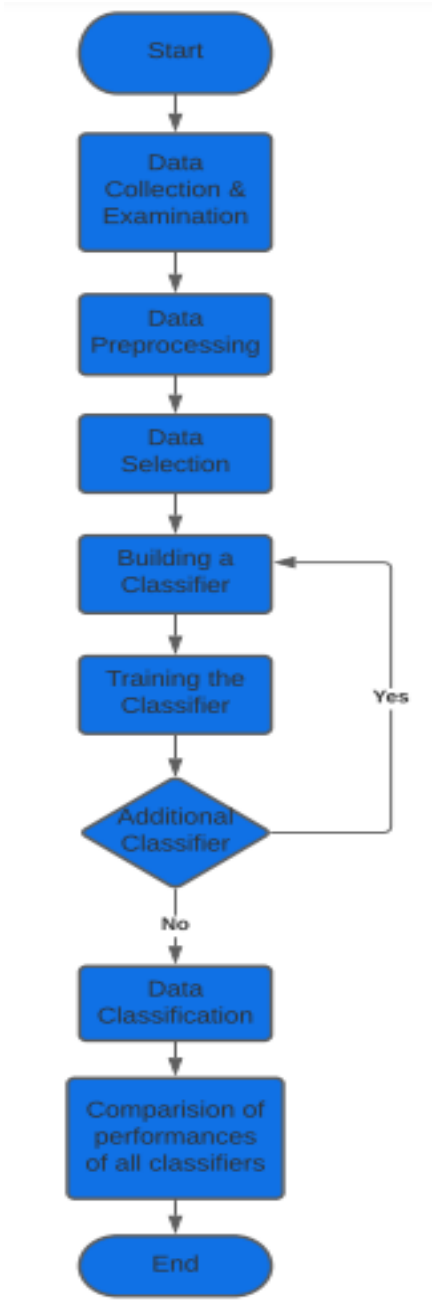

Fig 1. Architecture

5. PROPOSED SYSTEM

5.1. DATA COLLECTION:

As a society, we're generating data at an unprecedented rate. These data can be Numeric like temperature, loan amount, customer retention rate, and Categorical like gender, color, highest degree earned. Data collection is the process of gathering and measuring information from countless different sources. In order to use the data we collect to develop practical artificial intelligence (AI) and machine learning solutions, it must be collected and stored in a way that makes sense for the business problem at hand. Collecting data allows us to capture a record of past events so that we can use data analysis to find recurring patterns. From those patterns, we build predictive models using machine learning algorithms that look for trends and predict future changes.

Data collection is a major bottleneck in machine learning and an active research topic in multiple communities. There are largely two reasons data collection has recently become a critical issue. First, as machine learning is becoming more widely-used, we are seeing new applications that do not necessarily have enough labeled data. Second, unlike traditional machine learning, deep learning techniques automatically generate features, which saves feature engineering costs, but in return may require larger amounts of labeled data. Interestingly, recent research in data collection comes not only from the machine learning, natural language, and computer vision communities, but also from the data management community due to the importance of handling large amounts of data. Data collection largely consists of data acquisition, data labeling, and improvement of existing data or models.

Predictive models are only as good as the data from which they are built, so good data collection practices are crucial to developing high-performing models. The data need to be error-free (garbage in, garbage out) and contain relevant information for the task at hand. Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection

1. Inaccurate data: The collected data could be unrelated to the problem statement.
2. Missing data: Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
3. Data imbalance: Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.

4. Data bias: Depending on how the data, subjects and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age or region, for example. Data bias is difficult to detect and remove.

Several techniques can be applied to address those problems:

1. Pre-cleaned, freely available datasets: If the problem statement (for example, image classification, object recognition) aligns with a clean, pre-existing, properly formulated dataset, then take advantage of existing, open-source expertise.
2. Web crawling and scraping: Automated tools, bots and headless browsers can crawl and scrape websites for data.
3. Private data: ML engineers can create their own data. This is helpful when the amount of data required to train the model is small and the problem statement is too specific to generalize over an open-source dataset.
4. Custom data: Agencies can create or crowd source the data for a fee.

In our project, we utilized the "car.dataset" to train the machine learning algorithms. The "car.dataset" had the data with a total of 1727 rows and 7 columns. The dataset had a total of six attributes that were treated as inputs, and another single attribute was treated as the output. The six attributes are passed through the machine learning classifier as the input. The input attributes, which are also known as the characteristics of the car, are: buying, maintenance, doors, persons, lug_boot, and safety, whereas the output attribute was class.

5.2. DATA PREPROCESSING:

Some say that garbage in equals garbage out, and this is definitely the case. This basically means that we may have built a predictive model, but it doesn't matter if the data used to build the model is non-representative, low quality, error-ridden, and so on. The quality, amount, preparation, and selection of data is critical to the success of a machine learning solution. The first step to ensure success is to avoid selection bias. Selection bias occurs when the samples used to produce the model are not fully representative of cases that the model may be used for in the future, particularly with new and unseen data. Real-world raw data and images are often incomplete, inconsistent and lacking in certain behaviors or trends. They are also likely to contain many errors. So, once collected, they are pre-processed into a format the machine learning algorithm can use for the model.

Data is typically messy and often consists of missing values, useless values (e.g., NA), outliers, and so on. Prior to modeling and analysis, raw data needs to be parsed, cleaned, transformed, and pre-processed. This is typically referred to a data munging or data wrangling. For missing data, data is often imputed, which is a technique used to fill in, or substitute for missing values, and is very similar conceptually to interpolation. Feature selection becomes prominent, especially in the

data sets with many variables and features. It will eliminate unimportant variables and improve the accuracy as well as the performance of classification. Besides, we evaluate and compare each accuracy and performance of the classification model, such as, Logistic Regression, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Decision Trees. The highest accuracy of the model is the best classifier.

Pre-processing includes a number of techniques and actions:

1. Data cleaning: These techniques, manual and automated, remove data incorrectly added or classified.
2. Data imputations: Most ML frameworks include methods and APIs for balancing or filling in missing data. Techniques generally include imputing missing values with standard deviation, mean, median and k-nearest neighbors (knn) of the data in the given field.
3. Oversampling: Bias or imbalance in the dataset can be corrected by generating more observations/samples with methods like repetition, bootstrapping or Synthetic Minority Over-Sampling Technique (SMOTE), and then adding them to the under-represented classes.
4. Data integration: Combining multiple datasets to get a large corpus can overcome incompleteness in a single dataset.
5. Data normalization: The size of a dataset affects the memory and processing required for iterations during training. Normalization reduces the size by reducing the order and magnitude of data.

In our project, we performed data preprocessing on the dataset by eliminating the tuples that had null values or the values of unwanted data type. Moreover, through data visualization, we had come to know that there were no null values and that the dataset was well organized, with all the rows and columns completely filled with the appropriate data type. Later, as part of the data preprocessing, we replaced the names of columns in the dataset with new names in order to facilitate the machine learning algorithms. But, we had another problem in the form of categorical data. Usually, machine learning algorithms are fond of numerical data, whereas the data that was present in our dataset was categorical data, which is not recognizable by the machine learning algorithms. Thus, there is a need for the conversion of the categorical data into numerical data. This process is also known as Label Encoding. Again, as part of data preprocessing, we converted the categorical data to numerical data, where we replaced the data values with numerical values.

5.3. DATA SPLIT AND CLASSIFICATION:

5.3.1. DATA SPLIT:

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow us to compare the performance of machine learning algorithms for the predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when we have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

1. Train Dataset: Used to fit the machine learning model.
2. 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. This is how we expect to use the model in practice. Namely, to fit it on available data with known inputs and outputs, then make predictions on new examples in the future where we do not have the expected output or target values. The train-test procedure is appropriate when there is a sufficiently large dataset available. The idea of "sufficiently large" is specific to each predictive modeling problem. It means that there is enough data to split the dataset into train and test datasets and each of the train and test datasets are suitable representations of the problem domain. This requires that the original dataset is also a suitable representation of the problem domain.

A suitable representation of the problem domain means that there are enough records to cover all common cases and most uncommon cases in the domain. This might mean combinations of input variables observed in practice. It might require thousands, hundreds of thousands, or millions of examples. Conversely, the train-test procedure is not appropriate when the dataset available is small. The reason is that when the dataset is split into train and test sets, there will not be enough data in the training dataset for the model to learn an effective mapping of inputs to outputs. There

will also not be enough data in the test set to effectively evaluate the model performance. The estimated performance could be overly optimistic (good) or overly pessimistic (bad).

The scikit-learn Python machine learning library provides an implementation of the train-test split evaluation procedure via the train_test_split() function. The function takes a loaded dataset as input and returns the dataset split into two subsets. Ideally, we can split the original dataset into input (X) and output (y) columns, then call the function passing both arrays and have them split appropriately into train and test subsets. The size of the split can be specified via the "test_size" argument that takes a number of rows (integer) or a percentage (float) of the size of the dataset between 0 and 1. In our project we have used 60 percent of the data present in the dataset as the training data and remaining 40 percent of the data present in the dataset as the testing dataset to test the trained machine learning algorithm.

5.3.2. CLASSIFICATION:

In machine learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. Examples of classification problems include:

1. Given an example, classify if it is spam or not.
2. Given a handwritten character, classify it as one of the known characters.
3. Given recent user behavior, classify as churn or not.

From a modeling perspective, classification requires a training dataset with many examples of inputs and outputs from which to learn. A model will use the training dataset and will calculate how to best map examples of input data to specific class labels. As such, the training dataset must be sufficiently representative of the problem and have many examples of each class label.
Class labels are often string values, e.g. "spam," "not spam," and must be mapped to numeric values before being provided to an algorithm for modeling. This is often referred to as label encoding, where a unique integer is assigned to each class label, e.g. "spam" = 0, "no spam" = 1. There are many different types of classification algorithms for modeling classification predictive modeling problems. Below are the algorithms used for classification:

1. Logistic Regression
2. K-Nearest Neighbors
3. Support Vector Machine
4. Decision Trees

5.3.2.1. LOGISTIC REGRESSION:

Logistic regression is generally used where we have to classify the data into two or more classes. One is binary and the other is multi-class logistic regression. As the name suggests, the binary class has 2 classes that are Yes/No, True/False, 0/1, etc. In multi-class classification, there are

more than 2 classes for classifying data. Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for a given set of features(or inputs), X. Contrary to popular belief, logistic regression is a regression model. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like Linear regression, it assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

The sigmoid function is a mathematical function used to map the predicted values to probabilities. It maps any real value into another value within a range of 0 and 1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function. In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

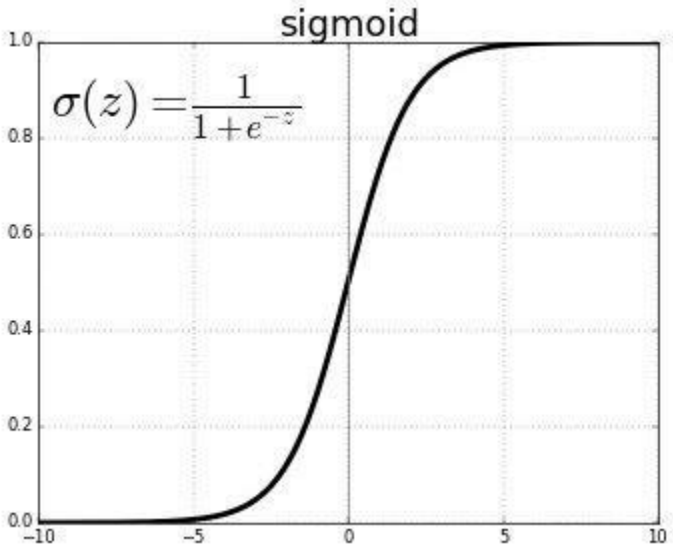

Fig 2. Sigmoid Function

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

1. We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

2. In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$\frac{y}{1-y}$; 0 for y= 0, and infinity for y=1

3. But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

The above equation is the final equation for Logistic Regression.

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

1. **Binomial:** In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
2. **Multinomial:** In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
3. **Ordinal:** In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

5.3.2.2. K- NEAREST NEIGHBORS:

K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new data point. The algorithm's learning is:

1. Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.
2. Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.
3. Non -Parametric: In KNN, there is no predefined form of the mapping function.

How KNN Works:

Consider the following figure. Let us say we have plotted data points from our training set on a two-dimensional feature space. As shown, we have a total of 6 data points (3 red and 3 blue).

Red data points belong to ‘class1’ and blue data points belong to ‘class2’. And yellow data point in a feature space represents the new point for which a class is to be predicted. Obviously, we say it belongs to ‘class1’ (red points)

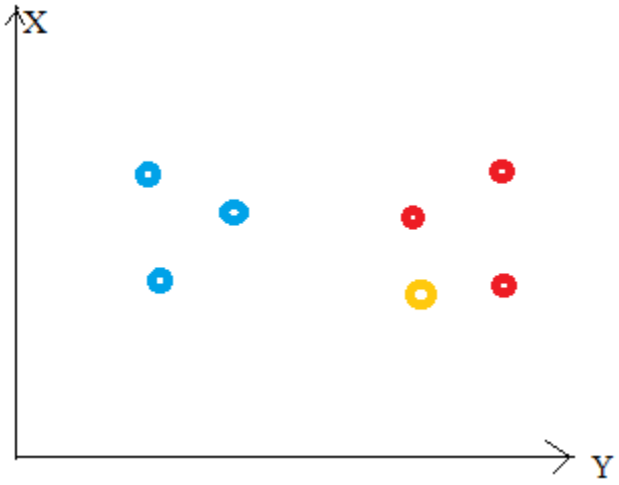

Fig 3. KNN

But why? Because its nearest neighbors belong to that class.! This is the principle behind K Nearest Neighbors. Here, nearest neighbors are those data points that have minimum distance in feature space from our new data point. And K is the number of such data points we consider in our implementation of the algorithm. Therefore, distance metric and K value are two important considerations while using the KNN algorithm. Euclidean distance is the most popular distance metric. You can also use Hamming distance, Manhattan distance, Minkowski distance as per your need. For predicting class/ continuous value for a new data point, it considers all the data points in the training dataset. Finds new data point’s ‘K’ Nearest Neighbors (Data points) from feature space and their class labels or continuous values. For classification: A class label assigned to the majority of K Nearest Neighbors from the training dataset is considered as a predicted class for the new data point. For regression: Mean or median of continuous values assigned to K Nearest Neighbors from training dataset is a predicted continuous value for our new data point

How to choose the value for K?

K is a crucial parameter in the KNN algorithm. Some suggestions for choosing K Value are:

1) Using error curves: The figure below shows error curves for different values of K for training and test data.

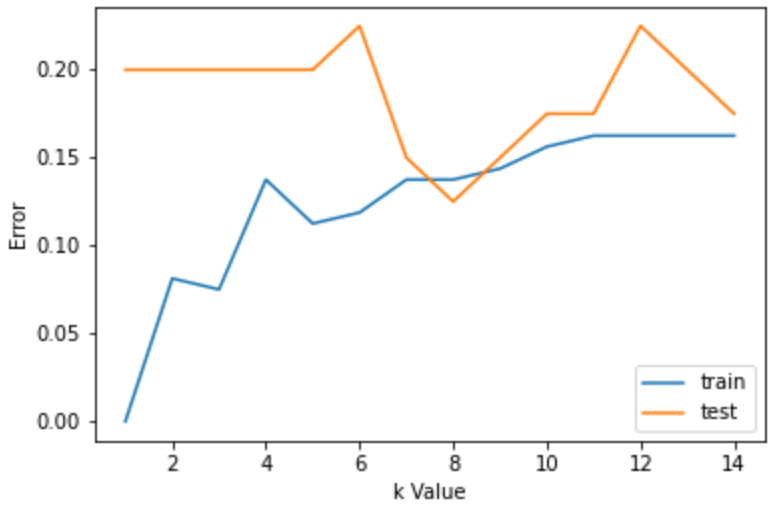

Fig 4. Choosing a value for K

- At low K values, there is overfitting of data/high variance. Therefore test error is high and train error is low. At K=1 in train data, the error is always zero, because the nearest neighbor to that point is that point itself. Therefore though training error is low test error is high at lower K values. This is called overfitting. As we increase the value for K, the test error is reduced.

- But after a certain K value, bias/ underfitting is introduced and test error goes high. So we can say initially test data error is high(due to variance) then it goes low and stabilizes and with further increase in K value, it again increases(due to bias). The K value when test error stabilizes and is low is considered as optimal value for K. From the above error curve, we can choose K=8 for our KNN algorithm implementation.

2) Also, domain knowledge is very useful in choosing the K value.
3) K value should be odd while considering binary(two-class) classification.

Required Data Preparation:

1) Data Scaling: To locate the data point in multidimensional feature space, it would be helpful if all features are on the same scale. Hence normalization or standardization of data will help.
2) Dimensionality Reduction: KNN may not work well if there are too many features. Hence dimensionality reduction techniques like feature selection, principal component analysis can be implemented.

3) Missing value treatment: If out of M features one feature data is missing for a particular example in the training set then we cannot locate or calculate distance from that point. Therefore deleting that row or imputation is required.

5.3.2.3. SUPPORT VECTOR MACHINES:

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification and regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features that we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

Hyperplane and Support Vectors in the SVM algorithm:

1) Hyperplane:
- There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.
- The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in image), then hyperplane will be a straight line. And if there are 3 features, then hyperplane will be a 2-dimension plane.
- We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

2) Support Vectors:
- The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

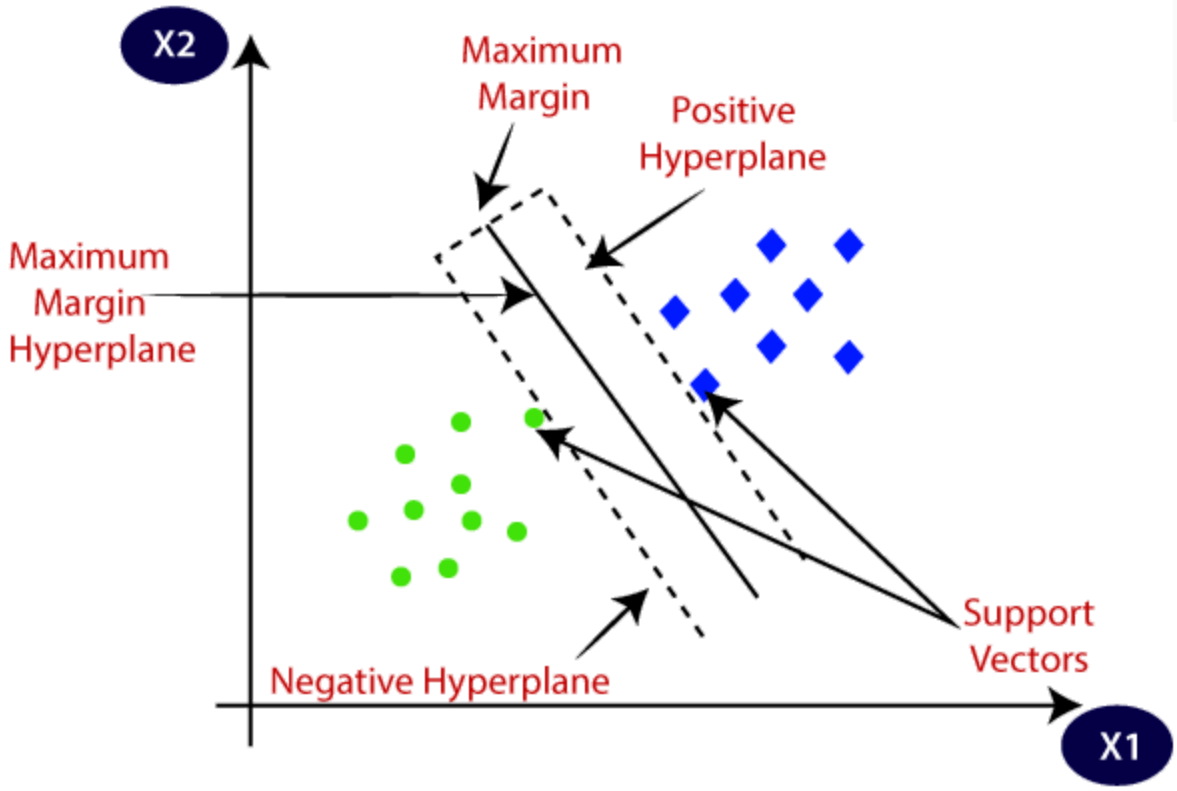

Fig 5. SVM

Types of SVM

1) Linear SVM : Linear SVM is used for data that are linearly separable i.e. for a dataset that can be categorized into two categories by utilizing a single straight line. Such data points are termed as linearly separable data, and the classifier is used described as a Linear SVM classifier.
2) Non-linear SVM: Non-Linear SVM is used for data that are non-linearly separable data i.e. a straight line cannot be used to classify the dataset. For this, we use something known as a kernel trick that sets data points in a higher dimension where they can be separated using planes or other mathematical functions. Such data points are termed as non-linear data, and the classifier used is termed as a Non-linear SVM classifier.

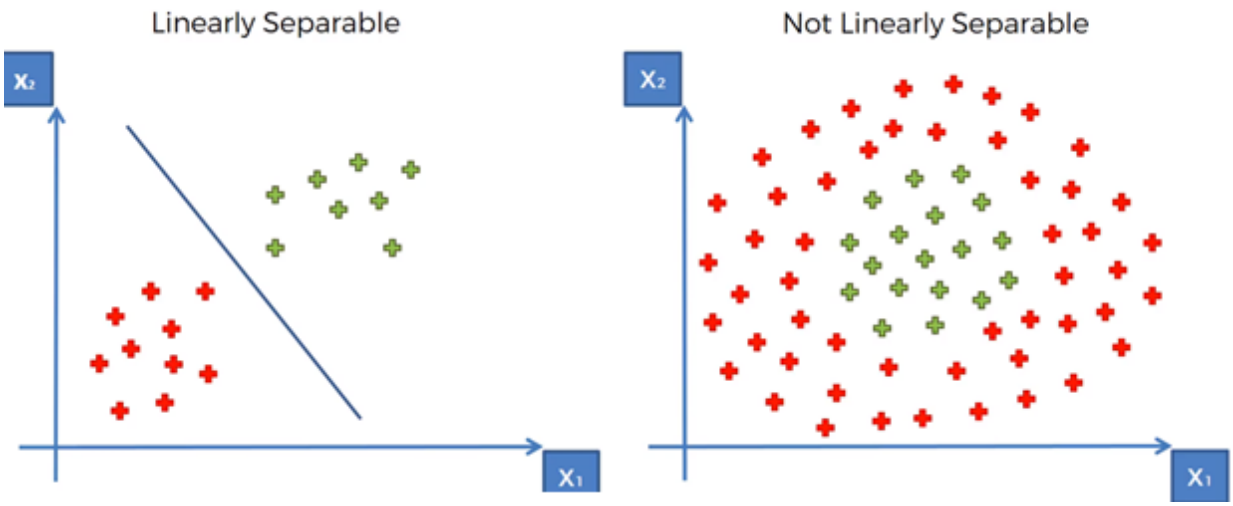

Fig 6. Types of SVM

5.3.2.4. DECISION TREES:

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

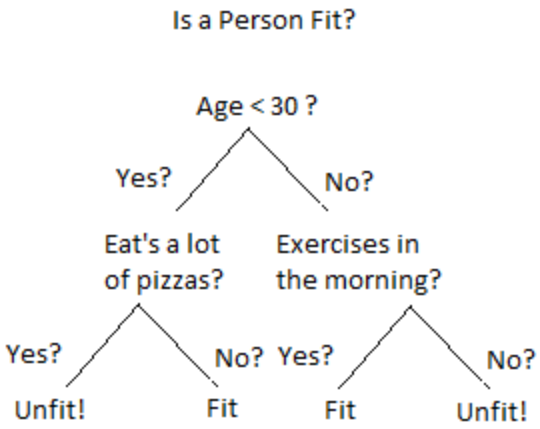

Fig 7. Decision Tree

There are two main types of Decision Trees:

1) Classification trees (Yes/No types):
   The outcome where we have a variable like ‘fit’ or ‘unfit’, ‘yes’ or ‘no’, ‘pass’ or ‘fail’, etc come under classification trees . Here the decision variable is Categorical.
2) Regression trees (Continuous data types):
   Here the decision or the outcome variable is Continuous, e.g. a number like 123. There are many algorithms out there which construct Decision Trees, but one of the best is called as ID3 Algorithm. ID3 Stands for Iterative Dichotomiser 3.

Decision Tree Terminologies

1) Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
2) Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3) Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4) Branch/Sub Tree: A tree formed by splitting the tree.
5) Pruning: Pruning is the process of removing the unwanted branches from the tree.
6) Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

5.4. TESTING ON SPECIFIC DATA:

As we have trained the machine learning algorithms, it is time for us to test these algorithms to evaluate how accurate they are at classification. Thus, we pass some random inputs to the classifiers and predict their output. As the algorithms have considered six attributes as the inputs, we have to pass six numerical inputs. Once the inputs are received at the classifier end, the output is predicted and the value lies between 0 and 5 (the value might be 1, 2, 3, or 4). By testing in this way, when we actually deal with this project, we can determine whether or not to acquire the car on a real-time basis.

7. SCREENSHOTS

```
#Replacing column names with other names
df = pd.read_csv("car.data")
df.rename(columns = {'vhigh':'buying','vhigh.1':'maintenance','2':'doors','2.1':'persons','small':'lug_boot','low':'safety','unac
df
```

Fig 8. Importing the dataset

	buying	maintenance	doors	persons	lug_boot	safety	Class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc
...	...	...	...	...	...	...	...
1722	low	low	5more	more	med	med	good
1723	low	low	5more	more	med	high	vgood
1724	low	low	5more	more	big	low	unacc
1725	low	low	5more	more	big	med	good
1726	low	low	5more	more	big	high	vgood

1727 rows × 7 columns

Fig 9. Dataset

```
print(df.isnull())
```

```
      buying  maintenance  doors  persons  lug_boot  safety  Class
0      False        False  False    False     False   False  False
1      False        False  False    False     False   False  False
2      False        False  False    False     False   False  False
3      False        False  False    False     False   False  False
4      False        False  False    False     False   False  False
...      ...          ...    ...      ...       ...     ...    ...
1722   False        False  False    False     False   False  False
1723   False        False  False    False     False   False  False
1724   False        False  False    False     False   False  False
1725   False        False  False    False     False   False  False
1726   False        False  False    False     False   False  False

[1727 rows x 7 columns]
```

Fig 10. Checking the presence of missing data

```
plt.figure(figsize = (10,10))
sns.heatmap(df.isnull(),yticklabels = False, cbar = False,cmap = 'plasma')
#Hence No Null Values or Missing Data
```

```
<AxesSubplot:>
```

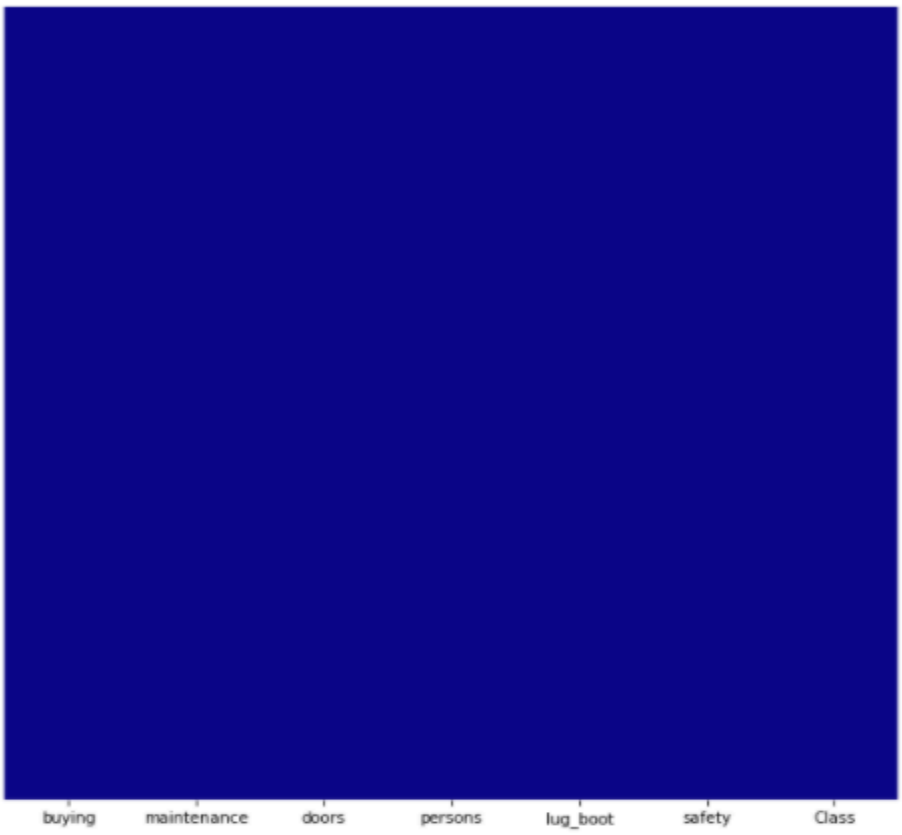

Fig 11. Data Visualization of Missing Data on the Data Set

```
df['buying'] = df['buying'].replace(['vhigh','high','med','low'],[4,3,2,1])
df['maintenance'] = df['maintenance'].replace(['vhigh','high','med','low'],[4,3,2,1])
df['doors'] = df['doors'].replace(['2','3','4','5more'],[2,3,4,5])
df['persons'] = df['persons'].replace(['2','4','more'],[2,4,5])
df['lug_boot'] = df['lug_boot'].replace(['small','med','big'],[1,2,3])
df['safety'] = df['safety'].replace(['med','high','low'],[2,3,1])
df['Class'] = df['Class'].replace(['unacc','acc','vgood','good'],[1,4,3,2])
```

Fig 12. Handling the Categorical Data

```
df
```

	buying	maintenance	doors	persons	lug_boot	safety	Class
0	4	4	2	2	1	2	1
1	4	4	2	2	1	3	1
2	4	4	2	2	2	1	1
3	4	4	2	2	2	2	1
4	4	4	2	2	2	3	1
...	...	...	...	...	...	...	...
1722	1	1	5	5	2	2	2
1723	1	1	5	5	2	3	3
1724	1	1	5	5	3	1	1
1725	1	1	5	5	3	2	2
1726	1	1	5	5	3	3	3

1727 rows × 7 columns

Fig 13. Updated Dataset After Handling the Categorical Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop('Class',axis=1),df['Class'], test_size=0.40,
                                                    random_state=101)
```

Fig 14. Data Split

7. RESULTS

```
from sklearn.linear_model import LogisticRegression
lm = LogisticRegression(max_iter=800)
```

```
lm.fit(X_train,y_train)
```

```
LogisticRegression(max_iter=800)
```

```
y_pred_lm = lm.predict(X_test)
```

Fig 15. Applying Logistic Regression

```
from sklearn.metrics import confusion_matrix , classification_report, accuracy_score
accuracy = []
cm_lm = confusion_matrix(y_test,y_pred_lm)
cm_lm
```

```
array([[465,   5,   0,  22],
       [  1,  10,   2,  15],
       [  0,   0,  17,   8],
       [ 40,   5,   3,  98]], dtype=int64)
```

```
print(classification_report(y_test,y_pred_lm))
```

```
              precision    recall  f1-score   support

           1       0.92      0.95      0.93       492
           2       0.50      0.36      0.42        28
           3       0.77      0.68      0.72        25
           4       0.69      0.67      0.68       146

    accuracy                           0.85       691
   macro avg       0.72      0.66      0.69       691
weighted avg       0.85      0.85      0.85       691
```

```
accuracy.append(accuracy_score(y_test,y_pred_lm))
```

Fig 16. Performance of Logistic Regression

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=42)
```

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=42)
```

```
y_pred_knn = knn.predict(X_test)
```

Fig 17. Applying KNN

```
cm_knn = confusion_matrix(y_test,y_pred_knn)
cm_knn
```

```
array([[484,   0,   0,   8],
       [  3,  12,   1,  12],
       [  0,   4,   8,  13],
       [ 17,   3,   0, 126]], dtype=int64)
```

```
print(classification_report(y_test,y_pred_knn))
```

```
              precision    recall  f1-score   support

           1       0.96      0.98      0.97       492
           2       0.63      0.43      0.51        28
           3       0.89      0.32      0.47        25
           4       0.79      0.86      0.83       146

    accuracy                           0.91       691
   macro avg       0.82      0.65      0.69       691
weighted avg       0.91      0.91      0.90       691
```

```
accuracy.append(accuracy_score(y_test,y_pred_knn))
```

Fig 18. Performance of KNN

```
from sklearn.svm import SVC
svc = SVC()
```

```
svc.fit(X_train,y_train)
```

```
SVC()
```

```
y_pred_svc = svc.predict(X_test)
```

Fig 19. Applying SVM

```
cm_svc = confusion_matrix(y_test,y_pred_svc)
cm_svc
```

```
array([[481,   0,   0,  11],
       [  0,  21,   1,   6],
       [  0,   0,  18,   7],
       [  4,   4,   0, 138]], dtype=int64)
```

```
print(classification_report(y_test,y_pred_svc))
```

```
              precision    recall  f1-score   support

           1       0.99      0.98      0.98       492
           2       0.84      0.75      0.79        28
           3       0.95      0.72      0.82        25
           4       0.85      0.95      0.90       146

    accuracy                           0.95       691
   macro avg       0.91      0.85      0.87       691
weighted avg       0.95      0.95      0.95       691
```

```
accuracy.append(accuracy_score(y_test,y_pred_svc))
```

Fig 20. Performance SVM

```
from sklearn.tree import DecisionTreeClassifier
dst = DecisionTreeClassifier(max_depth = 50, random_state = 0,criterion = 'entropy')
```

```
dst.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=50, random_state=0)
```

```
y_pred_dst = dst.predict(X_test)
```

Fig 21. Applying Decision Tree

```
cm_dst = confusion_matrix(y_test,y_pred_dst)
cm_dst
```

```
array([[488,   0,   0,   4],
       [  0,  24,   0,   4],
       [  0,   0,  24,   1],
       [  9,   3,   0, 134]], dtype=int64)
```

```
print(classification_report(y_test,y_pred_dst))
```

```
              precision    recall  f1-score   support

           1       0.98      0.99      0.99       492
           2       0.89      0.86      0.87        28
           3       1.00      0.96      0.98        25
           4       0.94      0.92      0.93       146

    accuracy                           0.97       691
   macro avg       0.95      0.93      0.94       691
weighted avg       0.97      0.97      0.97       691
```

```
accuracy.append(accuracy_score(y_test,y_pred_dst))
```

Fig 22. Performance of Decision Tree

```
ML_Algo = ['Logistic Regression','KNN','SVM','Decision Tree']
df = pd.DataFrame(list(zip(ML_Algo, accuracy)),columns =['Algorithm_Name', 'Score'])
```

```
vrep = sns.light_palette("green", as_cmap = True)
s = df.style.background_gradient(cmap = vrep)
s
```

	Algorithm_Name	Score
0	Logistic Regression	0.853835
1	KNN	0.911722
2	SVM	0.952243
3	Decision Tree	0.969609

Fig 23. Comparing Performance of all the Algorithms

8. OUTPUT

```
sns.set(style = "whitegrid")
ax = sns.barplot(y = "Algorithm_Name",x = "Score",data = df)
```

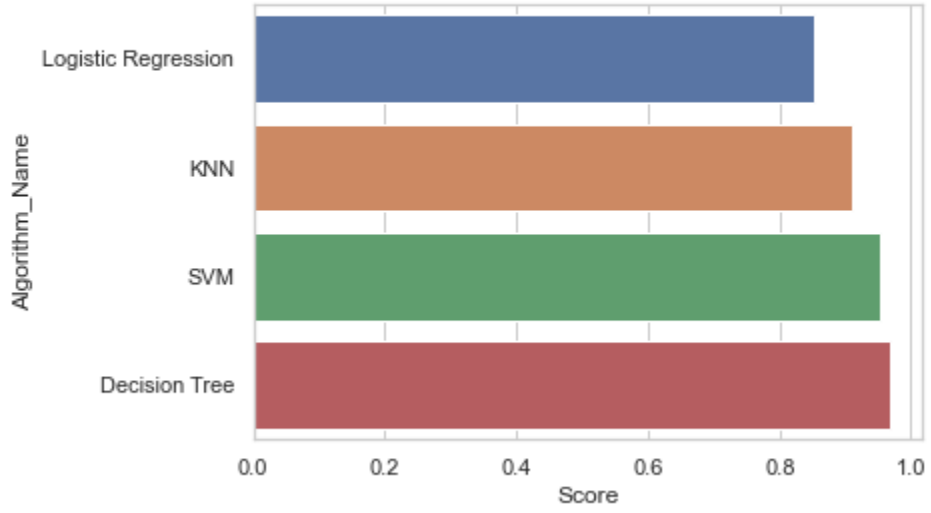

Fig 25. Graphical Representation of Performance of Various Algorithms

```
new_input = [[1,1,5,5,2,2]]

print(dst.predict(new_input))
[2]
```

Fig 26. Testing on Random Input

9. CONCLUSION

Car model acquisition is a project that assists a variety of businesses involved in the resale of automobiles, particularly cars. In this project, we developed a method for selecting cars based on their attributes. Every automobile has its unique features, such as seating capacity, baggage boot, maintenance, and safety, to name a few. However, our goal is to select automobiles that possess critical features. Various machine learning methods, such as SVM, KNN, decision trees, and logistic regression, were trained using a dataset. The algorithms were examined after the training and found to have varying levels of categorization accuracy. The Decision Tree was the most reliable of the four algorithms, effectively classifying the cars with an accuracy of 96.96%.

Car acquisition can be a challenging task due to the high number of attributes that should be considered for the accurate prediction. The major step in the prediction process is collection and preprocessing of the data. In this research data preprocessing is carried out in order to normalize, standardize and clean data to avoid unnecessary noise for machine learning algorithms. Data cleaning is one of the processes that increases prediction performance, yet insufficient for the cases of complex data sets as the one in this research. Although, this system has achieved astonishing performance, our aim for the future research is to test this system to work successfully with various data sets. We will extend our test data with eBay and OLX used cars data sets and validate the proposed approach. Moreover, it would be fascinating to see what a combination of more classifiers might produce.

10. BIBLIOGRAPHY

[1] Agencija za statistiku BiH. (n.d.), retrieved from: http://www.bhas.ba . [accessed July 18, 2018.]

[2] Listiani, M. (2009). Support vector regression analysis for price prediction in a car leasing application (Doctoral dissertation, Master thesis, TU Hamburg-Harburg).

[3] Richardson, M. S. (2009). Determinants of used car resale value. Retrieved from: https://digitalcc.coloradocollege.edu/islandora/object /coccc%3A1346 [accessed: August 1, 2018.]

[4] Wu, J. D., Hsu, C. C., & Chen, H. C. (2009). An expert system of price forecasting for used cars using adaptive neuro-fuzzy inference. Expert Systems with Applications, 36(4), 7809-7817.

[5] Du, J., Xie, L., & Schroeder, S. (2009). Practice Prize Paper—PIN Optimal Distribution of Auction Vehicles System: Applying Price Forecasting, Elasticity Estimation, and Genetic Algorithms to Used-Vehicle Distribution. Marketing Science, 28(4), 637-644.

[6] Gongqi, S., Yansong, W., & Qiang, Z. (2011, January). New Model for Residual Value Prediction of the Used Car Based on BP Neural Network and Nonlinear Curve Fit. In Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on (Vol. 2, pp. 682-685). IEEE.

[7] Pudaruth, S. (2014). Predicting the price of used cars using machine learning techniques. Int. J. Inf. Comput. Technol, 4(7), 753-764.

[8] Noor, K., & Jan, S. (2017). Vehicle Price Prediction System using Machine Learning Techniques. International Journal of Computer Applications, 167(9), 27-31.

[9] Auto pijaca BiH. (n.d.), Retrieved from: https://www.autopijaca.ba. [accessed August 10, 2018].

[10] Weka 3 - Data Mining with Open Source Machine Learning Software in Java. (n.d.), Retrieved from: https://www.cs.waikato.ac.nz/ml/weka/. [August 04, 2018].

[11] Ho, T. K. (1995, August). Random decision forests. In Document analysis and recognition, 1995., proceedings of the third international conference on (Vol. 1, pp. 278-282). IEEE.

[12] Russell, S. (2015). Artificial Intelligence: A Modern Approach (3rd edition). PE.

[13] Ben-Hur, A., Horn, D., Siegelmann, H. T., & Vapnik, V. (2001). Support vector clustering. Journal of machine learning research, 2(Dec), 125-137.

[14] Aizerman, M. A. (1964). Theoretical foundations of the potential function method in pattern recognition learning. Automation and remote control, 25, 821- 837.

[15] 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.19.2 documentation. (n.d.). Retrieved from: http://scikitlearn.org/stable/modules/generated/sklearn.ensemble .RandomForestClassifier.html [accessed: August 30, 2018].

[16] Used cars database. (n.d.) Retrieved from: https://www.kaggle.com/orgesleka/used-carsdatabase. [accessed: June 04, 2018].

[17] OLX. (n.d.), Retrieved from: https://olx.ba. [accessed August 05,2018].