Delivery and Tracking application

A report Submitted
For partial completion of the degree of
Bachelor of Engineering

By

Md Hammad Rasheed (211220200016)
Md Amash Shams(211220200001)

Under the supervision of
Mr. Md Zeeshan Rasheed, Last Year Btech Student, Dept. of IT
(Duration : 4th May, 2023 to 3rd June, 2023)

IMARAT INSTITUTE OF COMPUTER & ELECTRONICS, PHUWARISHARIFF, PATNA, BIHAR, INDIA 2023

ACKNOWLEDGEMENT

We take this opportunity to thank all the people who have rendered their full support to our project work. We render our thanks to our mam and sir, Professor of our college who encouraged us to do the Project. We are grateful to Perveen Agarwal, CCO Textbook for facilitating all the amenities required for carrying out this project. We express our sincere gratitude to skill academy textbook faculties, Dean Academics for providing an excellent environment in the college. We are also thankful to Tamosa Mam, Ahana Guchait Mam and Pijus Bairi Sir, faculties of the Department for providing us with both time and amenities to make this project a success within the given schedule. We are also thankful to our guide Uttam Grade sir Skill Academy Placement Trainer, for hir valuable guidance and encouragement given to us throughout the project work. We would like to thank the entire IT Department faculty and Skill Academy faculties, who helped us directly and indirectly in the completion of the project. We sincerely thank our friends and family for their constant motivation during the project work.

ABSTRACT

In this fast-moving generation, the present study proposes the newer concept of predicting the prices of certain items. With an idea and motivation to help everyone we came up with a solution to get an appropriate estimate of one's car using Machine Learning Techniques which will save a lot of time and money. A car price prediction has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes is examined for the reliable and accurate prediction. The production of cars has been steadily increasing in the past decade, with over 70 million passenger cars being produced in the year 2016. This has given rise to the used car market, which on its own has become a booming industry. The recent advent of online portals has facilitated the need for both the customer and the seller to be better informed about the trends and patterns that determine the value of a used car in the market. To build a model for predicting the price of used cars in, we applied one of the machine learning techniques i.e., Linear Regression. Using linear regression, there are multiple independent variables, but one and only one dependent variable whose actual and predicted values are compared to find precision of results. Our paper proposes a system where price is dependent variable which is predicted, and this price is derived from factors like kilometers driven, car purchase year, Car Company, car model, and the fuel type.

Keywords:
Car Price Predictor, Machine Learning, Python, Linear Regression, Dependent variable, NumPy, Pandas etc.

INDEX

Introduction

One of the main areas of research in machine learning is the prediction of the price of cars. It is based on finance and the marketing domain. It is a major research topic in machine learning because the price of a car depends on many factors. Some of the factors that contribute a lot to the price of a car are:

Brand
Model
Year of Purchase
Driver
Owner Type
Engine
Transmission etc.

If one ignores the brand of the car, a car manufacturer primarily fixes the price of a car based on the features it can offer a customer. Later, the brand may raise the price depending on its goodwill, but the most important factors are what features a car gives you to add value to your life. So, in the section below, I will walk you through the task of training a car price prediction model with machine learning using the Python programming language.

Software Requirements

Software: (Anaconda/VS code)

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

Web Brouwer:
(Google Chrome/ Mozilla Firefox/ Safari Web Browser / Microsoft Edge)

A web browser takes you anywhere on the internet. It retrieves information from other parts of the web and displays it on your desktop or mobile device. The information is transferred using the Hypertext Transfer Protocol, which defines how text, images and video are transmitted on the web.

GitHub:

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

Technologies Required

NumPy:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant.

Pandas:

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Matplotlib:

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible. Create publication quality plots.

Seaborn:

Seaborn is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand your data.

Scikit-Learn:

Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling. Through scikit-learn, we can implement various machine learning models for regression, classification, clustering, and statistical tools for analyzing these models.

StreamLit:

Streamlit is an open-source Python library that makes it easy to create and share beautiful, custom web apps for machine learning and data science. In just a few minutes you can build and deploy powerful data apps.

Source Code

Importing necessary libraries.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


import warnings
warnings.simplefilter("ignore")
```

Importing the csv dataframe and Analysing the dataframe

```
dataframe = pd.read_csv("CAR DETAILS.csv")
dataframe.head()
```

Analysing the datatypes of columns of the dataframe

```
dataframe.dtypes
```

Storing a copy of a dataframe

```
df= dataframe.copy()
```

Analysing the copied dataset

```
df.head()
```

EDA (Exploratory Data Analysis)

Analysing the info of dataframe

```
dataframe.info
```

Analysing the description of dataframe

```
dataframe.describe
```

Analysing null values of dataframe

```
dataframe.isnull().sum()
```

Analysing duplicated values of dataframe

```
dataframe.duplicated().sum()
```

Source Code

Droping the duplicated values of dataframe. And analysing the dataframe

```
dataframe.drop_duplicates(inplace = True)
dataframe.duplicated().sum()
```

Analysing the columns of dataframe

```
columns=dataframe.columns
Columns
```

Analysing the unique values of the brand column of dataframe

```
dataframe['Brand'].unique()
```

Analysing the unique value of all the columns of dataframe

```
for column in columns:
    if column == "selling_price" or column == "km_driven":
        pass

    else:
        print(f"{column} \n {dataframe[column].unique()}")
        print("----------------------------------------------------------------------------------------------------")
```

Data Pre-Processing

```
from sklearn.preprocessing import LabelEncoder
```

Create an instance of the OneHotEncoder

```
label_encoder = LabelEncoder()

def label_encode(column):


    # Fit and transform the column
    dataframe[column] = label_encoder.fit_transform(dataframe[column])

    # Print the encoded DataFrame
    print(dataframe.head())
```

Source Code

Applying label encoder on all the required columns

```
label_encode("Brand")

label_encode("fuel")

label_encode("seller_type")

label_encode("transmission")

label_encode("owner")
```

Analysing theKm_driven column of dataframe.

```
dataframe["km_driven"]
```

Plotting the box plot for Km_driven column of dataframe.

```
# Plotting box plot
plt.figure(figsize=(4,4))  # Set the figure size
plt.boxplot(dataframe["km_driven"])

# Adding title and labels
plt.title('Box Plot of km_driven')
plt.xlabel('km_driven')
plt.ylabel('Distance in Kilometers')

# Display the plot
plt.show()
```

Source Code

Building user defined function for removing Outliers from a column and dataframe

```
def Outlier_free(col):
    Q3=np.quantile(dataframe[col],0.75)
    Q1=np.quantile(dataframe[col],0.25)

    IQR = Q3 - Q1

    lower_limit = Q1 - 1.5*IQR
    upper_limit = Q3 + 1.5*IQR

    suri = dataframe[((dataframe[col] >= lower_limit) & (dataframe[col] <=upper_limit ))]

    dataframe[col] = np.where(dataframe[col] >= upper_limit,
        upper_limit,
        np.where(dataframe[col] <= lower_limit,
        lower_limit,
        dataframe[col]))

    plt.figure(figsize=(4,4))
    plt.boxplot(dataframe[col])
    plt.xlabel(f"{col}")
    plt.title(f"Outlier_free_{col}")
    plt.show()
```

Removing outliers from Km_driven columns and Plotting box plot

```
Outlier_free("km_driven")
```

Splitting data into Dependent y and Independent X variables

```
X= dataframe.drop(['selling_price','name'], axis=1)
y= dataframe.selling_price

print(X.shape)
print(y.shape)
```

Analysing the independent variable columns

```
X.columns
```

Source Code

Plotting and Analyzing correlation table

```
corr= X.corr()
plt.figure(figsize=(7,7))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```

Filtering the correlation table

```
Identify columns to drop
columns_to_drop = corr.columns[((corr.abs() > 0.75) & (corr.abs() > -0.75)).any()]

columns_to_drop
```

Building a user defined function to filter

```
def correlation(dataset, threshold1, threshold2):
    corr_= set()
    corr =  dataset.corr()
    for i in range(len(corr.columns)):
        for j in range(i):
            if abs(((corr.iloc[i, j]) > threshold1) | ((corr.iloc[i, j]) < threshold2)):
                colname_i = corr.columns[i]
                corr_.add(colname_i)
    return corr_


correlation(X, 0.75, -0.75)

X.Shape

cr = X.corr()

plt.figure(figsize=(7,7))
sns.heatmap(cr, annot=True, cmap='coolwarm')
plt.show()
```

Source Code

Scaling the X variables and building Standard Scaler model

```
from sklearn.preprocessing import StandardScaler
scale = StandardScaler()
```

Fitting and transforming the data for scaling

```
X_scaled = scale.fit_transform(X)


X_scaled
```

Spliting data into train and test data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X_scaled,y,test_size=.20,random_state=4)
```

Building a user defined function for evaluationg the model test, train accuracy score

```
def evaluate_score(model):
    from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
    import math
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)

    print(f"*--------------------------------------Score_{model}------------------------------------*")
    rr= r2_score(y_test, y_pred)
    print(f"The R2_Score is :- {rr} \n")

    # Evaluate the model on the training data
    train_score = model.score(x_train, y_train)

    # Evaluate the model on the test data
    test_score = model.score(x_test, y_test)

    # Print the training and test scores
    print(f"Training Score: {train_score} \n")
    print(f"Test Score: {test_score} \n " )
```

Source Code

Applying various Machine Learning model

Building and fitting Linear Regression ML model

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Analysing the evaluated score

```
evaluate_score(lr)
```

Building and fitting Random Forest Regressor ML model

```
from sklearn.ensemble import RandomForestRegressor
rfc= RandomForestRegressor(n_estimators=300,criterion='poisson', max_depth= 10,
max_samples=0.8)
```

Analysing the evaluated score

```
evaluate_score(rfc)
```

Building and fitting Bagging Regression ML model

```
from sklearn.ensemble import BaggingRegressor
bg = BaggingRegressor(n_estimators=500, verbose= 1)
```

Analysing the evaluated score

```
evaluate_score(bg)
```

Source Code

Building and fitting Extra Tree Regressor ML model

```
from sklearn.ensemble import ExtraTreesRegressor
etr=ExtraTreesRegressor(n_estimators=5000, max_depth=5, criterion="absolute_error")
```

Analysing the evaluated score

```
evaluate_score(etr)
```

Building and fitting Polynomial RegressionML model

```
from sklearn.preprocessing import PolynomialFeatures


# Create polynomial features
poly_features = PolynomialFeatures(degree=6)
X_poly = poly_features.fit_transform(x_train)

# Perform linear regression
regressor = LinearRegression()
regressor.fit(X_poly, y_train)


from sklearn.base import RegressorMixin
X_poly_test = poly_features.transform(x_test)
# Evaluate the model on the training data
train_score = regressor.score(X_poly, y_train)

# Evaluate the model on the test data
test_score = regressor.score(X_poly_test, y_test)
```

Source Code

Analysing the evaluated score

```
print(f"Training Score: {train_score} \n")
print(f"Test Score: {test_score} \n " )
```

Building and fitting Ada Boost Regression ML model

```
from sklearn.ensemble import AdaBoostRegressor
abr = AdaBoostRegressor(n_estimators=10000, learning_rate=1.15)
```

Analysing the evaluated score

```
evaluate_score(abr)
```

Building and fitting Support Vector Regression ML model

```
from sklearn.svm import SVR

svr = SVR(kernel='rbf',
    degree=3,
    gamma='scale',
    coef0=0.0,
    tol=0.001,
    C=1.0,
    epsilon=0.1,
    shrinking=True,
    cache_size=200,
    verbose=False,
    max_iter=-1,)
```

Analysing the evaluated score

```
evaluate_score(svr)
```

Building and fitting Gradient Boosting Regression ML model

```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(n_estimators=800, learning_rate=0.2,
random_state=45,max_features='auto’)
```

Analysing the evaluated score

```
evaluate_score(gbr)
```

Building and fitting Hist Gradient Boosting Regression ML model

```
from sklearn.ensemble import HistGradientBoostingRegressor

hbr = HistGradientBoostingRegressor(l2_regularization=0.8)
```

Analysing the evaluated score

```
evaluate_score(hbr)
```

Saving the best suitable model

We will continue with Random Forest regression ML Model. And saving that model

```
import pickle

model = rfc
with open('final_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

Loading that Ml Model

```
# Assuming your pickle file is named 'model.pkl'
with open('final_model.pkl', 'rb') as f:
    model = pickle.load(f)
```

Creating 20 random point table for the further test

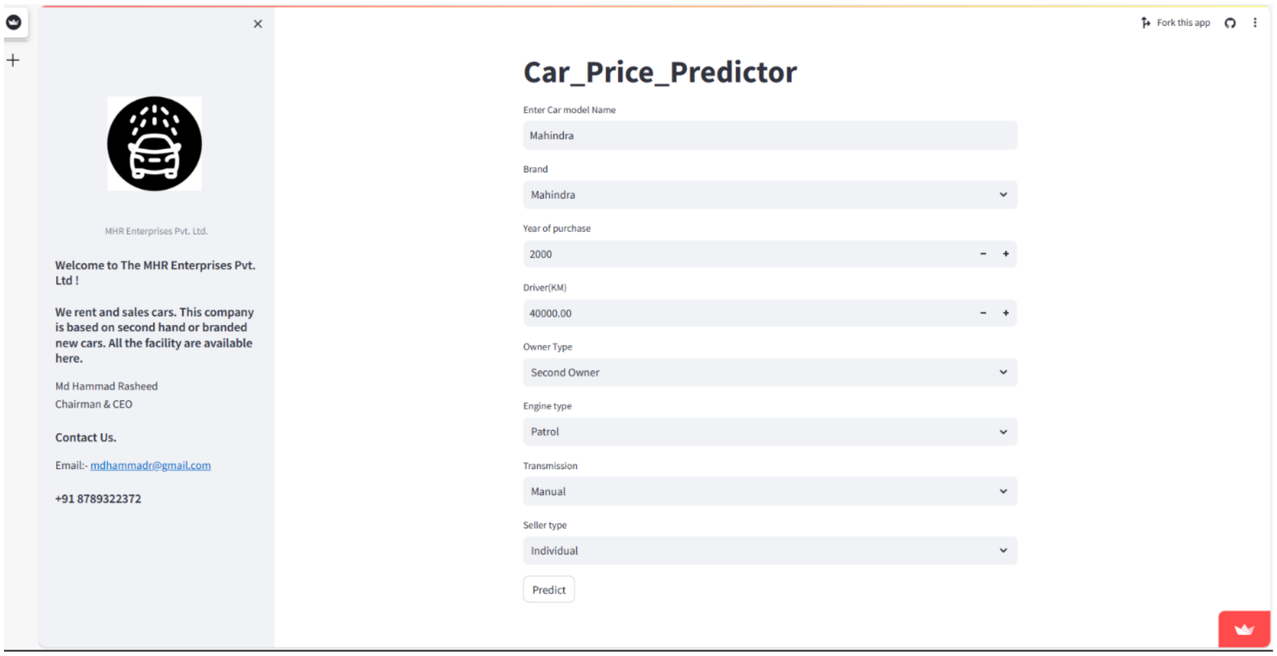
```
data = dataframe.sample(20)

Data
```

Saving the random point in pkl format

```
model1 = data
with open('data.pkl', 'wb') as f:
    pickle.dump(model1, f)
```

Car Price Predictor Interface in Streamlit Web App



MHR Enterprises Pvt. Ltd.

Welcome to The MHR Enterprises Pvt. Ltd !

We rent and sales cars. This company is based on second hand or branded new cars. All the facility are available here.

Md Hammad Rasheed
Chairman & CEO

Contact Us.

Email:- mdhammadr@gmail.com

+91 8789322372

Car_Price_Predictor

Enter Car model Name: Mahindra

Brand: Mahindra

Year of purchase: 2000

Driver(KM): 40000.00

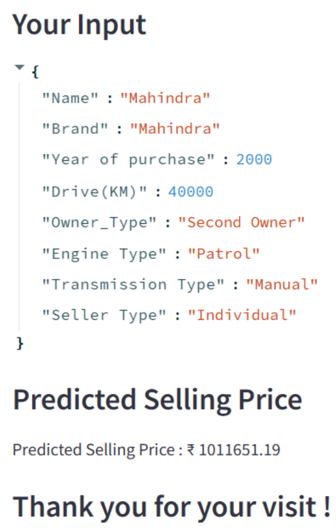
Owner Type: Second Owner

Engine type: Patrol

Transmission: Manual

Seller type: Individual

Predict



Your Input

```
{
  "Name" : "Mahindra"
  "Brand" : "Mahindra"
  "Year of purchase" : 2000
  "Drive(KM)" : 40000
  "Owner_Type" : "Second Owner"
  "Engine Type" : "Patrol"
  "Transmission Type" : "Manual"
  "Seller Type" : "Individual"
}
```

Predicted Selling Price

Predicted Selling Price : ₹ 1011651.19

Thank you for your visit !

Learning Objective/Internship Objectives

Objective and Problem Statement:
The main aim of this project is to predict the price of used cars using the various Machine Learning (ML) models. This can enable the customers to make decisions based on different inputs or factors namely

Brand
Model
Year of Purchase
Driver
Owner Type
Engine
Transmission etc.

to name a few characteristic features required by the customer. The project Car Price Prediction deals with providing the solution to these problems. Through this project, we will get to know which of the factors are significant and tell us how they affect the car’s worth in the market.

Exploratory Data Analysis (EDA)

It is a process used to analyse and summarize datasets, identifying characteristics, patterns, and relationships in the data before applying machine learning techniques.

Methods and techniques of EDA

There are several techniques and methods for performing an EDA, such as scatter plots, histograms, box plots, and descriptive statistics. The choice of techniques depends on the nature of the data and the goal of the analysis.

Data visualization

Data visualization is a powerful tool for identifying trends and patterns in datasets. Charts such as lines, bars, scatter, and box plots facilitate the identification of relationships between variables, frequency distributions, and the presence of outliers.

Descriptive statistics

Descriptive statistics summarize important information about the data, such as mean, median, mode, range, variance, and standard deviation. These measures can help identify central tendencies, dispersion, and skewness in the data.

Trends and patterns

Recognizing trends and patterns in the data is crucial for building effective machine learning models. These patterns can reveal valuable information about the data and provide insights to enhance models and improve predictions.

Data preparation and cleaning

Identifying and handling missing values

Missing values are common in datasets and can negatively impact the effectiveness of machine learning models. It is essential to identify and handle these values, either by filling them with appropriate information or removing them.

Treating categorical and numerical data

Categorical and numerical data must be treated differently during data preparation. It is necessary to transform categorical data into numerical variables using techniques such as one-hot encoding, while numerical data can be normalized or standardized.

Handling categorical data:

LabelEncoder() is an object from the sklearn.preprocessing library that converts categorical data into numerical labels.
A new DataFrame is created (data_label_encoded) as a copy of the preprocessed dataset.
The categorical column is transformed into numerical labels using the fit_transform method, and the result is saved in the new DataFrame. b. One-hot encoding:
OneHotEncoder() is an object from the sklearn.preprocessing library that creates new binary columns for each category in the original categorical column.
pd.get_dummies() is a pandas function that performs one-hot encoding on the specified columns.
The appropriate method for handling categorical data is chosen based on the specific dataset, and the result is saved as data_with_categorical_handled. In this example, one-hot encoding is used.

Handling numerical data:

a. Standard scaling:
StandardScaler() is an object from the sklearn.preprocessing library that standardizes numerical data by transforming it to have a mean of 0 and a standard deviation of 1.
A new DataFrame is created (data_standard_scaled) as a copy of the dataset with categorical data handled.
The numerical column is standardized using the fit_transform method, and the result is saved in the new DataFrame.

b. Min-max scaling:
MinMaxScaler() is an object from the sklearn.preprocessing library that scales numerical data to a specific range (default is [0, 1]).
A new DataFrame is created (data_min_max_scaled) as a copy of the dataset with categorical data handled.
The numerical column is scaled using the fit_transform method, and the result is saved in the new DataFrame.
The appropriate method for handling numerical data is chosen based on the specific dataset, and the result is saved as data_with_numerical_handled. In this example, standard scaling is used.

Final preprocessed dataset:
The final preprocessed dataset, data_final_preprocessed, is created by assigning the DataFrame with both categorical and numerical data handled. In this example, it is the DataFrame with one-hot encoding and standard scaling applied.

Note:
We already discussed in previous pages with proper code (See the Page 8-18).

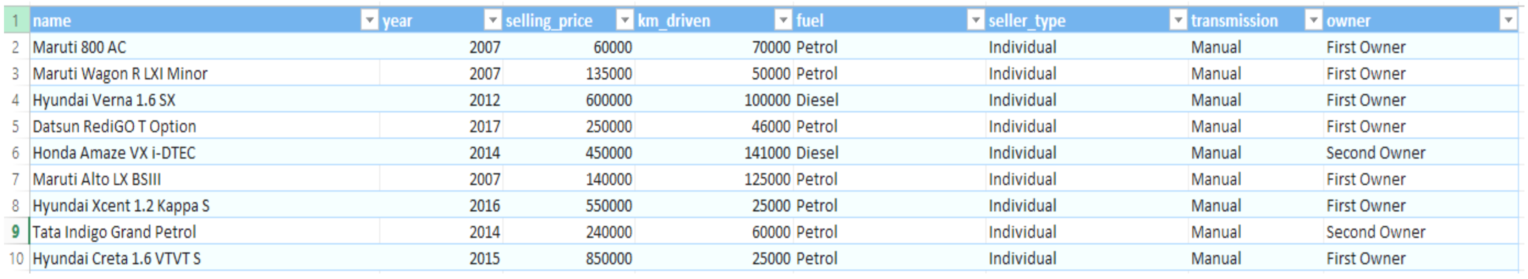
Motivation

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately[2-3]. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car prices.

Dataset

For this project, we are using the dataset on used car sales from all over the United States, available on Kaggle [1]. The features available in this dataset are Mileage, VIN, Make, Model, Year, State and City.

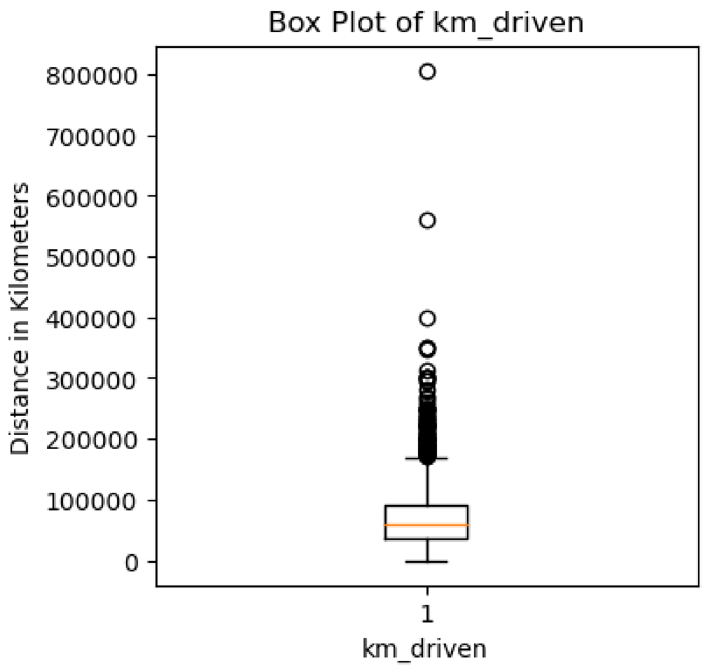
Dataset Preview



name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner
Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner
Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner
Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner
Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner

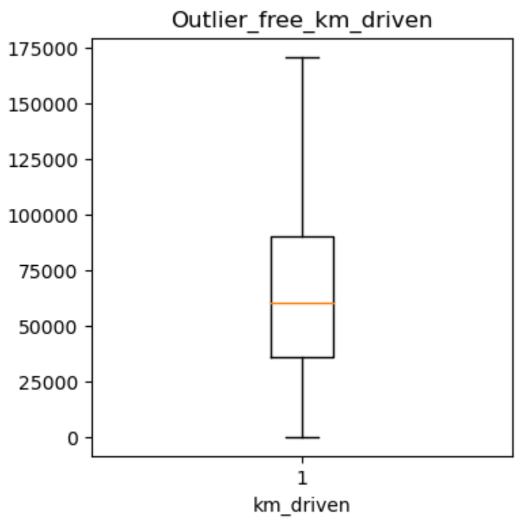
Plotting the box plot for Km_driven column of dataframe:

Box plots provide a quick visual summary of the variability of values in a dataset. They show the median, upper and lower quartiles, minimum and maximum values, and any outliers in the dataset. Outliers can reveal mistakes or unusual occurrences in data.



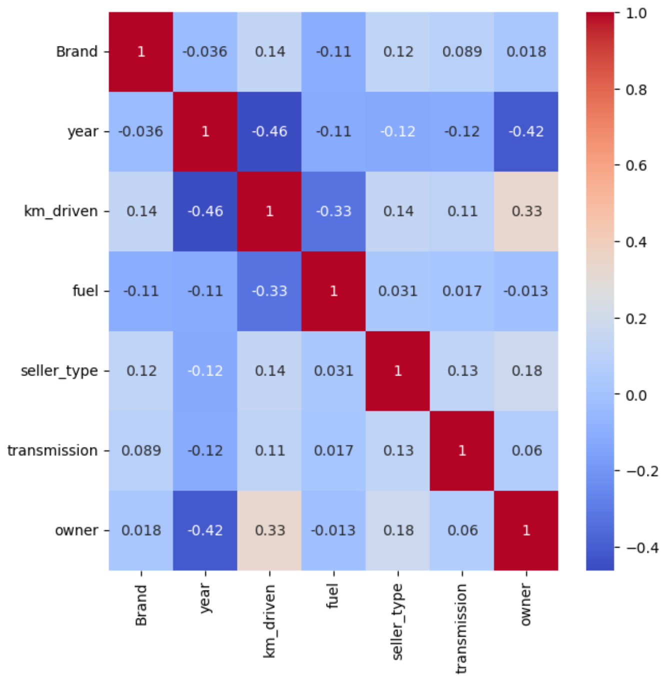
Removing outliers from Km_driven columns and Plotting box plot:

an outlier is an extremely high or extremely low data point relative to the nearest data point and the rest of the neighboring co-existing values in a data graph or dataset you're working with. Outliers are extreme values that stand out greatly from the overall pattern of values in a dataset or graph.



Plotting and Analyzing correlation table:

A correlation matrix is simply a table which displays the correlation coefficients for different variables. The matrix depicts the correlation between all the possible pairs of values in a table. It is a powerful tool to summarize a large dataset and to identify and visualize patterns in the given data.



	Brand	year	km_driven	fuel	seller_type	transmission	owner
Brand	1	-0.036	0.14	-0.11	0.12	0.089	0.018
year	-0.036	1	-0.46	-0.11	-0.12	-0.12	-0.42
km_driven	0.14	-0.46	1	-0.33	0.14	0.11	0.33
fuel	-0.11	-0.11	-0.33	1	0.031	0.017	-0.013
seller_type	0.12	-0.12	0.14	0.031	1	0.13	0.18
transmission	0.089	-0.12	0.11	0.017	0.13	1	0.06
owner	0.018	-0.42	0.33	-0.013	0.18	0.06	1

Methodology

We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a 90% - 10% split for the training and test data. To reduce the time required for training, we used 500 thousand examples from our dataset. Linear Regression, Random Forest and Gradient Boost were our baseline methods. For most of the model implementations, the open-source Scikit-Learn package [7] was used.

1. Linear Regression Model

Linear Regression was chosen as the first model due to its simplicity and comparatively small training time. The features, without any feature mapping, were used directly as the feature vectors. No regularization was used since the results clearly showed low variance.

2. Random Forest Model

Random Forest is an ensemble learning based regression model. It uses a model called decision tree, specifically as the name suggests, multiple decision trees to generate the ensemble model which collectively produces a prediction. The benefit of this model is that the trees are produced in parallel and are relatively uncorrelated, thus producing good results as each tree is not prone to individual errors of other trees. This uncorrelated behavior is partly ensured by the use of Bootstrap Aggregation or bagging providing the randomness required to produce robust and uncorrelated trees. This model was hence chosen to account for the large number of features in the dataset and compare a bagging technique with the following gradient boosting methods.

3. Bagging Regression Model

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once.

4. Tree Regression Model

Tree-based regression models are known for their simplicity and efficiency when dealing with domains with large number of variables and cases. Regression trees are obtained using a fast divide and conquer greedy algorithm that recursively partitions the given training data into smaller subsets.

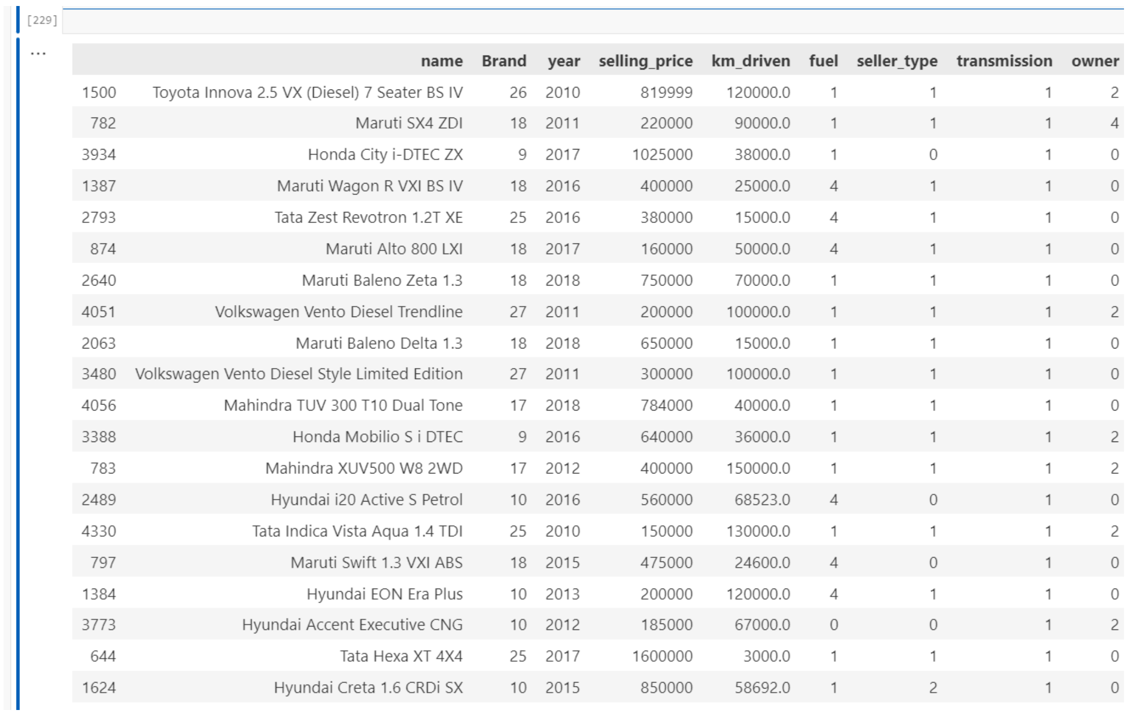
5. Polynomial Regression Model.

Polynomial regression is a kind of linear regression in which the relationship shared between the dependent and independent variables Y and X is modeled as the nth degree of the polynomial. This is done to look for the best way of drawing a line using data points.

6. SVM

What is SVM model in machine learning?
A support vector machine (SVM) is a type of supervised learning algorithm used in machine learning to solve classification and regression tasks; SVMs are particularly good at solving binary classification problems, which require classifying the elements of a data set into two groups.

Creating 20 random point table for the further test



[229]

	name	Brand	year	selling_price	km_driven	fuel	seller_type	transmission	owner
1500	Toyota Innova 2.5 VX (Diesel) 7 Seater BS IV	26	2010	819999	120000.0	1	1	1	2
782	Maruti SX4 ZDI	18	2011	220000	90000.0	1	1	1	4
3934	Honda City i-DTEC ZX	9	2017	1025000	38000.0	1	0	1	0
1387	Maruti Wagon R VXI BS IV	18	2016	400000	25000.0	4	1	1	0
2793	Tata Zest Revotron 1.2T XE	25	2016	380000	15000.0	4	1	1	0
874	Maruti Alto 800 LXI	18	2017	160000	50000.0	4	1	1	0
2640	Maruti Baleno Zeta 1.3	18	2018	750000	70000.0	1	1	1	0
4051	Volkswagen Vento Diesel Trendline	27	2011	200000	100000.0	1	1	1	2
2063	Maruti Baleno Delta 1.3	18	2018	650000	15000.0	1	1	1	0
3480	Volkswagen Vento Diesel Style Limited Edition	27	2011	300000	100000.0	1	1	1	0
4056	Mahindra TUV 300 T10 Dual Tone	17	2018	784000	40000.0	1	1	1	0
3388	Honda Mobilio S i DTEC	9	2016	640000	36000.0	1	1	1	2
783	Mahindra XUV500 W8 2WD	17	2012	400000	150000.0	1	1	1	2
2489	Hyundai i20 Active S Petrol	10	2016	560000	68523.0	4	0	1	0
4330	Tata Indica Vista Aqua 1.4 TDI	25	2010	150000	130000.0	1	1	1	2
797	Maruti Swift 1.3 VXI ABS	18	2015	475000	24600.0	4	0	1	0
1384	Hyundai EON Era Plus	10	2013	200000	120000.0	4	1	1	0
3773	Hyundai Accent Executive CNG	10	2012	185000	67000.0	0	0	1	2
644	Tata Hexa XT 4X4	25	2017	1600000	3000.0	1	1	1	0
1624	Hyundai Creta 1.6 CRDi SX	10	2015	850000	58692.0	1	2	1	0

Results
The results of our tests were quantified in terms of the R^2 score of our predictions. score is a statistical R^2 measure of how close the data are to the fitted regression line.

Learning Algorithm	R^2 Score on Test Data	R^2 Score on Training Data	Training Time
Linear Regression	0.87	0.87	15 minutes
Gradient Boost	0.64	0.64	130 minutes
Random Forest	0.88	0.98	75 minutes
Light GBM	0.81	0.82	104 seconds
XGBoost	0.78	0.81	180 minutes
KMeans + LinReg	0.88	0.89	70 minutes
Deep Neural Network	0.85	0.85	10 hours

Compared to Linear Regression, most Decision-Tree based methods did not perform comparably well. This can be attributed to the apparent linearity of the dataset. We believe that It can also be attributed to the difficulty in tuning the hyperparameters for most gradient boost methods. The exception to this is the Random Forest method which marginally outperforms Linear Regression. However Random Forests tend to overfit the dataset due to the tendency of growing longer trees. This was worked upon by restricting the depth of trees to different values and it was observed that beyond limiting depth to 36 resulted in negligible improvement in prediction performance but progressively increased overfitting. As expected lightGBM performed marginally better than XGBoost but had a significantly faster training time. Building up from the relatively good performance of Linear Regression, the KMeans + Linear Regression Ensemble Learning Method (with K = 3) produced the best R score on test data without high variance as 2 it fits linear relationships categorically. The deep neural network was converging to local minima due to small batch-sizes.

Contributions

Both the team members contributed equally towards this project.
Link to project repository : Click Here
Link: (https://github.com/mhrofficial/Car_Price_Predictor.git)

References

1. https://www.kaggle.com/jpayne/852k-used-car-listings
2. N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used car by using regression models," *2018 5th International Conference on Business and Industrial Research (ICBIR)*, Bangkok, 2018, pp. 115-119.
3. Listiani M. 2009. Support Vector Regression Analysis for Price Prediction in a Car Leasing Application. Master Thesis. Hamburg University of Technology
4. Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016.
5. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." *Advances in Neural Information Processing Systems*. 2017.
6. Fisher, Walter D. "On grouping for maximum homogeneity." *Journal of the American statistical Association* 53.284 (1958): 789-798.
7. https://scikit-learn.org/stable/modules/classes.html: Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.