Artificial Intelligence & Machine Learning

Predicting the Price of Used Cars & Bikes

Tejas M

Supraja P

Rachana JM

Collect the data of used cars and bikes and try to predict the price using the user input information of car or bike data.

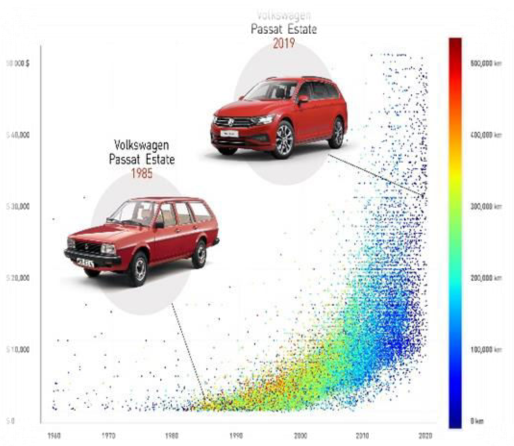
Introduction

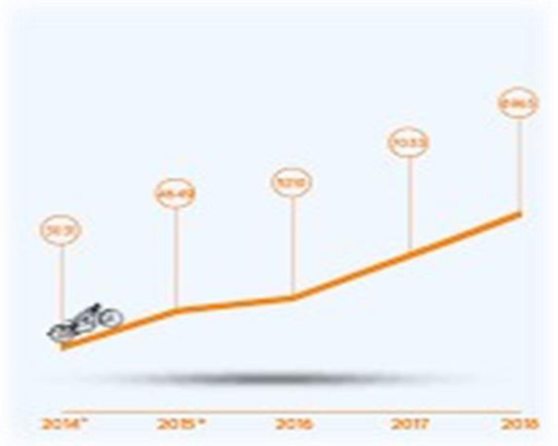
Buying a used vehicle over a brand-new one is always a smart choice given that the used vehicle you are going to buy is in good condition. However, the most important thing while buying a second-hand vehicle is understanding its fair market valuation. Many things are to be examined while purchasing a used vehicle to ensure that you are getting a perfect deal. From its condition to the price value, all things need to be tested.

A used car or bike, a first hand car or bike, or a second car or a bike, is a vehicle that has previously had one or more retail owners. Used cars or bikes are sold through a variety of outlets, including franchise and independent car or bike dealers,rental cars or bikes companies, buy here pay here dealerships, leasing offices, auctions, and private party sales. Some car or bike retailers offer "no-haggle prices","certified" used cars and bikes, and extended service or plans or warranties.

Used car or bikes pricing reports typically produce three forms of pricing information.

- Dealer or retail price is the price expected to pay if buying from a licensed new-car or bike or used-car or bike dealer.
- Dealer trade-in price or wholesale price is the price a shopper should expect to receive from a dealer if trading in a car or bike.This is also the price that a dealer will typically pay for a car or bike at a dealer wholesale auction.
- Private-party price is the price expected to pay if buying from an individual. A private-party seller is hoping to get more money than they would with a trade-in to a dealer. A private-party buyer is hoping to pay less than the dealer retail price.





Code to Predict the Price of Used Cars or Bikes

Importing the packages

First, import all the libraries/packages which are necessary to analyse the given dataset about used cars or bikes.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
```

Importing the dataset

data = pd.read_csv('dataset.csv')
print(data)

```
[ ] data = pd.read_csv('dataset.csv')
```

```
print(data)
```

```
                         Vehicle_Name Vehicle_Type  ...  Kilometers Owner_Type
0             Royal Enfield Classic 350         Bike  ...         350          0
1                             Honda Dio         Bike  ...        5650          3
2      Royal Enfield Classic Gunmetal Grey      Bike  ...       12000          0
3        Yamaha Fazer FI V 2.0 [2016-2018]      Bike  ...       23000          0
4                  Yamaha SZ [2013-2014]        Bike  ...       21000          0
..                                   ...          ...  ...         ...        ...
594                    Maruti Alto 800 LXI          Car  ...       49000          0
595          Maruti Swift Dzire 1.2 Vxi BSIV        Car  ...       70000          0
596       Mahindra Scorpio VLS AT 2.2 mHAWK         Car  ...       90000          0
597                   Mahindra Bolero SLX          Car  ...       25000          0
598          Mahindra XUV500 W11 Option AWD        Car  ...        4000          0

[599 rows x 6 columns]
```

As the dataset is too large to analyse, here only the head part of the dataset is extracted for better analyzation of data.

```
[ ] data.head()
```

	Vehicle_Name	Vehicle_Type	Selling_Price	Fuel_Type	Kilometers	Owner_Type
0	Royal Enfield Classic 350	Bike	175000	Petrol	350	First Owner
1	Honda Dio	Bike	45000	Petrol	5650	Second Owner
2	Royal Enfield Classic Gunmetal Grey	Bike	150000	Petrol	12000	First Owner
3	Yamaha Fazer FI V 2.0 [2016-2018]	Bike	65000	Petrol	23000	First Owner
4	Yamaha SZ [2013-2014]	Bike	20000	Petrol	21000	First Owner

Data.Info() is to get the concise summary of the dataset in the dataframe.

```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 599 entries, 0 to 598
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Vehicle_Name   599 non-null    object
 1   Vehicle_Type   599 non-null    object
 2   Selling_Price  599 non-null    int64
 3   Fuel_Type      599 non-null    object
 4   Kilometers     599 non-null    int64
 5   Owner_Type     599 non-null    object
dtypes: int64(2), object(4)
memory usage: 28.2+ KB
```

Data.isnull() is to check whether the data cells are empty or not if empty it shows true if not empty it shows false.

```
[ ] data.isnull()
```

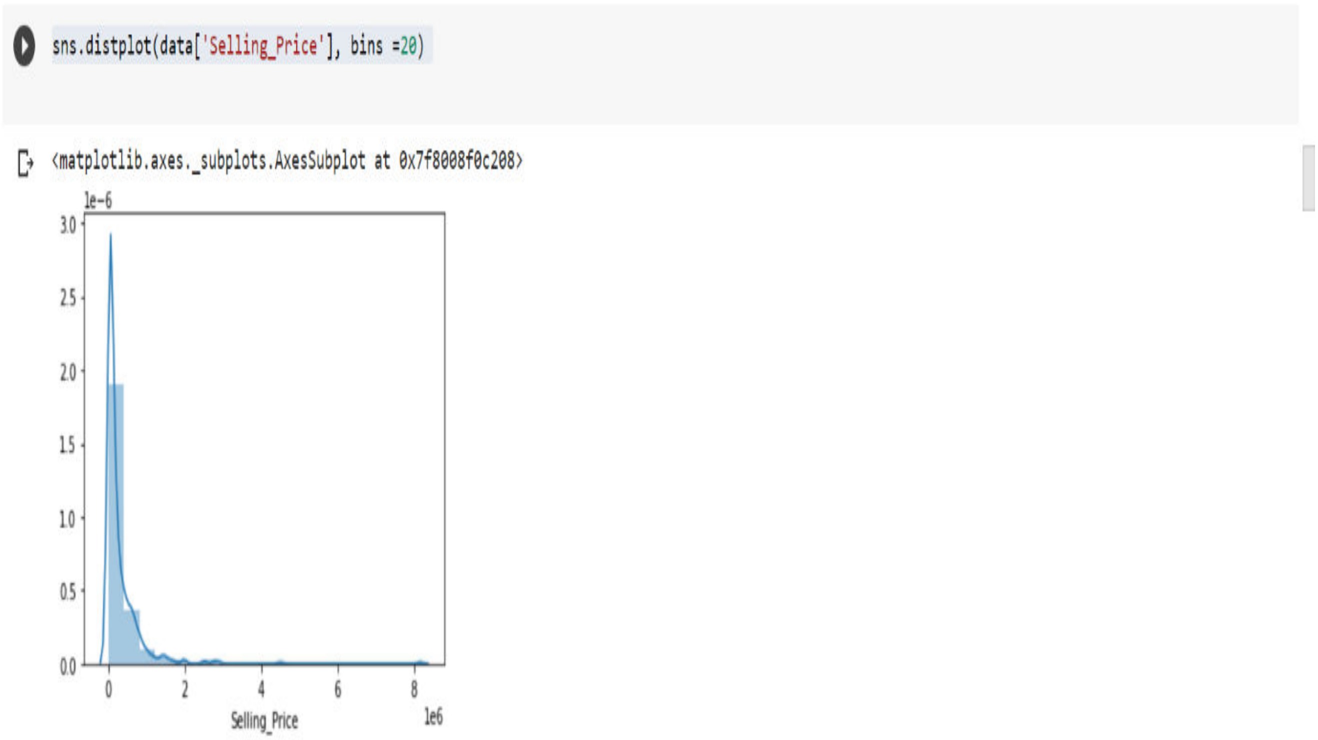
	Vehicle_Name	Vehicle_Type	Selling_Price	Fuel_Type	Kilometers	Owner_Type
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
594	False	False	False	False	False	False
595	False	False	False	False	False	False
596	False	False	False	False	False	False
597	False	False	False	False	False	False
598	False	False	False	False	False	False

599 rows × 6 columns

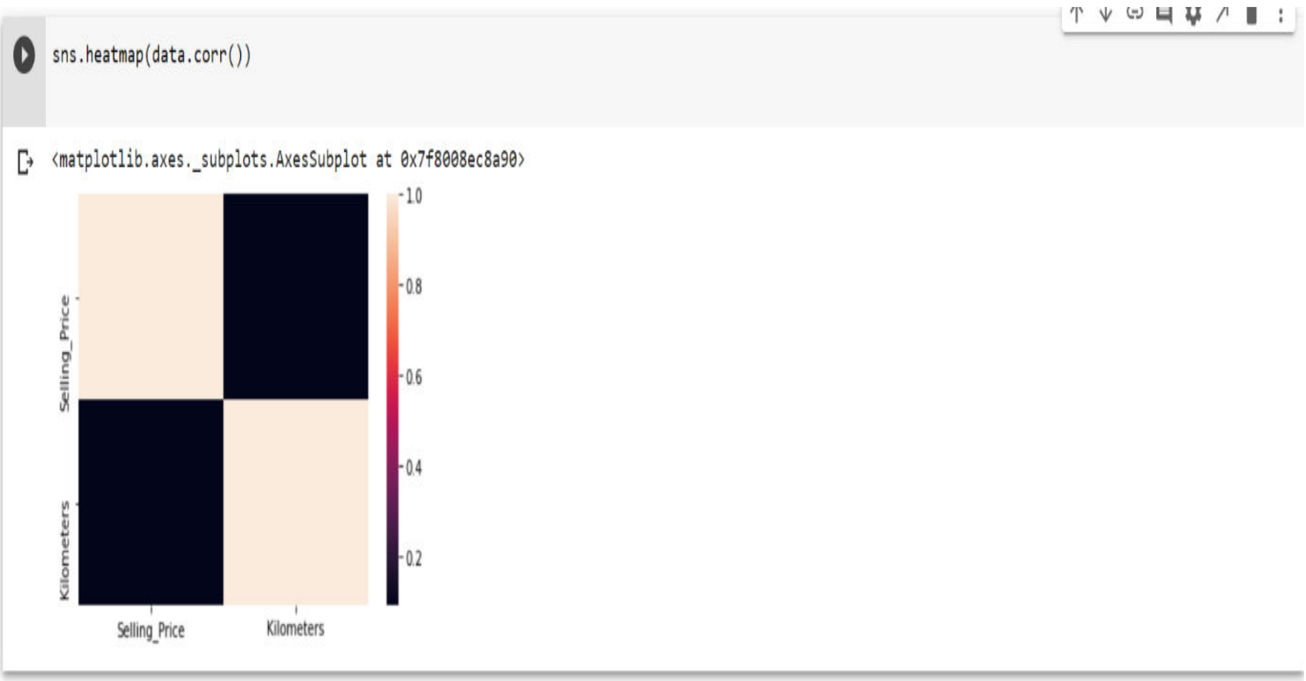
sns.pairplot(data)is to plot pairwise relationship between dependent and independent values.

Data Visualization

sns.distplot(data['Selling_Price'], bins =20)



sns.heatmap(data.corr())



Convert categorical variable into dummy/indicator variables

```
x = pd.get_dummies(data,columns = ["Vehicle_Name", "Vehicle_Type", "Fuel_Type", "
Owner_Type"],drop_first = True)
y = data['Selling_Price']
```

Linear Regression

```
from sklearn.linear_model import LinearRegression
```

LinearRegression fits a linear model with coefficients to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

```
lm = LinearRegression()
```

Training and Testing the Data

In this process, 20% of the data was split for the test data and 80% of the data was taken as train data.

```
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, random_state=45
)
lm.fit(X_train,y_train)
print(lm.intercept_)
coeff_df = pd.DataFrame(lm.coef_,x.columns,columns=['Coefficient'])
```

coeff_df
p = lm.predict(X_test)

```
[ ] from sklearn.linear_model import LinearRegression
```

```
[ ] lm = LinearRegression()
```

```
[ ] 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, random_state=45)
```

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

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[ ] print(lm.intercept_)
```

```
4.0745362639427185e-10
```

```
coeff_df = pd.DataFrame(lm.coef_,x.columns,columns=['Coefficient'])
coeff_df
```

	Coefficient
Selling_Price	1.000000e+00
Kilometers	-3.788225e-16
Vehicle_Name_Audi A4 2.0 TDI 177 Bhp Premium Plus	5.071444e-21
Vehicle_Name_Audi A6 2.0 TDI Design Edition	8.567308e-10
Vehicle_Name_Audi A6 2.0 TDI Premium Plus	6.235031e-10
...	...
Fuel_Type_Petrol	-1.326777e-10
Owner_Type_1	-5.904774e-11
Owner_Type_2	1.284803e-10
Owner_Type_3	6.316069e-11
Owner_Type_4	-1.293293e-10

398 rows × 1 columns

Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R^2 score of 0.0.

Unlike most other scores, R^2 score may be negative (it need not actually be the square of a quantity R).

This metric is not well-defined for single samples and will return a NaN value if n_samples is less than two.

```
[ ] p = lm.predict(X_test)
```

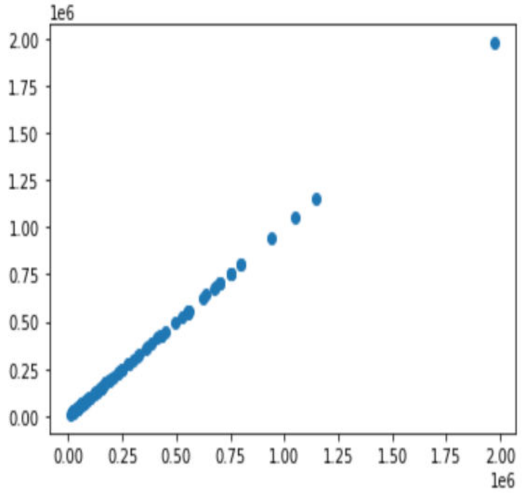
```
[ ] from sklearn.metrics import r2_score
    r2_score(y_test, p)
```

```
1.0
```

A scatter plot of *y* vs. *x* with varying marker size and/or color.

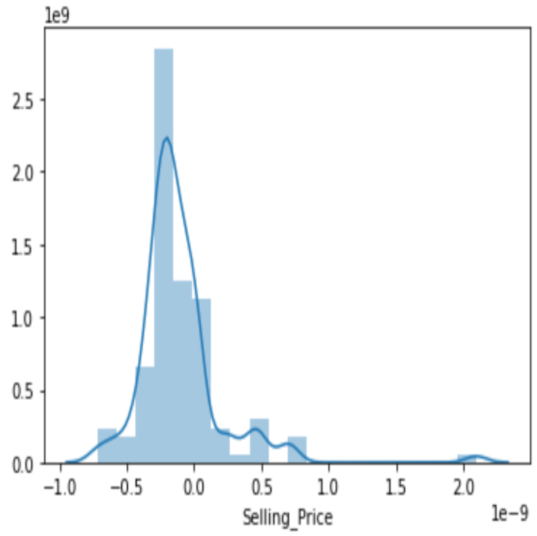
```
plt.scatter(y_test,p)
```

```
<matplotlib.collections.PathCollection at 0x7f800483d160>
```



Displot() is a figure-level function with a similar flexibility over the kind of plot to draw.

```
[ ] sns.distplot((y_test-p),bins=20);
```




```
[ ] from sklearn import metrics
    print('MAE:', metrics.mean_absolute_error(y_test, p))
    print('MSE:', metrics.mean_squared_error(y_test, p))
    print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, p)))
```

```
MAE: 2.395609044469893e-10
MSE: 1.1530737817404113e-19
RMSE: 3.395694011156499e-10
```

The sklearn module implements several loss, score, and utility functions to measure regression performance. Some of those have been enhanced to handle the multioutputcase.

Mean Absolute Error

The mean absolute error function computes mean absolute error a risk metric corresponding to the expected value of the absolute error loss or l1-norm loss.

If y^i is the predicted value of the i-th sample, and yi is the corresponding true value, then the mean absolute error (MAE) estimated over n samples is defined as

MAE(y,y^)=1nsamples∑i=0nsamples−1|yi−y^i|.

Mean Squared Error

The mean squared error function computes the mean squared error, a risk metric corresponding to the expected value of the squared (quadratic) error or loss.

If y^i is the predicted value of the i-th sample, and yi is the corresponding true value, then the mean squared error (MSE) estimated over n samples is defined as

MSE(y,y^)=1nsamples∑i=0nsamples−1(yi−y^i)2.

Root Mean Square Error

"How similar, on average, are the numbers in list1 to list2?". The two lists must be the same size. I want to "wash out the noise between any two given elements, wash out the size of the data collected, and get a single number feel for change over time".

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - p_i)^2}$$

Conclusion

By performing different models, it was aimed to get different perspectives and eventually compared their performance. With this study, it purpose was to predict prices of used cars by using a dataset . With the help of the data visualizations and exploratory data analysis, the dataset was uncovered and features were explored deeply. The relation between features were examined. At the last stage, predictive models were applied to predict price of cars or bikes in an order. By considering all metrics, it can be concluded that the best model for the prediction for used car prices.Regression model gave the best MAE, MSE and RMSE values.