Minor Project Report

on

[Price prediction of used Cars]

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

in partial fulfilment of the requirements for the award of degree of

Master of Computer Applications

LOVELY FACULTY OF TECHNOLOGY & SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

[2023-2024]

Submitted By :	**Supervised By :**
Sachin Thakur (12215076)	Dr. Tarandeep Singh Walia
Vivek Kumar Gupta (12215379)	(Dept. of Computer Application)
Intiyaz Ali (12215366)	

Table of Content

1. Introduction about project

Introduction to Project : Price predicition of used Cars.

The "Used Car Price Prediction" project is a data-driven initiative designed to provide an accurate and reliable estimation of the market value of preowned automobile. This project will allow users to :

1. Data Entry Form:
    - **Make and Model:** The car's make (brand) and model.
    - **Year:** The manufacturing year of the car.
    - **Mileage:** The number of miles or kilometers the car has been driven.
    - **Fuel Type:** The type of fuel the car uses (e.g., gasoline, diesel, electric).
    - **Transmission Type:** Whether the car has an automatic or manual transmission.
    - **Number of Owners:** The number of previous owners.
    - **Condition:** The overall condition of the car (e.g., excellent, good, fair).
    - **Location:** The geographical location where the car is being sold.
    - **Additional Features:** Any additional features or upgrades the car has (e.g., sunroof, leather seats).

2. **Data Submission:**
   After entering the necessary information, the user submits the form, and the data is sent to the backend of the application for processing.

3. **Backend Processing:**
   The backend of the application, which typically involves machine learning models and algorithms, processes the user's input data. It uses historical data and pricing models to generate a price estimate for the used car.

4. **Price Prediction:**
   The predicted price is calculated, taking into account various factors and variables. Machine learning algorithms may analyze how different features impact a car's price based on the historical data and correlations.

5. **User Display:**
   The estimated price is then displayed to the user on the same web page or within the app, providing an approximate value for the used car they are interested in.

Benefits of this Project :

Certainly, here's a more concise list of the benefits of the "Price Prediction of Used Cars" project:

- Informed Buying and Selling: Helps consumers make informed decisions when buying or selling used cars.
- Price Transparency: Promotes transparency by showing how various factors affect car prices.
- Fair Pricing: Facilitates fair pricing in the used car market for both buyers and sellers.
- Time and Effort Savings: Saves time and effort by automating the price estimation process.
- Reduced Information Asymmetry: Reduces information gaps between buyers and sellers.
- Market Insights: Provides insights into market trends for dealerships and the automotive industry.
- Enhanced User Experience: Offers a user-friendly tool for estimating car prices.
- User Feedback and Improvement: Uses user feedback to improve price predictions over time.

Target Audience

The target audience for a "Price Prediction of Used Cars" project includes a diverse group of individuals and businesses involved in the buying, selling, or analysis of used cars.

Conlusion

In conclusion, this project offers a valuable solution to a diverse set of users within the used car market and related industries. This project leverages data and machine learning to provide accurate and transparent price estimates for used cars, benefiting both buyers and sellers. The key advantages include informed decision-making, pricing transparency, and fair transactions, while also saving time and reducing information gaps.

2. Project Modules and its description

The project modules for the Price prediction of used Cars and their descriptions:

1. **Data Collection Module:**

This module is responsible for gathering data about used cars. The data may come from various sources, such as online car listings, classified ads, or databases of historical car sales. It collects details about car make, model, year, mileage, features, location, and other relevant factors.

2. Data Preprocessing Module:

Raw data collected from various sources often requires cleaning and preprocessing. This module cleans the data by handling missing values, outliers, and inconsistencies. It may also encode categorical variables and normalize or scale numerical features.

3. Feature Engineering Module:

In feature engineering, relevant features or variables are selected and transformed to improve the model's ability to predict car prices. For example, you might create new features like "age of the car" or "average market price of cars of the same make and model."

4. Machine Learning Model Module:

This module involves the development and training of machine learning models for price prediction. Common models used for regression tasks in this project include linear regression, decision trees, random forests, support vector regression, or more advanced techniques like gradient boosting and neural networks. Multiple models may be trained and evaluated to select the best performing one.

5. Evaluation and Validation Module:

To ensure the model's accuracy, this module validates the performance of the machine learning models. It uses techniques like cross-validation, and metrics such as mean squared error (MSE) or mean absolute error (MAE) to assess the model's ability to predict car prices.

6. User Interface Module:

This module provides the user interface for data input and price estimation. It's typically implemented as a web application or mobile app. Users enter details about the used car, and the estimated price is displayed. The module communicates with the backend for processing.

7. Backend Processing Module:

The backend processes user input, applies the machine learning model to generate a price estimate, and returns the result to the user interface. It may also include functions for data retrieval, model loading, and result presentation.

8. User Feedback Module:

This optional module gathers user feedback and ratings on the accuracy of the price estimates. The feedback can be used for model refinement and improvements over time.

9. Deployment Module:

Once the model is trained and the system is developed, it needs to be deployed to a server or cloud platform, making it accessible to users.

10. Database Module:

A database is used to store historical car data, pricing models, and user feedback. It helps in data retrieval and model updates.

3. Coding

**Packages to load in **

Import Data and Required Packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
from six.moves import urllib

warnings.filterwarnings("ignore")

%matplotlib inline
```

Import Data and Required Packages

```
df_automobile = pd.read_csv('cardekho_dataset.csv', index_col=[0])
df_automobile
df_automobile.head()
df_automobile.tail()
```

Get Information About Our Dataset Like the Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

```
print('The Column Name, Record Count and Data Types are as follows: ')
df_automobile.info()
```

Admin Access

```


df_automobile = pd.read_csv("cardekho_dataset.csv")

# Function to add data to the DataFrame
def add_data(dataframe):
    new_data = {}
    for col in dataframe.columns:
        new_data[col] = [input(f"Enter the value for {col}: ")]
    dataframe = pd.concat([dataframe, pd.DataFrame(new_data)],
ignore_index=True)
    return dataframe

# Function to update data in the DataFrame
def update_data(dataframe):
    index_to_update = int(input("Enter the index of the entry to
update: "))
    column_to_update = input("Enter the column to update: ")
    new_value = input("Enter the new value: ")
    dataframe.at[index_to_update, column_to_update] = new_value
    return dataframe

# Function to delete data from the DataFrame
def delete_data(dataframe):
    index_to_delete = int(input("Enter the index of the entry to
delete: "))
    dataframe = dataframe.drop(index_to_delete)
    dataframe = dataframe.reset_index(drop=True)
    return dataframe

while True:
    choice = input("Enter 'A' to add data, 'U' to update data, 'D' to
delete data, or 'Q' to quit: ").upper()

    if choice == 'A':
        df_automobile = add_data(df_automobile)
    elif choice == 'U':
        df_automobile = update_data(df_automobile)
    elif choice == 'D':
        df_automobile = delete_data(df_automobile)
    elif choice == 'Q':
        break
    else:
        print("Invalid choice. Please try again.")

print(df_automobile)
```

Data Cleaning

```
df_clean = df_automobile.copy()
df_clean.head()
df_clean.drop('Unnamed: 0' , axis =1 , inplace = True )
```

```
df=df_clean.replace('?',np.NAN)
df.isnull().sum()
df
```

```
print('Missing Value Presence in different columns of DataFrame are as 
follows : ')
print('-'*100)
total=df.isnull().sum().sort_values(ascending=False)
percent=(df.isnull().sum()/df.isnull().count()*100).sort_values(ascendi
ng=False)
pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
```

We drop the missing values of Engine because they are less than 1% of the total data.

```
df.dropna(axis=0,inplace=True)
df
```

```
df['engine'] = df['engine'].apply(pd.to_numeric)
```

```
# Defining numerical & categorical columns
numeric_features = [feature for feature in df.columns if 
df[feature].dtype != 'O']
categorical_features = [feature for feature in df.columns if 
df[feature].dtype == 'O']

# print columns
print('We have {} numerical features : 
{}'.format(len(numeric_features), numeric_features))
print('\nWe have {} categorical features : 
{}'.format(len(categorical_features), categorical_features))
```

```
df.to_csv('cleandata.csv', index=False)
df
```

```
print('Summary Statistics of numerical features for DataFrame are as
follows:')
print('-'*100)
df.describe()
```

Summary Statistics of numerical features for DataFrame are as follows:

```
print('Summary Statistics of categorical features for DataFrame are as
follows:')
print('-'*100)
df.describe(include= 'object')
```

Summary Statistics of categorical features for DataFrame are as follows:

Exploratory Data Analysis

Numerical Features

```
plt.figure(figsize=(15, 15))
plt.suptitle('Univariate Analysis of Numerical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.0)

for i in range(0, len(numeric_features)):
    plt.subplot(3, 3, i+1)
    sns.kdeplot(x=df[numeric_features[i]],shade=True, color='b')
    plt.xlabel(numeric_features[i])
    plt.tight_layout()
```

```
plt.figure(figsize=(15, 15))
plt.suptitle('Box Plot of Numerical Features', fontsize=20,
fontweight='bold', alpha=0.8, y=1.0)

for i in range(0, len(numeric_features)):
    plt.subplot(3, 3, i+1)
    sns.boxplot(x=df[numeric_features[i]])
    plt.xlabel(numeric_features[i])
    plt.tight_layout()
```

Report

- Km_driven, max_power, selling_price, and engine are right skewed and postively skewed.
- Outliers in km_driven, engine, selling_price, and max power.

Catrgorical Features

```
plt.figure(figsize=(20, 15))
plt.suptitle('Univariate Analysis of Categorical Features',
fontsize=20, fontweight='bold', alpha=0.8, y=1.0)

cat1 = [ 'brand', 'seller_type', 'fuel_type', 'transmission_type']
for i in range(0, len(cat1)):
    plt.subplot(2, 2, i+1)
    sns.countplot(x=df[cat1[i]])
    plt.xlabel(cat1[i])
    plt.xticks(rotation=45)
    plt.tight_layout()
```

Bivariate Analysis

```
continuous_features=[feature for feature in numeric_features if
len(df[feature].unique())>=10]
print('Num of continuos features:',continuous_features)
```

Num of continuos features: ['vehicle_age', 'km_driven', 'mileage', 'engine', 'max_power', 'selling_pri ce']

```
fig = plt.figure(figsize=(15, 20))

for i in range(0, len(continuous_features)):
    ax = plt.subplot(8, 2, i+1)

    sns.scatterplot(data= df ,x='selling_price',
y=continuous_features[i], color='b')
    plt.xlim(0,25000000) # Limit to 25 lakhs Rupees to view clean
    plt.tight_layout()
```

Result

- **Lower Vehicle age has more selling price than Vehicle with more age.**
- **Engine CC has positive effect on price,Vehicle with 2000 cc and below are mostly priced below 5lakh.**
- **Kms Driven has negative effect on selling price.**

Multivariate Analysis

Check Multicollinearity in Numerical features

```
df[numeric_features].corr()
```

```
plt.figure(figsize=(10,5))
sns.heatmap(data = df[numeric_features].corr(), annot= True, cmap=
'plasma', vmin= -1 , vmax= 1, linecolor='white', linewidths=2)
plt.show()
```

Data Visualization

Most Sold Cars

```
print('Top 10 Sold Cars on CarDekho Website')


df.car_name.value_counts()[0:10]
```

```
plt.subplots(figsize=(10,5))
sns.countplot(x="car_name", data=df,ec = "black",palette="Set1",order =
df['car_name'].value_counts().index)
plt.title("Top 10 Most Sold Car", weight="bold",fontsize=20, pad=20)
plt.ylabel("Count", weight="bold", fontsize=20)
plt.xlabel("Car Name", weight="bold", fontsize=16)
plt.xticks(rotation= 45)
plt.xlim(-1,10.5)
plt.show()
```

- Check mean price of Hyundai i20 which is most sold

```
i20 = df[df['car_name'] == 'Hyundai i20']['selling_price'].mean()
print(f'The mean price of Hyundai i20 is {i20:.2f} Rupees')
```

The mean price of Hyundai i20 is 544015.49 Rupees

Result

- As per the Chart these are top 10 most selling cars in used car website.
- Of the total cars sold Hyundai i20 shares 5.8% of total ads posted and followed by Maruti Swift Dzire.
- Mean Price of Most Sold Car is 5.4 lakhs.
- This Feature has impact on the Target Variable.

Most Sold Brand

```
print('Top 10 Most Sold Car Brand')


df.brand.value_counts()[0:10]
```

```
plt.subplots(figsize=(10,5))
sns.countplot(x="brand", data=df,ec = "black",palette="Set2",order =
df['brand'].value_counts().index)
plt.title("Top 10 Most Sold Brand", weight="bold",fontsize=20, pad=20)
plt.ylabel("Count", weight="bold", fontsize=14)
plt.xlabel("Brand", weight="bold", fontsize=16)
plt.xticks(rotation= 45)
plt.xlim(-1,10.5)
plt.show()
```

Check the Mean price of Maruti brand which is most sold

```
maruti = df[df['brand'] == 'Maruti']['selling_price'].mean()
print(f'The mean price of Maruti is {maruti:.2f} Rupees')
```

The mean price of Maruti is 486990.15 Rupees

Result

- As per the Chart Maruti has the most share of Ads in Used car website and Maruti is the most sold brand.
- Following Maruti we have Hyundai and Honda.
- Mean Price of Maruti Brand is 4.8 lakhs.

Costlier Brand

```
brand = df.groupby('brand').selling_price.max()
brand =
brand.to_frame().sort_values('selling_price',ascending=False)[0:10]

print('-'*50)
print('Top 10 Costlier Brands on CarDekho Website')
print('-'*50)

brand
```

```
plt.subplots(figsize=(10,5))
sns.barplot(x=brand.index, y=brand.selling_price,ec =
"black",palette="Set2")
plt.title("Brand vs Highest Selling Price", weight="bold",fontsize=20,
pad=20)
plt.ylabel("Selling Price", weight="bold", fontsize=15)
plt.xlabel("Brand Name", weight="bold", fontsize=16)
plt.xticks(rotation=90)
plt.show()
```

Result

- Costliest Brand sold is Ferrari at 3.95 Crores.
- Second most costliest car Brand is Rolls-Royce as 2.42 Crores.
- Brand name has very clear impact on selling price.

Most Mileage Car Brand

```
mileage=
df.groupby('brand')['mileage'].mean().sort_values(ascending=False).head
(15)

print('-'*50)
print('Most Mileage Car Brand on CarDekho Website')
print('-'*50)

mileage.to_frame()
```

```
plt.subplots(figsize=(10,5))
sns.barplot(x=mileage.index, y=mileage.values, ec = "black",
palette="Set2")
plt.title("Brand vs Mileage", weight="bold",fontsize=20, pad=20)
plt.ylabel("Mileage in Kmpl", weight="bold", fontsize=15)
plt.xlabel("Brand Name", weight="bold", fontsize=12)
plt.ylim(0,25)
plt.xticks(rotation=45)
plt.show()
```

Most Mileage Car

```
mileage_Car=
df.groupby('car_name')['mileage'].mean().sort_values(ascending=False).h
ead(10)

print('-'*50)
print('Most Mileage Car on CarDekho Website')
print('-'*50)

mileage_Car.to_frame()
```

```
plt.subplots(figsize=(10,5))
sns.barplot(x=mileage_Car.index, y=mileage_Car.values, ec = "black",
palette="Set1")
plt.title("Car Name vs Mileage", weight="bold",fontsize=20, pad=20)
plt.ylabel("Mileage in Kmpl", weight="bold", fontsize=15)
plt.xlabel("Car Name", weight="bold", fontsize=12)
plt.ylim(0,27)
plt.xticks(rotation=45)
plt.show()
```

Kilometers Driven Vs Selling Price

```
plt.subplots(figsize=(10,5))
sns.scatterplot(x="km_driven", y='selling_price', data=df,ec =
"white",color='b', hue='fuel_type')
plt.title("Kilometer Driven vs Selling Price",
weight="bold",fontsize=20, pad=20)
plt.ylabel("Selling Price", weight="bold", fontsize=20)
plt.xlim(-10000,800000) #used limit for better visualization
plt.ylim(-10000,10000000)
plt.xlabel("Kilometer driven", weight="bold", fontsize=16)
plt.show()
```

Result
Many Cars were sold with kms between 0 to 20k Kilometers

Low Kms driven cars had more selling price compared to cars which had more kms driven.

Vehicle Age vs Selling Price

```
plt.subplots(figsize=(15,5))
sns.lineplot(x='vehicle_age',y='selling_price',data=df,color='b')
plt.ylim(0,2500000)
plt.show()
```

Result

- As the Vehicle age increases the price also get reduced.
- Vehicle age has Negative impact on selling price

Vehicle Age vs Mileage

```
vehicle_age =
df.groupby('vehicle_age')['mileage'].median().sort_values(ascending=Fal
se)
vehicle_age.to_frame().head(5)
```

```
plt.subplots(figsize=(10,5))
sns.boxplot(x=df.vehicle_age, y= df.mileage, palette="Set1")
plt.title("Vehicle Age vs Mileage", weight="bold",fontsize=20, pad=20)
plt.ylabel("Mileage", weight="bold", fontsize=20)
plt.xlabel("Vehicle Age in Years", weight="bold", fontsize=16)
plt.show()
```

Result

- As the Age of vehicle increases the median of mileage drops.
- Newer Vehicles have more mileage median older vehicle.

```
oldest =
df.groupby('car_name')['vehicle_age'].max().sort_values(ascending=False
).head(10)
oldest.to_frame()
```

Result

- Maruti Alto is the Oldest car available 29 years old in the used car website followed by BMW 3 for 25 years old.

Fuel Type Vs Selling Price

```
fuel =
df.groupby('fuel_type')['selling_price'].median().sort_values(ascending
=False)
fuel.to_frame()
```

```
plt.subplots(figsize=(10,5))
sns.barplot(x=fuel.index, y=fuel.values, ec = "black",
palette="Set2_r")
plt.title("Fuel type vs Selling Price", weight="bold",fontsize=20,
pad=20)
plt.ylabel("Median Selling Price", weight="bold", fontsize=15)
plt.xlabel("Fuel Type", weight="bold", fontsize=10)
plt.show()
```

Result

- Electric cars have highers selling average price.
- Followed by Diesel and Petrol.
- Fuel Type is also important feature for the Target variable.

Most Sold Fuel type

```
plt.subplots(figsize=(10,5))
sns.countplot(x=df.fuel_type, ec = "black", palette="Set2_r")
plt.title("Fuel Type Count", weight="bold",fontsize=20, pad=20)
plt.ylabel("Count", weight="bold", fontsize=15)
plt.xlabel("Fuel Type", weight="bold", fontsize=12)
plt.show()
```

Result

- Petrol and Diesel dominate the used car market in the website.
- The most sold fuel type Vechicle is Petrol.
- Followed by diesel and CNG and least sold is Electric

Feature Engineering

```
df.head()
```

Removing unnecessary features

Dropping car_name ,brand and model

These features are not directly correlated with the price of car and they can actually introduce noise into the model. For example, two cars with the same features but different brands may have different prices. This is because brand reputation and perceived quality can play a role in determining the price of a car. By dropping the car_name ,brand and model, we can create a model that is more accurate and reliable.

```
df_model=df.copy()

df_model
```

```
df_model.drop(labels=['car_name','brand','model'],axis=1,inplace=True)

df_model
```

Converting Categorical Columns into numerical / Encoding the Categorical Columns

```
df_model['fuel_type'].unique()
```

array(['Petrol', 'Diesel', 'CNG', 'LPG', 'Electric'], dtype=object)

```
df_model['fuel_type'] = 
df_model['fuel_type'].map({'Petrol':0,'Diesel':1,'CNG':2,'LPG':3,'Elect
ric':4})
```

```
df_model['fuel_type'].unique()
```

array([0, 1, 2, 3, 4])

```
df_model['seller_type'].unique()
```

array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)

```
df_model['seller_type'] =
df_model['seller_type'].map({'Dealer':0,'Individual':1,'Trustmark
Dealer':2})
```

```
df_model['seller_type'].unique()
```

array([1, 0, 2])

```
df_model['transmission_type'].unique()
```

array(['Manual', 'Automatic'], dtype=object)

```
df_model['transmission_type']
=df_model['transmission_type'].map({'Manual':0,'Automatic':1})
```

```
df_model['transmission_type'].unique()
```

array([0, 1])

```
df_model.dtypes
```

Store Feature Matrix In X and Response(Target) In Vector y

```
X=df_model.drop('selling_price',axis=1)       # Feature Matrix
X
```

```
y=df_model['selling_price']     # Target Variable
y
```

Splitting The Dataset Into The Training Set And Test Set

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,rando
m_state=42)
```

Import The models

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
```

Model Training

```
lr = LinearRegression()
lr.fit(X_train,y_train)

rf = RandomForestRegressor()
rf.fit(X_train,y_train)
```

Prediction on Test Data

```
y_pred1 = lr.predict(X_test)
y_pred2 = rf.predict(X_test)
print(y_pred1)
print(y_pred2)
```

Evaluating the Algorithm

```
from sklearn import metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
```

```
score1 = metrics.r2_score(y_test,y_pred1)
score2 = metrics.r2_score(y_test,y_pred2)
```

```
print(score1,score2)
```

```
final_data = pd.DataFrame({'Models':['LR','RF'],
             "R2_SCORE":[score1,score2]})
final_data
```

```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming 'final_data' is your DataFrame

sns.barplot(x='Models', y='R2_SCORE', data=final_data)
plt.xticks(rotation=90)  # Rotate x-axis labels for better readability
plt.show()
```

Save The Model

```
xg_final = rf.fit(X,y)
xg_final
```

```
import joblib
joblib.dump(xg_final,'car_price_predictor')
```

```
model = joblib.load('car_price_predictor')
```

Prediction on New Data

```
import pandas as pd

data_new = pd.DataFrame({
    'vehicle_age': 6,
    'km_driven': 25000,
    'seller_type': 0,
    'fuel_type': 1,
    'transmission_type': 1,
    'mileage': 15,
    'engine': 1500,
    'max_power': 45,
    'seats': 5,
}, index=[0])

# Print the created DataFrame
print(data_new)
```

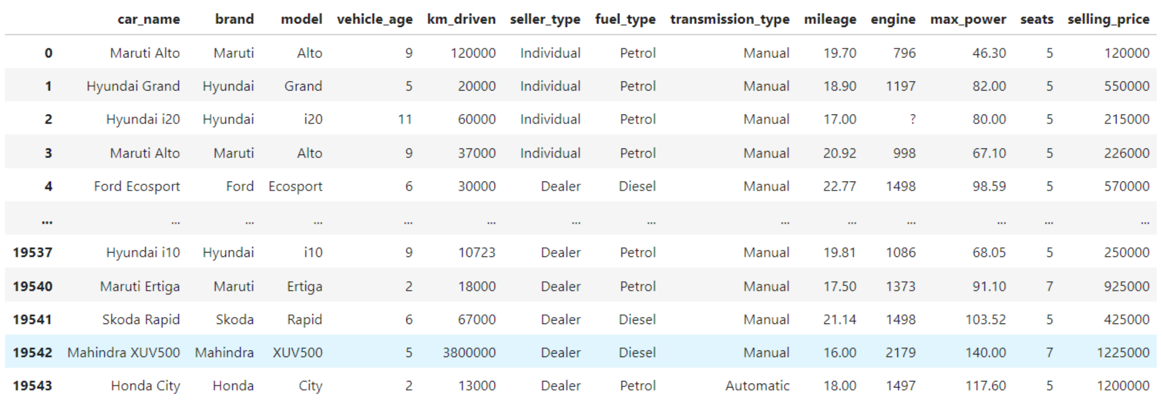
```
model.predict(data_new)
```

GUI

```
from tkinter import *
import joblib

def show_entry_fields():
    p1 = float(e1.get())
    p2 = float(e2.get())
    p3 = float(e3.get())
    p4 = float(e4.get())
    p5 = float(e5.get())
    p6 = float(e6.get())
    p7 = float(e7.get())
    p8 = float(e8.get())
    p9 = float(e9.get())

    model = joblib.load('car_price_predictor')
    data_new = pd.DataFrame({
        'vehicle_age': [p1],
        'km_driven': [p2],
        'seller_type': [p3],
        'fuel_type': [p4],
        'transmission_type': [p5],
        'mileage': [p6],
        'engine': [p7],
        'max_power': [p8],
        'seats': [p9],
    })
    result = model.predict(data_new)
    result_label.config(text="Car Purchase amount: " + str(result[0]))
    print("Car Purchase amount", result[0])

def clear_fields():
    e1.delete(0, 'end')
    e2.delete(0, 'end')
    e3.delete(0, 'end')
    e4.delete(0, 'end')
    e5.delete(0, 'end')
    e6.delete(0, 'end')
    e7.delete(0, 'end')
    e8.delete(0, 'end')
    e9.delete(0, 'end')
    result_label.config(text="")

master = Tk()
master.title("Car Price Prediction Using Machine Learning")
label = Label(master, text="Car Price Prediction Using Machine
Learning", bg="black", fg="white")
label.grid(row=0, columnspan=2)

Label(master, text="vehicle_age").grid(row=1)
Label(master, text="Kms_Driven").grid(row=2)
Label(master, text="seller_type").grid(row=3)
Label(master, text="fuel_type").grid(row=4)
Label(master, text="Transmission").grid(row=5)
Label(master, text="mileage").grid(row=6)
Label(master, text="engine").grid(row=7)
Label(master, text="max_power").grid(row=8)
Label(master, text="seats").grid(row=9)

e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)

e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)
e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)

Button(master, text='Predict', command=show_entry_fields).grid(row=10,
column=0)
Button(master, text='Clear', command=clear_fields).grid(row=10,
column=1)

result_label = Label(master, text="")
result_label.grid(row=11, columnspan=2)

mainloop()
```

4. Screenshot Output

Data Loading :

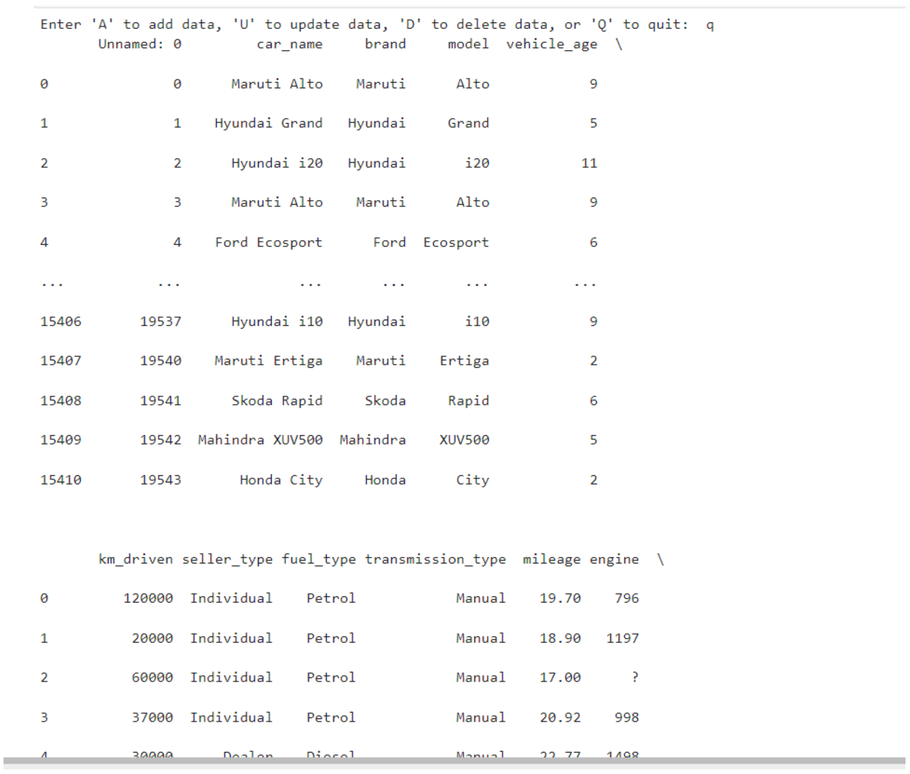


	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type	transmission_type	mileage	engine	max_power	seats	selling_price
0	Maruti Alto	Maruti	Alto	9	120000	Individual	Petrol	Manual	19.70	796	46.30	5	120000
1	Hyundai Grand	Hyundai	Grand	5	20000	Individual	Petrol	Manual	18.90	1197	82.00	5	550000
2	Hyundai i20	Hyundai	i20	11	60000	Individual	Petrol	Manual	17.00	?	80.00	5	215000
3	Maruti Alto	Maruti	Alto	9	37000	Individual	Petrol	Manual	20.92	998	67.10	5	226000
4	Ford Ecosport	Ford	Ecosport	6	30000	Dealer	Diesel	Manual	22.77	1498	98.59	5	570000
...	...	...	...	...	...	...	...	...	...	...	...	...	...
19537	Hyundai i10	Hyundai	i10	9	10723	Dealer	Petrol	Manual	19.81	1086	68.05	5	250000
19540	Maruti Ertiga	Maruti	Ertiga	2	18000	Dealer	Petrol	Manual	17.50	1373	91.10	7	925000
19541	Skoda Rapid	Skoda	Rapid	6	67000	Dealer	Diesel	Manual	21.14	1498	103.52	5	425000
19542	Mahindra XUV500	Mahindra	XUV500	5	3800000	Dealer	Diesel	Manual	16.00	2179	140.00	7	1225000
19543	Honda City	Honda	City	2	13000	Dealer	Petrol	Automatic	18.00	1497	117.60	5	1200000

Get Information About Our Dataset Like the Total Number of Rows, Total Number of Columns, Datatypes of Each Column And Memory Requirement

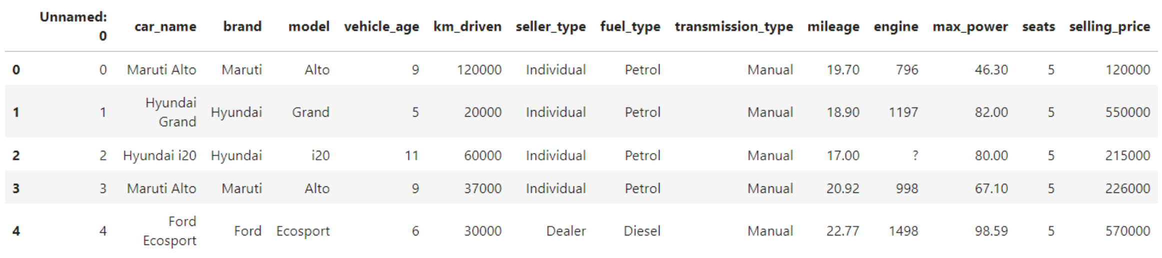
```
The Column Name, Record Count and Data Types are as follows:

<class 'pandas.core.frame.DataFrame'>

Index: 15411 entries, 0 to 19543

Data columns (total 13 columns):

 #   Column             Non-Null Count  Dtype

---  ------             --------------  -----

 0   car_name           15411 non-null  object

 1   brand              15411 non-null  object

 2   model              15411 non-null  object

 3   vehicle_age        15411 non-null  int64

 4   km_driven          15411 non-null  int64

 5   seller_type        15411 non-null  object

 6   fuel_type          15411 non-null  object

 7   transmission_type  15411 non-null  object

 8   mileage            15411 non-null  float64

 9   engine             15411 non-null  object

 10  max_power          15411 non-null  float64

 11  seats              15411 non-null  int64

 12  selling_price      15411 non-null  int64

dtypes: float64(2), int64(4), object(7)
```

Admin Access

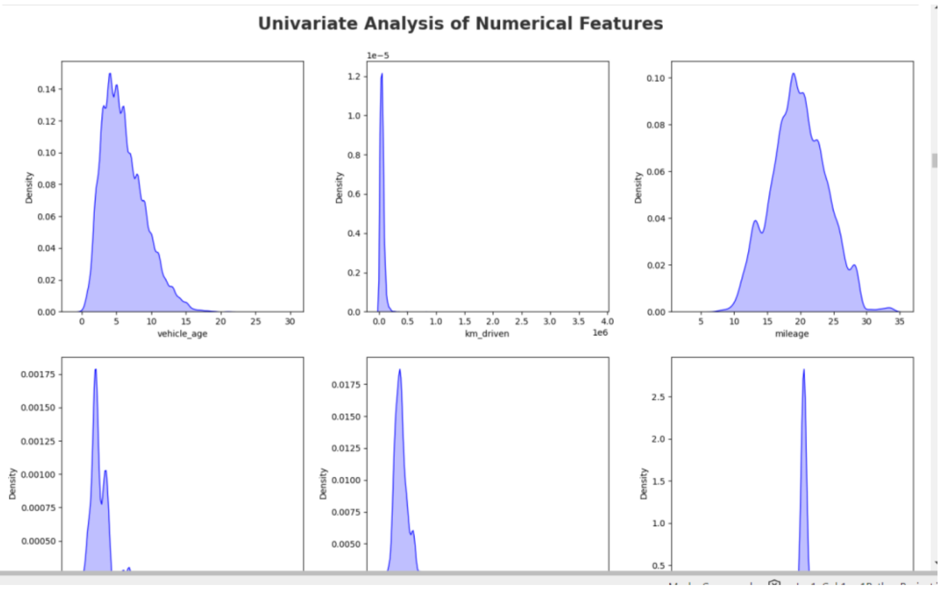


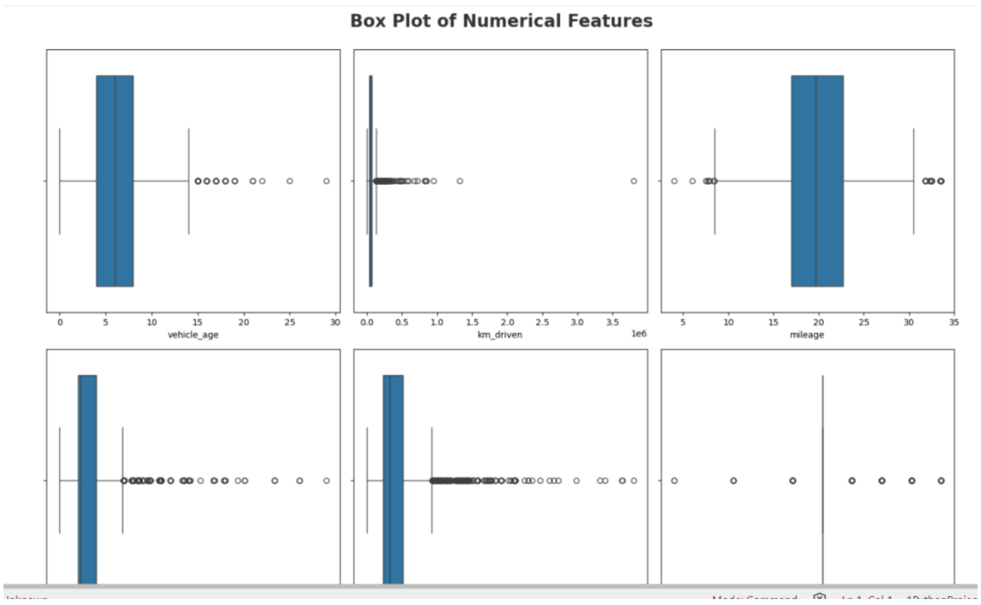
```
Enter 'A' to add data, 'U' to update data, 'D' to delete data, or 'Q' to quit:  q
       Unnamed: 0          car_name      brand     model  vehicle_age  \

0               0       Maruti Alto     Maruti      Alto            9

1               1     Hyundai Grand    Hyundai     Grand            5

2               2       Hyundai i20    Hyundai       i20           11

3               3       Maruti Alto     Maruti      Alto            9

4               4     Ford Ecosport       Ford  Ecosport            6

...           ...               ...        ...       ...          ...

15406       19537       Hyundai i10    Hyundai       i10            9

15407       19540     Maruti Ertiga     Maruti    Ertiga            2

15408       19541       Skoda Rapid      Skoda     Rapid            6

15409       19542  Mahindra XUV500   Mahindra    XUV500            5

15410       19543        Honda City      Honda      City            2


       km_driven seller_type fuel_type transmission_type  mileage  engine  \

0         120000  Individual    Petrol            Manual    19.70     796

1          20000  Individual    Petrol            Manual    18.90    1197

2          60000  Individual    Petrol            Manual    17.00       ?

3          37000  Individual    Petrol            Manual    20.92     998

4          30000      Dealer    Diesel            Manual    22.77    1498
```

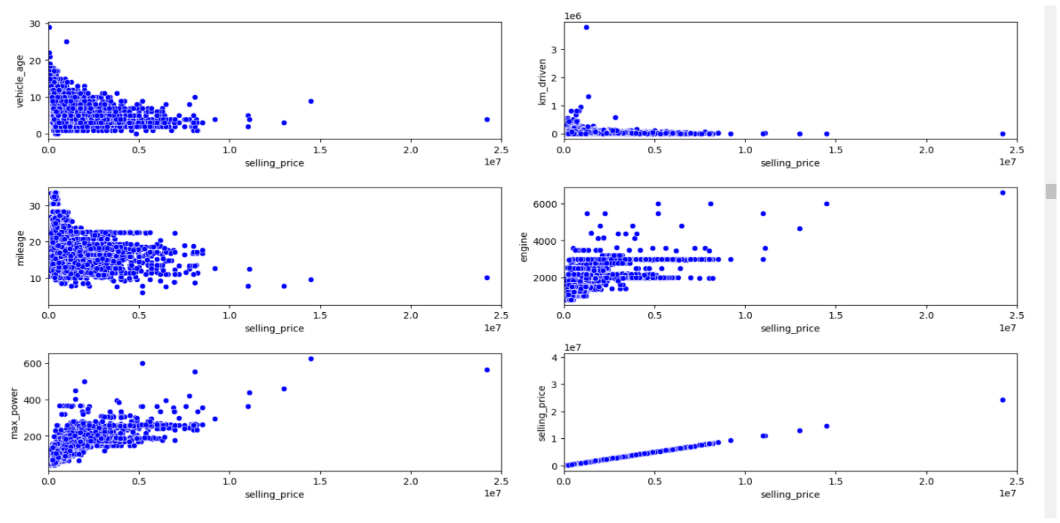
Data Cleaning

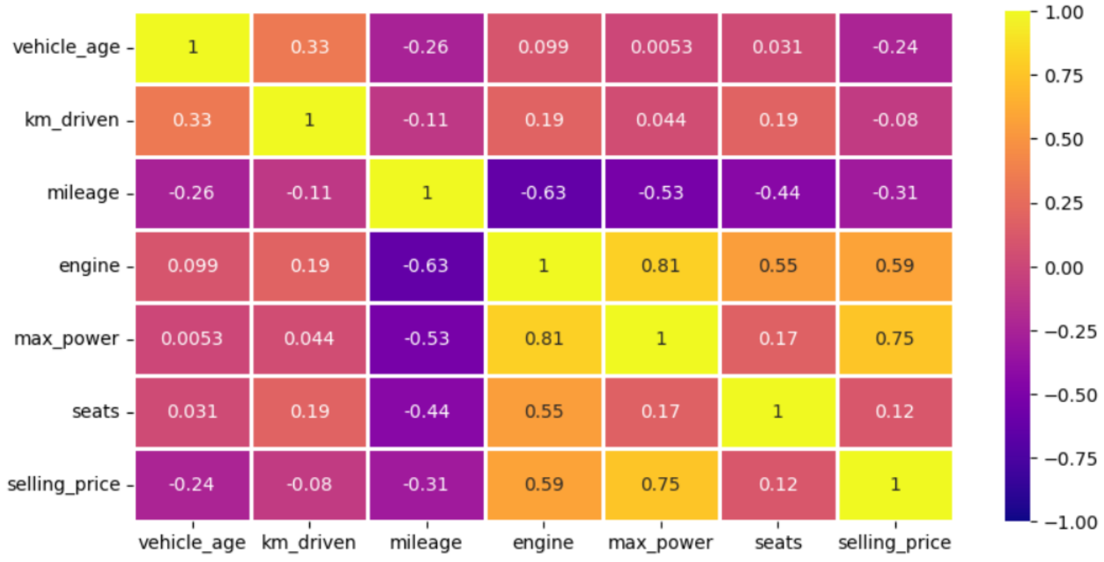


	Unnamed: 0	car_name	brand	model	vehicle_age	km_driven	seller_type	fuel_type	transmission_type	mileage	engine	max_power	seats	selling_price
0	0	Maruti Alto	Maruti	Alto	9	120000	Individual	Petrol	Manual	19.70	796	46.30	5	120000
1	1	Hyundai Grand	Hyundai	Grand	5	20000	Individual	Petrol	Manual	18.90	1197	82.00	5	550000
2	2	Hyundai i20	Hyundai	i20	11	60000	Individual	Petrol	Manual	17.00	?	80.00	5	215000
3	3	Maruti Alto	Maruti	Alto	9	37000	Individual	Petrol	Manual	20.92	998	67.10	5	226000
4	4	Ford Ecosport	Ford	Ecosport	6	30000	Dealer	Diesel	Manual	22.77	1498	98.59	5	570000

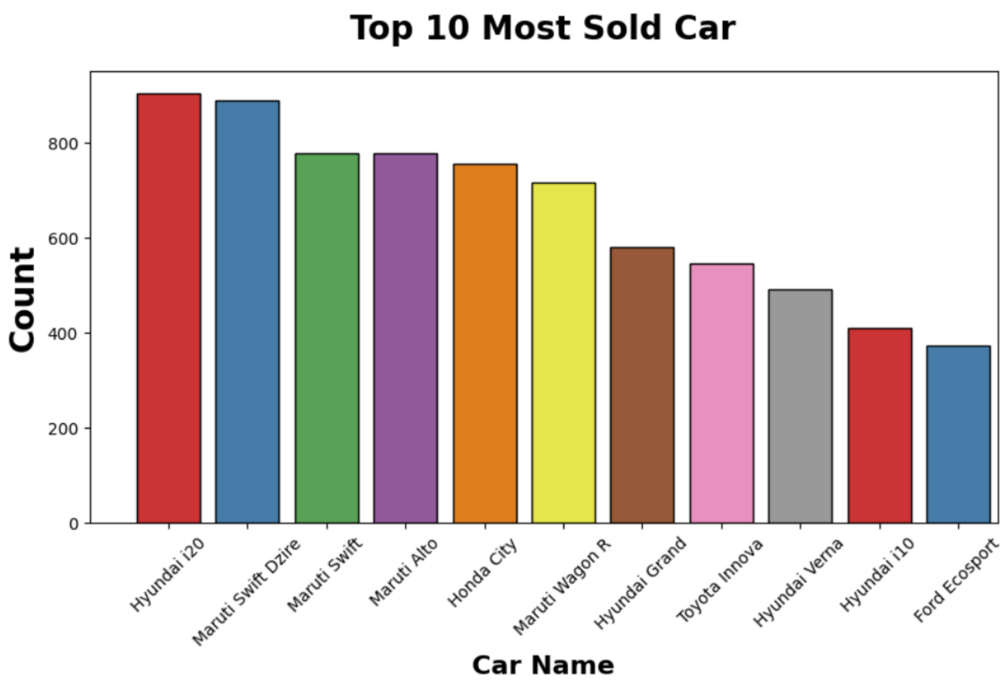


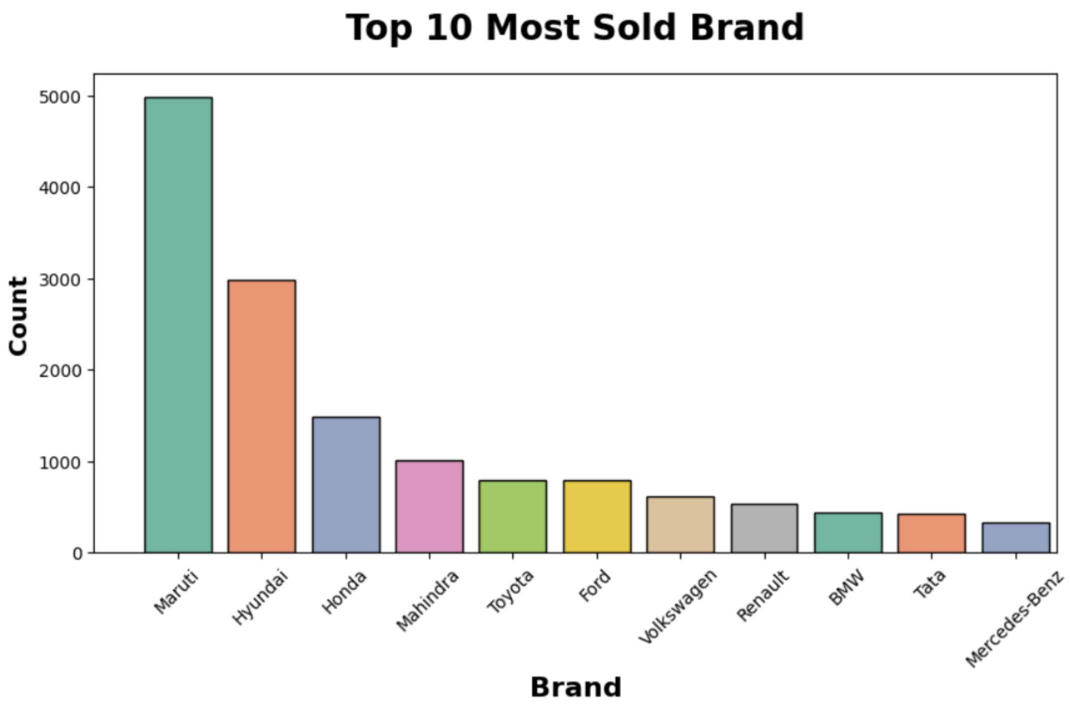

Box Plot of Numerical Features
vehicle_age
km_driven
1e6
mileage

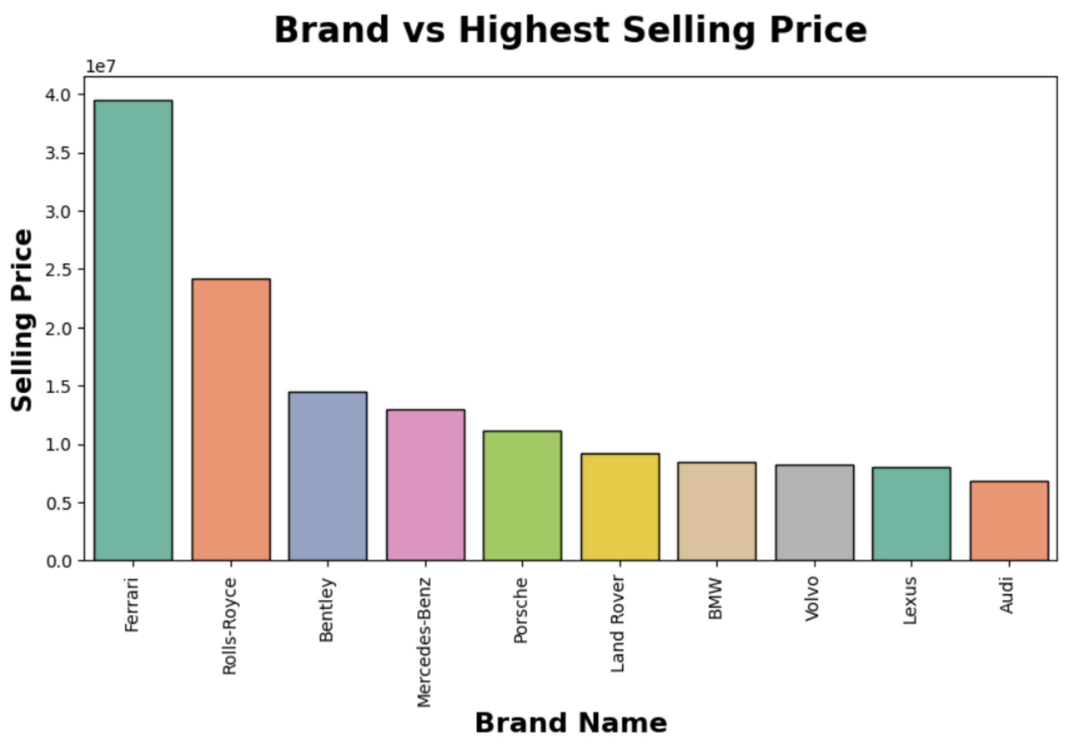

vehicle_age
km_driven
mileage
engine
max_power
selling_price
1e6
1e7

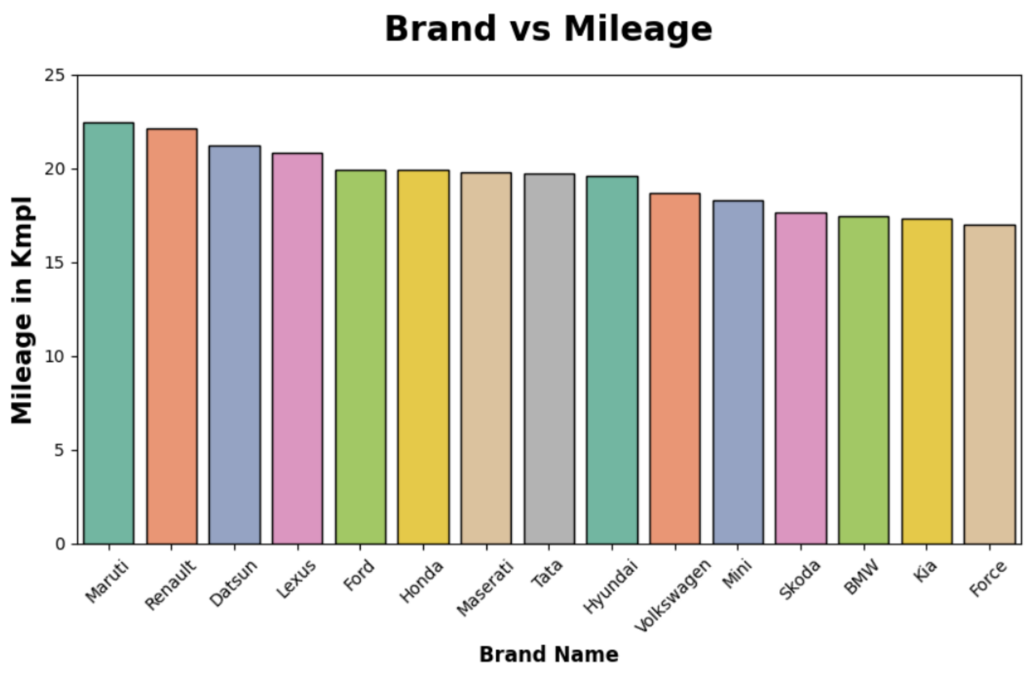

vehicle_age 1 0.33 -0.26 0.099 0.0053 0.031 -0.24
km_driven 0.33 1 -0.11 0.19 0.044 0.19 -0.08
mileage -0.26 -0.11 1 -0.63 -0.53 -0.44 -0.31
engine 0.099 0.19 -0.63 1 0.81 0.55 0.59
max_power 0.0053 0.044 -0.53 0.81 1 0.17 0.75
seats 0.031 0.19 -0.44 0.55 0.17 1 0.12
selling_price -0.24 -0.08 -0.31 0.59 0.75 0.12 1
vehicle_age km_driven mileage engine max_power seats selling_price
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00

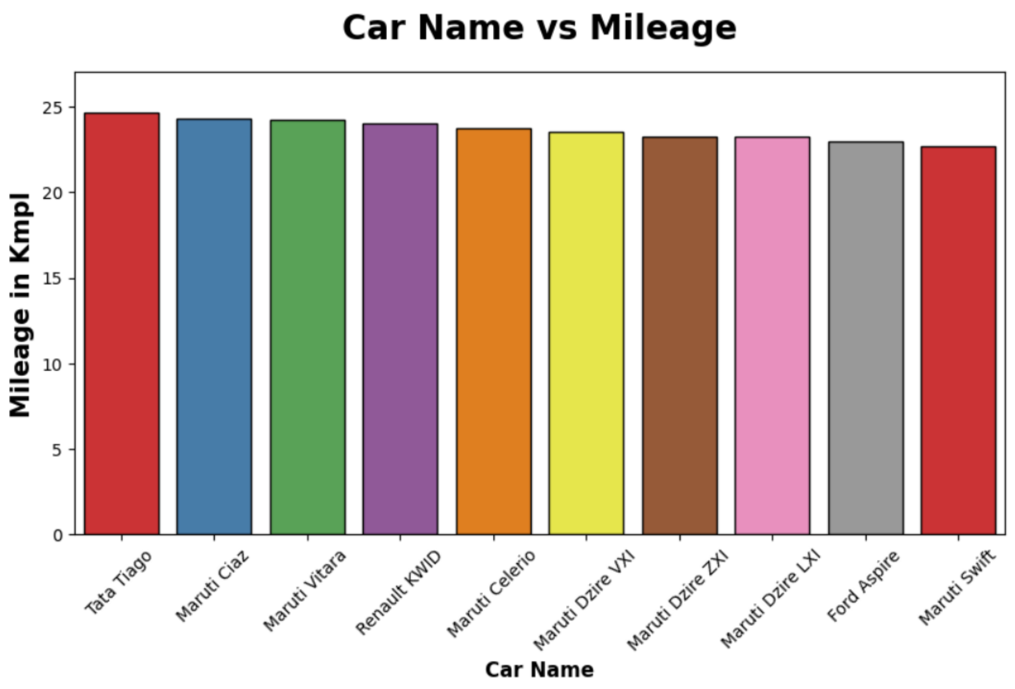
Data Visualization

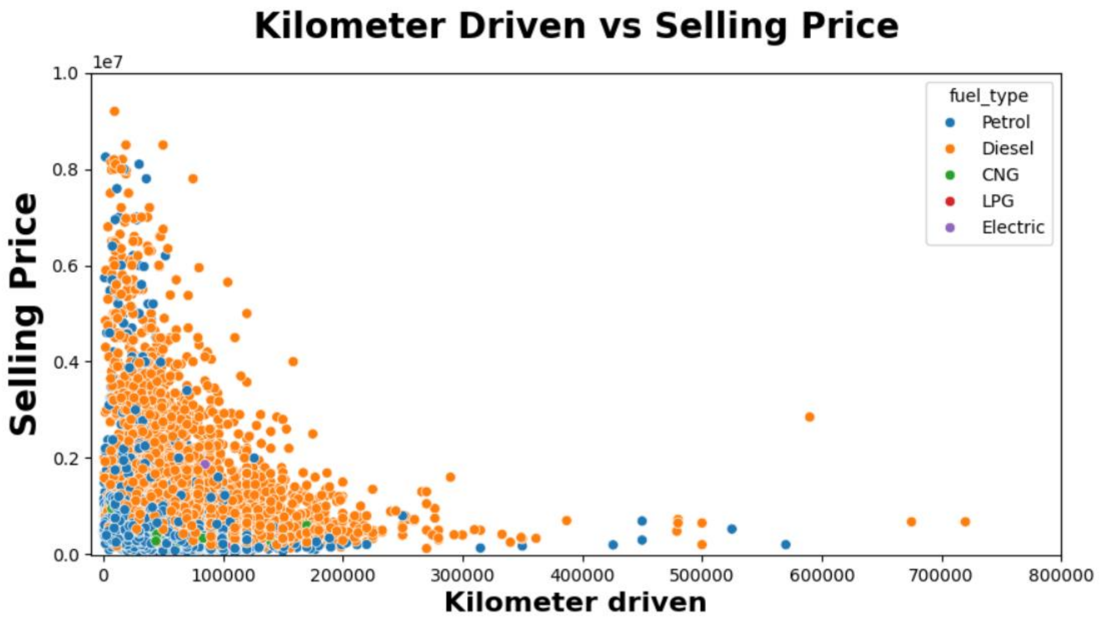


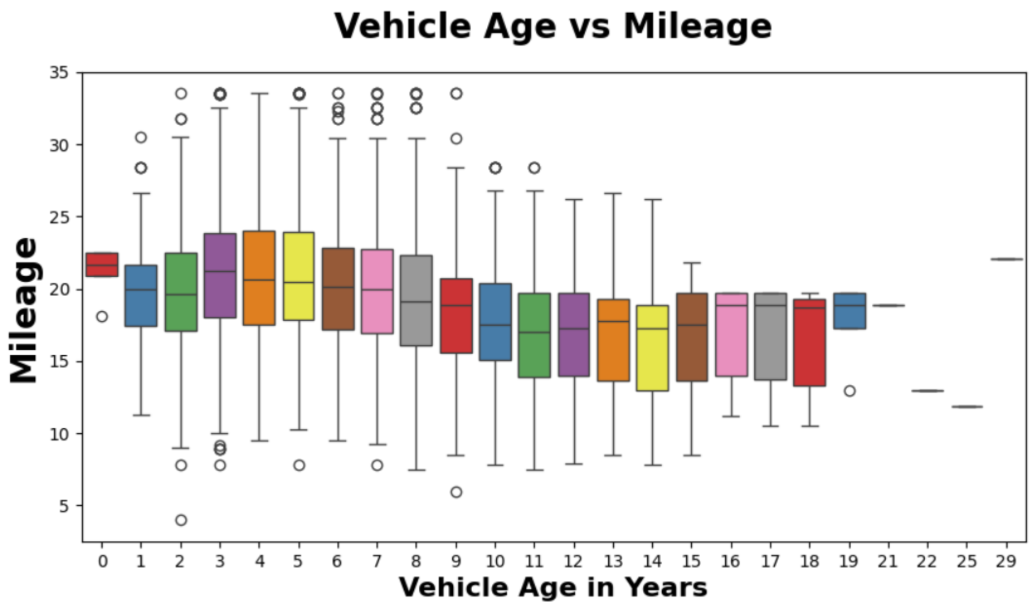


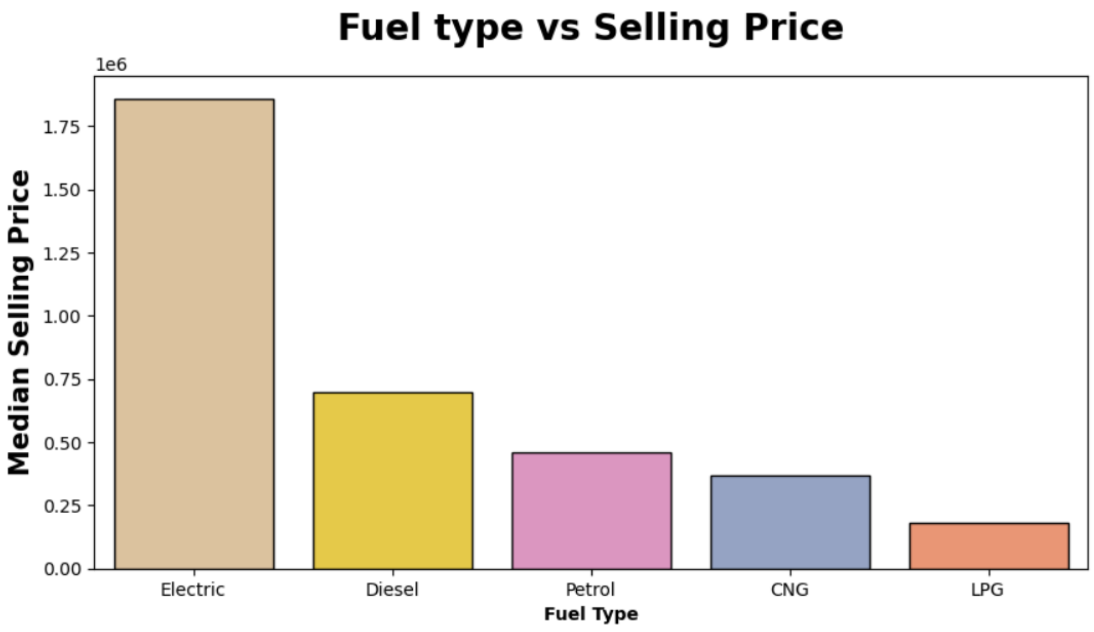

Brand vs Highest Selling Price
1e7
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Selling Price
Ferrari
Rolls-Royce
Bentley
Mercedes-Benz
Porsche
Land Rover
BMW
Volvo
Lexus
Audi
Brand Name

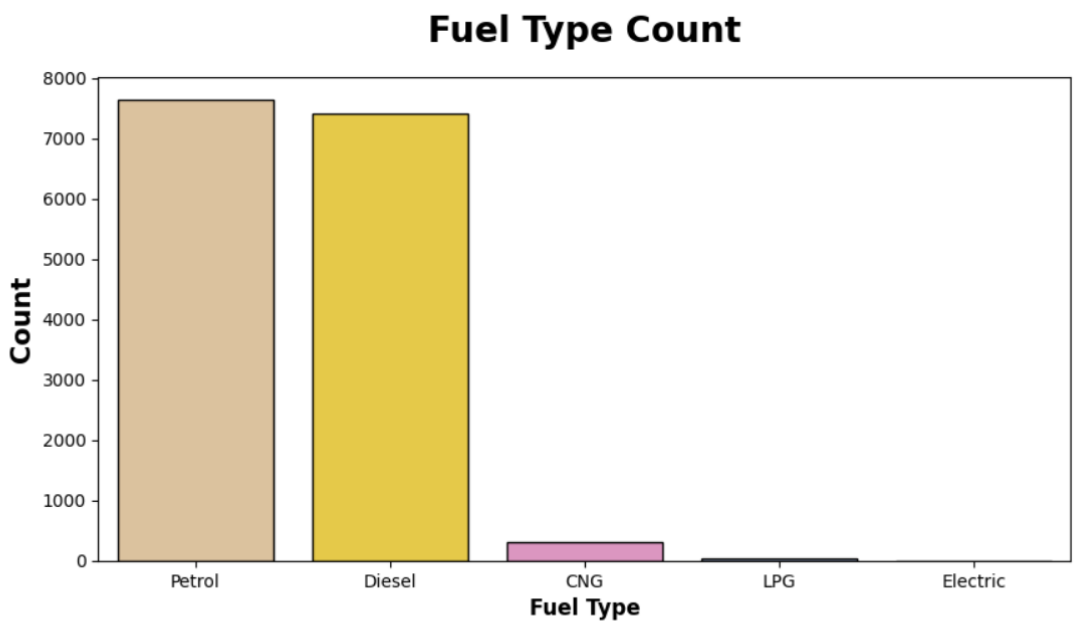

Brand vs Mileage
25
20
15
10
5
0
Mileage in Kmpl
Maruti
Renault
Datsun
Lexus
Ford
Honda
Maserati
Tata
Hyundai
Volkswagen
Mini
Skoda
BMW
Kia
Force
Brand Name


Car Name vs Mileage
Mileage in Kmpl
0
5
10
15
20
25
Tata Tiago
Maruti Ciaz
Maruti Vitara
Renault KWID
Maruti Celerio
Maruti Dzire VXI
Maruti Dzire ZXI
Maruti Dzire LXI
Ford Aspire
Maruti Swift
Car Name


Kilometer Driven vs Selling Price
1e7
Selling Price
0.0
0.2
0.4
0.6
0.8
1.0
0
100000
200000
300000
400000
500000
600000
700000
800000
Kilometer driven
fuel_type
Petrol
Diesel
CNG
LPG
Electric


Vehicle Age vs Mileage
Mileage
Vehicle Age in Years
35
30
25
20
15
10
5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 21 22 25 29


Fuel type vs Selling Price
1e6
Median Selling Price
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Electric
Diesel
Petrol
CNG
LPG
Fuel Type


Fuel Type Count
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
Petrol
Diesel
CNG
LPG
Electric
Fuel Type