USED CARS PRICE PREDICTION

MANOJ KUMAR D

ABSTRACT:

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in India. Our results show that Random Forest model and K-Nearest Neighbors with linear regression yield the best results, but are compute heavy. Conventional linear regression also yielded satisfactory results, with the advantage of a significantly lower training time in comparison to the above methods.

INTRODUCTION:

Driverless cars are getting closer to reality and at a faster pace than ever. But it is still a bit farfetched dream to have one in your garage. For the time being, there are still a lot of combustion and hybrid cars that roar around the road, for some it chills. Though the overall data on sales of automobiles shows a huge drop in sales in the last couple of years, cars are still a big attraction for many. Cars are more than just a utility for many. They are often the pride and status of the family. We all have different tastes when it comes to owning a car or at least when thinking of owning one.

BUSINESS UNDERSTANDING:

- Companies can restrict the selling price of the used car being posted by the customer in their respective websites.
- Companies can provide a visualization to customers for a better understanding of their car selling price.
- Companies can have Fraud Customers who are posting cars for higher prices.
- Companies can expand their network based on the number of cars being sold the next year by prediction.

LIBRARIES USED:

- NumPy (for Numerical Analysis)
- Pandas (for handling data files)
- Matplotlib (for visualizations inline & figure settings)
- Seaborn (for better relational visualizations)
- Scikit Learn (for model building & data pre-processing)

ALGORITHM:

- Linear Regression
- K-Nearest Neighbors
- Random Forest Regression

DATASET:

For this project, we are using the dataset on used car sales from all over the United States, available on Kaggle

You can find data on the link as follows: https://github.com/manojd441/used-cars-price-prediction

DATA EXPLORATION:

Parameter	Description
Name	The brand and model of the car
Location	The location in which the car is being sold or is available for purchase
Year	The year or edition of the model
Kilometers_Driven	The total kilometres driven in the car by the previous owner(s) in KM
Fuel_Type	The type of fuel used by the car
Transmission	The type of transmission used by the car
Owner_Type	Whether the ownership is Firsthand, Second hand or other
Mileage	The standard mileage offered by the car company in kmpl or km/kg
Engine	The displacement volume of the engine in cc
Power	The maximum power of the engine in bhp
Seats	The number of seats in the car
New_Price	The price of a new car of the same model
Price	The price of the used car in INR Lakhs

The data (for predictive modeling) contains the information of cars sold between the periods of 1998 to 2019. The data has many crucial factors which are important for the prediction of the price. As the dataset is too large to analyze, here only the head part of the dataset is extracted for better analyzation of data.

```
df = pd.read_excel("data/Data_Train.xlsx")
df.head()
```

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74

Now we can also know ,what type of data is being used

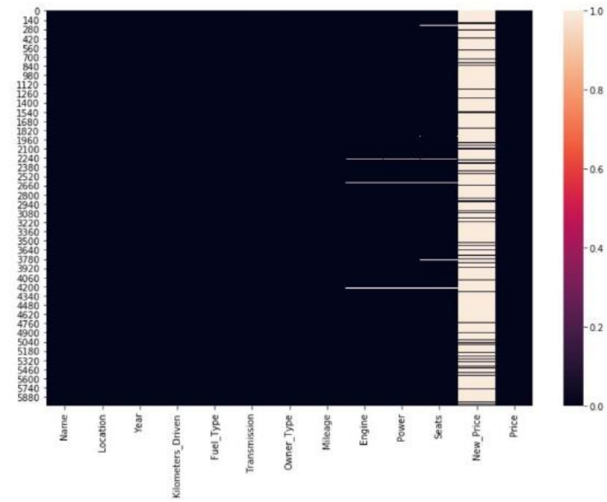
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               6019 non-null   object
 1   Location           6019 non-null   object
 2   Year               6019 non-null   int64
 3   Kilometers_Driven  6019 non-null   int64
 4   Fuel_Type          6019 non-null   object
 5   Transmission       6019 non-null   object
 6   Owner_Type         6019 non-null   object
 7   Mileage            6017 non-null   object
 8   Engine             5983 non-null   object
 9   Power              5983 non-null   object
 10  Seats              5977 non-null   float64
 11  New_Price          824 non-null    object
 12  Price              6019 non-null   float64
dtypes: float64(2), int64(2), object(9)
memory usage: 611.4+ KB
```

Using Heatmap we can check the Null Values availability in the Data provided.

```
plt.figure(figsize=(12,8))
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x213fcdafee0>
```

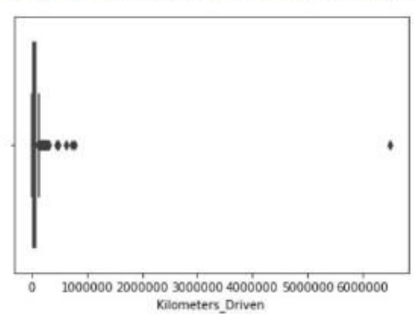


DATA CLEANING:

- In our current project data cleaning plays a major role
- I have dropped a New_Price column because it has many null values
- I have dropped the data rows that consist of null values
- After looking at the box plots, I have seen that too many outliers are restricting the plot structure so removed some of the data cells with higher value to avoid model underfitting.
- Example: Removing the top 50 outliers from Kilometers_Driven column.

```
sns.boxplot(df.Kilometers_Driven)
```

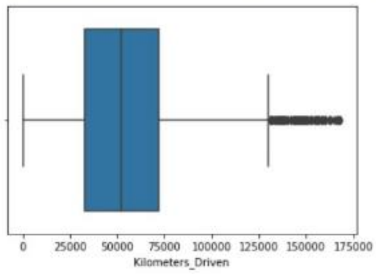
```
<matplotlib.axes._subplots.AxesSubplot at 0x21382d7f700>
```



```python
for i in np.arange(50):
    index = df[df.Kilometers_Driven == df.Kilometers_Driven.max()].index
    df.drop(index=index, inplace=True, axis=1)
df.reset_index(inplace=True)
```

```python
sns.boxplot(df.Kilometers_Driven)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x213ffd841f0>
```

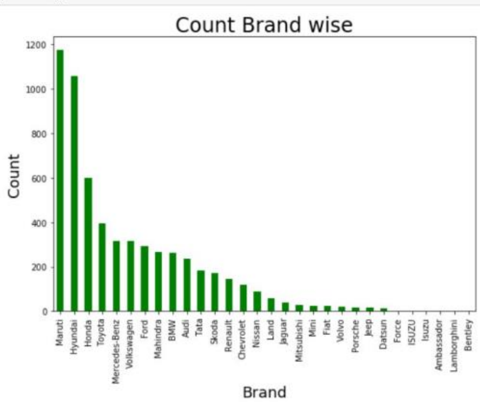


DATA VISUALIZATION:

Some of the statistics obtained from the data visualizations are

- Frequency Distributions

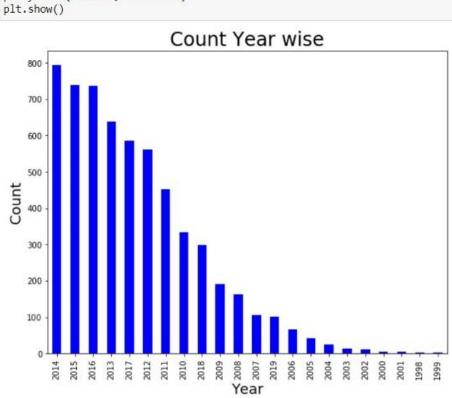
```python
plt.figure(figsize = (9,6))
pd.value_counts(df.Brand).plot.bar(color='g')
plt.title('Count Brand wise', size = 24)
plt.xlabel('Brand', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```



✓ Maruti, Hyundai & Honda tops the list as most selling used car companies.

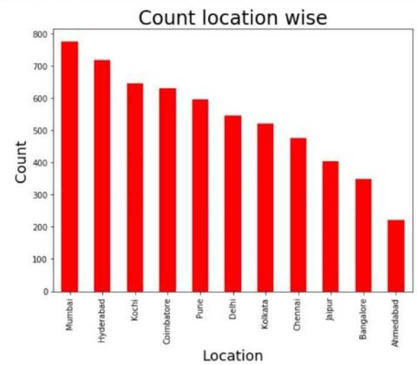
✓ Ambassador & ISUZU makes least in the list

```python
plt.figure(figsize = (9,7))
pd.value_counts(df.Year).plot.bar(color='b')
plt.title('Count Year wise', size = 24)
plt.xlabel('Year', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```



✓ 2014,2015,2016 tops the list as most selling used cars in a year

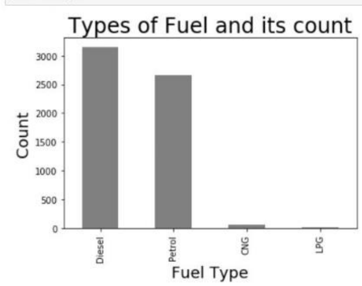
```python
plt.figure(figsize = (8,6))
pd.value_counts(df.Location).plot.bar(color='r')
plt.title('Count location wise', size = 24)
plt.xlabel('Location', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```



✓ Mumbai & Hyderabad stands as the top location where used cars sold in most.

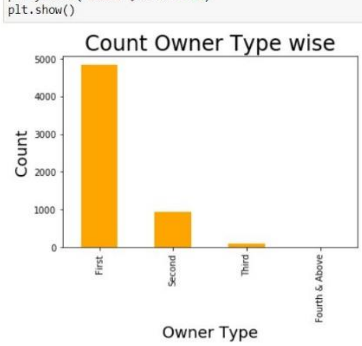
✓ Bangalore & Ahmedabad stands least in the list

```python
plt.figure(figsize = (6,4))
pd.value_counts(df.Fuel_Type).plot.bar(color='grey')
plt.title('Types of Fuel and its count', size = 24)
plt.xlabel('Fuel Type', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```



✓ 95% of selling cars use Petrol & Diesel

```python
plt.figure(figsize = (6,4))
pd.value_counts(df.Owner_Type).plot.bar(color='orange')
plt.title('Count Owner Type wise', size = 24)
plt.xlabel('Owner Type', size = 18)
plt.ylabel('Count', size = 18)
plt.show()
```



✓ More than 75% of cars handled by a single owner (First-Hand)

✓ 15% of cars are second handed

MODEL BUILDING:

Before building the model, convert the categorical data into numerical categories for machines to understand using the sci-kit learn preprocessing LabelEncoder pre-defined function.

```
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

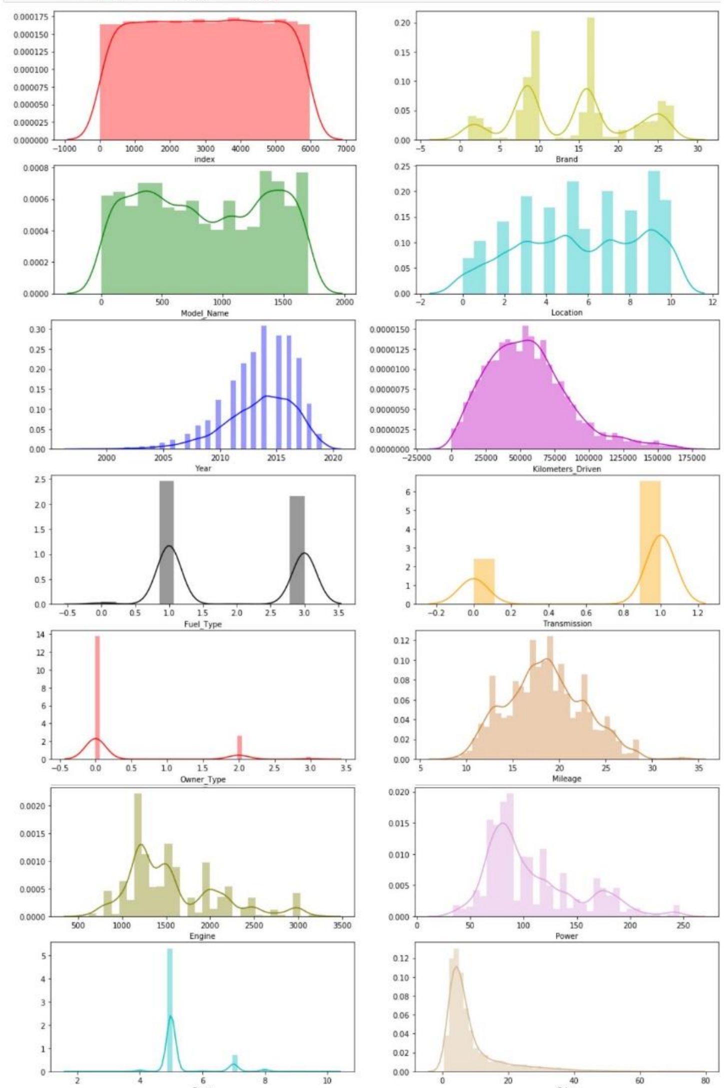
```
df.Brand = LabelEncoder().fit_transform(df.Brand)
df.Model_Name = LabelEncoder().fit_transform(df.Model_Name)
df.Location = LabelEncoder().fit_transform(df.Location)
df.Owner_Type = LabelEncoder().fit_transform(df.Owner_Type)
df.Fuel_Type = LabelEncoder().fit_transform(df.Fuel_Type)
df.Transmission = LabelEncoder().fit_transform(df.Transmission)
```

```
df.head()
```

	Brand	Model_Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
0	18	1579	9	2010	72000	0	1	0	26.60	998.0	58.16	5.0	1.75
1	10	450	10	2015	41000	1	1	0	19.67	1582.0	126.20	5.0	12.50
2	9	889	2	2011	46000	3	1	0	18.20	1199.0	88.70	5.0	4.50
3	18	606	2	2012	87000	1	1	0	20.77	1248.0	88.76	7.0	6.00
4	1	93	3	2013	40670	1	0	2	15.20	1968.0	140.80	5.0	17.74

The Frequency distribution of each field data represents below.

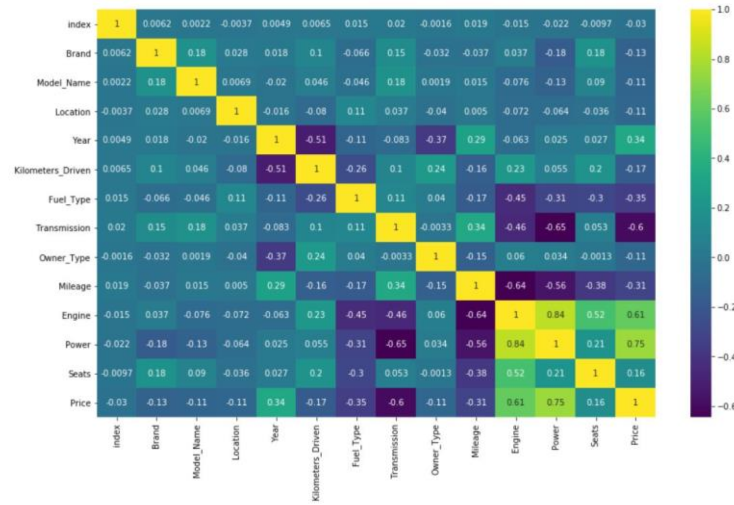
```
plt.figure(figsize=(16,26))
c = ['r','y','g','c','b','m','k','orange','r', 'peru', 'olive','plum', 'c', 'tan']
for i in np.arange(0,14):
    plt.subplot(7,2,i+1)
    sns.distplot(df[df.columns[i]], color=c[i])
```



Before training the model, we have to check the correlation between the dependent & independent variables.

```
plt.figure(figsize=(16,10))
ax = sns.heatmap(df.corr(), annot=True, cmap='viridis')
ax
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x213fcad2e80>
```



- ✓ Negatively co-related values will underfit the data, so check the model fitting procedure by dropping highly negative related fields of data to achieve better accuracies.
- ✓ By altering the dropping values of our Data, we can conclude, by dropping Model Name & Transmission fields gives our model the best fir & validation accuracy
- ✓ We can see that the price is highly related to Engine and Power.

Split the data into X and Y

```
X = df.drop(labels=['Price', 'Model_Name', 'Transmission'], axis=1)
Y=df['Price']
```

Scale the data to achieve accurate training results and split the data for training and testing.

```
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=25)
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(4215, 10) (1405, 10) (4215,) (1405,)
```

Import the libraries which are required for model building.

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,r2_score
```

- Fit the data using respective models & predict the value
- Obtain the predicted values using model. Predict () function and store the values in a variable.
- As the function is a regression model, score function will help us find the accuracy of our model. Our model will be much accurate when the score is nearer to 1.0

```
reg=LinearRegression()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
print('Mean Squared Error:',mean_squared_error(y_test,y_pred))
print('Accuracy:',reg.score(x_test,y_test))
```

```
Mean Squared Error: 24.980735746250527
Accuracy: 0.668742862582052
```

Using Linear Regression model gives 66.87% accuracy for the given data.

```
reg=KNeighborsRegressor()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
print('Mean Squared Error:',mean_squared_error(y_test,y_pred))
print('Accuracy:',reg.score(x_test,y_test))
```

```
Mean Squared Error: 13.363972971530249
Accuracy: 0.8227869876993357
```

Using K-Neighbors Regression model gives 82.27% accuracy for the given data.

```
reg=RandomForestRegressor()
reg.fit(x_train,y_train)
y_pred=reg.predict(x_test)
print('Mean Squared Error:',mean_squared_error(y_test,y_pred))
print('Accuracy:',reg.score(x_test,y_test))
```

```
Mean Squared Error: 7.145036887869365
Accuracy: 90.52532123047456
```

Using Random Forest Regression model gives 90.57% accuracy for the given data.

CONCLUSION:

- Random Forest Regressor gives best accurate model when compared with Linear regressor& K-Neighbors regressor.
- Data Cleaning played a major role in achieving better accuracy
- Visualizing Data helped us a lot to identify the patterns of data
- Removing the outliers increased the model accuracy by 10%, which is a huge improvement
- Successfully obtained a Random Forest Regression model with 90.57% of accuracy from the data given.