CAR PRICE PREDICTION PROJECT

Submitted by:

Vijay Venkatesh K

ACKNOWLEDGMENT

I have taken efforts in this project by referring from below websites,

1.www.google.com

2.www.analyticsvidhya.com
3.www.medium.com

INTRODUCTION

BUSINESS PROBLEM FRAMING

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. We have to make car price valuation model.

REVIEW OF LITERATURE

This project is more about exploration, feature engineering and classification that can be done on this data. Since we scrape huge amount of data that includes more car related features, we can do better data exploration and derive some interesting features using the available columns.

The goal of this project is to build an application which can predict the car prices with the help of other features. In the long term, this would allow people to better explain and reviewing their purchase with each other in this increasingly digital world.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL/ANALYTICAL MODELLING OF THE PROBLEM

In our scrapped dataset, our target variable "**selling_price**" is a continuous variable. Therefore, we will be handling this modelling problem as classification.

This project is done in two parts:

- ➔ Data Collection phase
- ➔ Model building phase

Data Collection phase:

You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you. More the data better the model

In this section You need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometres, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback. Note – The data which you are collecting is important to us. Kindly don't share it on any public platforms.

Model building phase:

➔After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

➔Follow the complete life cycle of data science. Include all the steps like:

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

DATA SOURCES AND THEIR FORMATS

➔We collected the data from difference websites like olx and cars24. The data is scrapped using Web scraping technique and the framework used is Selenium.

➔We scrapped nearly 10000 of the data and saved each data in a separate data frame

➔In the end, we combined all the data frames into a single data frame and it looks like as follows:

```
#Creating a dataframe and checking the data extracted
olx=pd.DataFrame({'name':brand,'year':year,'selling_price':price,'km_driven':km,'fuel':fuel,'transmission':transmission})
```

```
olx
```

	name	year	selling_price	km_driven	fuel	transmission
0	Hyundai	2017	5,25,000	2,200 km	Petrol	Manual
1	Hyundai	2013	5,95,000	91,500 km	Diesel	Manual
2	Ford	2017	7,75,000	36,000 km	Diesel	Manual
3	Honda	2015	4,00,000	90,000 km	Diesel	Manual
4	Maruti Suzuki	2010	2,30,000	40,000 km	Petrol	Manual
...	...	...	...	...	...	...
9995	Hyundai	2012	3,25,000	65,000 km	Petrol	Manual
9996	Maruti Suzuki	2018	2,90,000	85,000 km	CNG & Hybrids	Manual
9997	Maruti Suzuki	2010	3,20,000	72,000 km	Petrol	Manual
9998	Tata	2012	1,85,000	70,000 km	Diesel	Manual
9999	Ford	2018	8,75,000	53,764 km	Diesel	Manual

10000 rows × 6 columns

```
#Saving in a csv file
olx.to_csv('olxdata.csv')
```

DATA PRE-PROCESSING

Checking for null-values

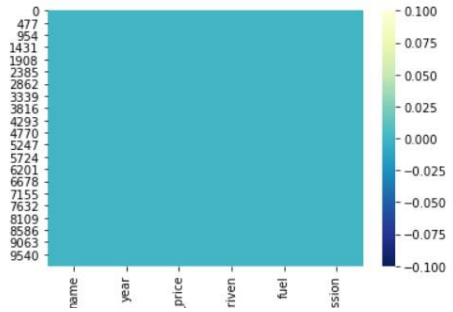
```
#Checking for missing data
df.isnull().sum()
```

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
transmission     0
dtype: int64
```

As we can see, there are no null values in the dataset and we can proceed further

```
#Plotting heatmap for missing data
sns.heatmap(df.isnull(), cmap='YlGnBu_r')
```

```
<AxesSubplot:>
```



Data Cleaning

- Removing words from object data and converting them into int data type using lambda operation.
- Removing ‘,’, replacing missing data with highest weightage data.
- Removing unnecessary features

Converting categorical data into numeric data

```
#Converting fuel and transmission data using LabelEncoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
col=['fuel','transmission']  #columns to be converted
for i in col:
    df[i] = le.fit_transform(df[i])
```

```
#Checking the dataset
df.head()
```

	name	year	selling_price	km_driven	fuel	transmission
0	Hyundai	2017	525000.0	2200	5	1
1	Hyundai	2013	595000.0	91500	2	1
2	Ford	2017	775000.0	36000	2	1
3	Honda	2015	400000.0	90000	2	1
4	Maruti Suzuki	2010	230000.0	40000	5	1

Statistical summary of the dataset

```
df.describe()    #Statistical summary of the dataset
```

	year	selling_price	km_driven	fuel	transmission
count	10000.00000	1.000000e+04	1.000000e+04	10000.00000	10000.000000
mean	2013.69860	6.608371e+05	6.914651e+04	3.30960	0.859800
std	4.02124	1.204508e+06	5.868048e+04	1.51439	0.347212
min	1983.00000	0.000000e+00	0.000000e+00	0.00000	0.000000
25%	2011.00000	2.549990e+05	3.500000e+04	2.00000	1.000000
50%	2014.00000	4.500000e+05	6.000000e+04	2.00000	1.000000
75%	2017.00000	6.770000e+05	9.000000e+04	5.00000	1.000000
max	2021.00000	6.300000e+07	2.360457e+06	5.00000	1.000000

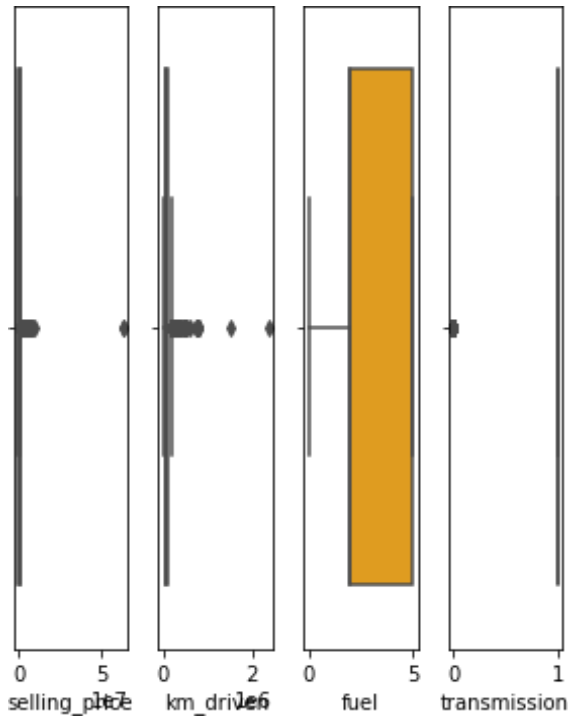
Observations:

1.There is not much difference between the mean and the median

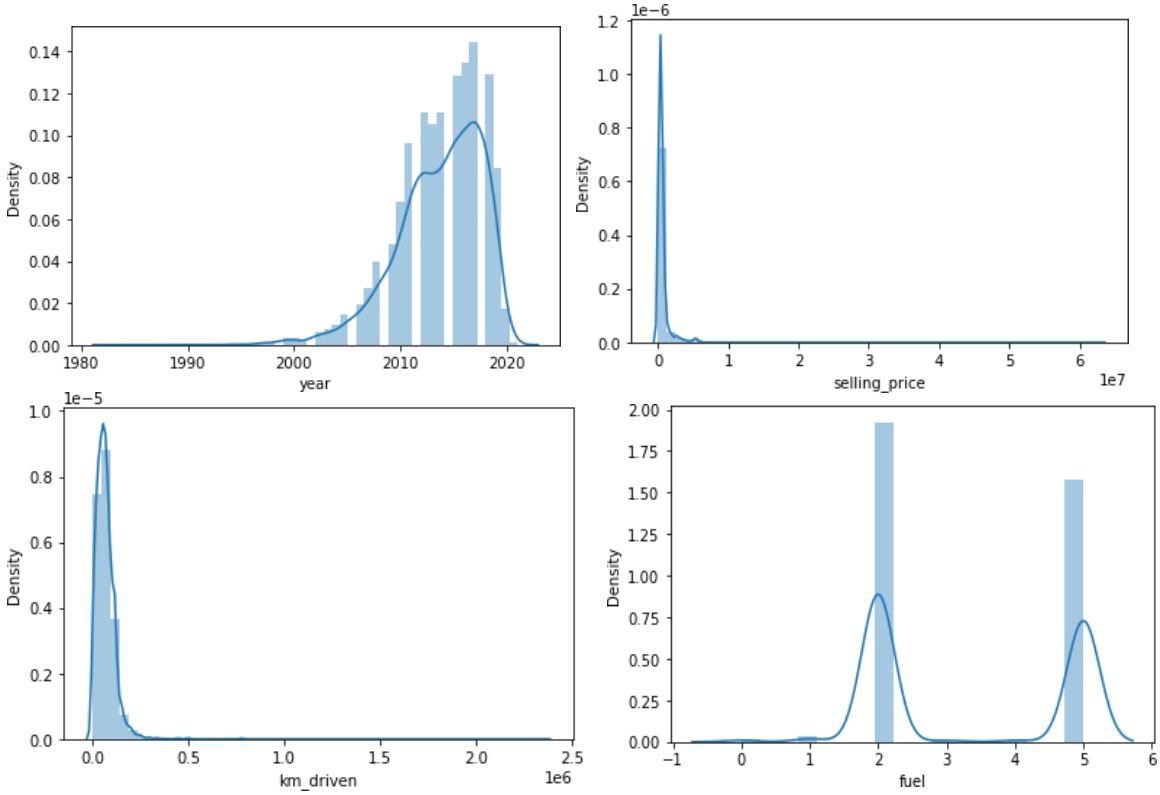
2.The minimum value is 0 for all other columns.

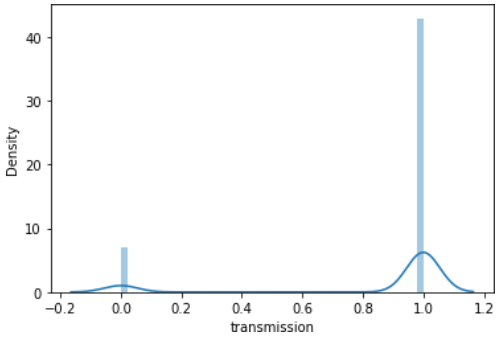
3.The 75th percentile and max value difference for km_driven column is high and it is due to the presence of outliers.

Checking outliers



Checking skewness and data distribution





Handling outliers using z-score method

```
#Removing outliers
df_new=df[(z<3).all(axis=1)]
df_new
```

	year	selling_price	km_driven	fuel	transmission
0	2017	525000.0	2200	5	1
1	2013	595000.0	91500	2	1
2	2017	775000.0	36000	2	1
3	2015	400000.0	90000	2	1
4	2010	230000.0	40000	5	1
...	...	...	...	...	...
9995	2012	325000.0	65000	5	1
9996	2018	290000.0	85000	1	1
9997	2010	320000.0	72000	5	1
9998	2012	185000.0	70000	2	1
9999	2018	875000.0	53764	2	1

9660 rows × 5 columns

```
df.shape   #Original data dimensions
```

```
(10000, 5)
```

```
df_new.shape  #New data dimensions
```

```
(9660, 5)
```

Nearly 300+ rows of data had outliers and they had been handled by using z-score method

HARDWARE AND SOFTWARE REQUIREMENTS AND TOOLS USED

For doing this project, the hardware used is a laptop with high end specification and a stable internet connection. While coming to software part, I had used anaconda navigator and in that I have used **Jupyter notebook** to do my python programming and analysis.

For using an CSV file, Microsoft excel is needed. In Jupyter notebook, I had used lots of python libraries to carry out this project and I have mentioned below with proper justification:

Libraries Used:

Importing required libraries

```
#Basic libraries
import pandas as pd
import numpy as np

#Visualization libraries
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#Importing required metrices and model for the dataset
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
from sklearn.linear_model import LinearRegression,Lasso,ElasticNet,Ridge
from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from sklearn.model_selection import GridSearchCV

#Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

MODEL/S DEVELOPMENT AND EVALUATION

Preparing dataset for model training

```
df_x=df_new.drop('selling_price',axis=1)   #Independent variables
y=df_new['selling_price']    #Dependent or Target Variable
```

```
#Checking x data
df_x.head()
```

	year	km_driven	fuel	transmission
0	2017	2200	5	1
1	2013	91500	2	1
2	2017	36000	2	1
3	2015	90000	2	1
4	2010	40000	5	1

```
#Checking y data after splitting
y.head()
```

```
0    525000.0
1    595000.0
2    775000.0
3    400000.0
4    230000.0
Name: selling_price, dtype: float64
```

Treating skewness and scaling the data

```
#We are treating skewness by using square root transform
for col in df_x.skew().index:
    if col in df_x.describe().columns:
        if df_x[col].skew()>0.55:
            df_x[col]=np.sqrt(df_x[col])
        if df_x[col].skew()<-0.55:
            df_x[col]=np.sqrt(df_x[col])
```

```
df_x.skew()   #Checking skewness after treating it
```

```
year           -0.656801
km_driven      -0.173734
fuel            0.156582
transmission   -2.214140
dtype: float64
```

We can see that skewness has been treated and we can proceed further model building process

Scaling the data

Sometimes model can be biased to higher values in dataset, so it is better to scale the dataset so that we can bring all the columns in common range. We can use StandardScaler here.

```
#Scaling the dataset using StandardScaler
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(df_x)
x=pd.DataFrame(x,columns=df_x.columns)
```

Model Building

Finding best random state and r2_score

```
#Finding the best random state and r2_score
for i in range(42,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=i)
    lr=LinearRegression()
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    if round(r2_score(y_train,pred_train)*100,1)==round(r2_score(y_test,pred_test)*100,1):
        print('At random state',i,',the model performs well')
        print('Training r2_score is: ',r2_score(y_train,pred_train)*100)
        print('Testing r2_score is: ',r2_score(y_test,pred_test)*100)
```

```
At random state 48 ,the model performs well
Training r2_score is:  49.815054894606924
Testing r2_score is:  49.78050062810894
At random state 73 ,the model performs well
Training r2_score is:  49.81662674234254
Testing r2_score is:  49.75466781460837
```

We can see that at random_state=48, the best r2_score is obtained so that we can create our train_test_split with this random state.

```
#Creating train_test_split using best random_state
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=48,test_size=.20)
```

After initializing the instances of the model, we will run a for loop with the required metrics and other regression related scores and we will append the obtained values to empty lists. The code will be as follows:

```
#Finding the required metrices for all models together using a for loop
Model=[]
score=[]
cvs=[]
sd=[]
mae=[]
mse=[]
rmse=[]
for name,model in models:
    print('******************************',name,'**************************')
    print('\n')
    Model.append(name)
    model.fit(x_train,y_train)
    print(model)
    pre=model.predict(x_test)
    print('\n')
    AS=r2_score(y_test,pre)
    print('r2_score: ',AS)
    score.append(AS*100)
    print('\n')
    sc=cross_val_score(model,x,y,cv=5,scoring='r2').mean()
    print('cross_val_score: ',sc)
    cvs.append(sc*100)
    print('\n')
    std=cross_val_score(model,x,y,cv=5,scoring='r2').std()
    print('Standard Deviation: ',std)
    sd.append(std)
    print('\n')
    MAE=mean_absolute_error(y_test,pre)
    print('Mean Absolute Error: ',MAE)
    mae.append(MAE)
    print('\n')
    MSE=mean_squared_error(y_test,pre)
    print('Mean Squared Error: ',MSE)
    mse.append(MSE)
    print('\n')
    RMSE=np.sqrt(mean_squared_error(y_test,pre))
    print('Root Mean Squared Error: ',RMSE)
```

The final outputs of the models we used are stored in a dataframe and it is shown below:

```
#Finalizing the result
result=pd.DataFrame({'Model':Model, 'r2_score': score, 'Cross_val_score':cvs, 'Standard_deviation':sd,
                     'Mean_absolute_error':mae, 'Mean_squared_error':mse, 'Root_Mean_Squared_error':rmse})
result
```

	Model	r2_score	Cross_val_score	Standard_deviation	Mean_absolute_error	Mean_squared_error	Root_Mean_Squared_error
0	Linear Regression	49.780501	49.634395	0.016118	245143.393316	1.596549e+11	399568.420291
1	Lasso Regression	49.780509	49.634394	0.016118	245142.909691	1.596549e+11	399568.385978
2	Elastic Net Regression	45.587343	45.318972	0.015353	234774.440477	1.729856e+11	415915.334760
3	Ridge Regression	49.780536	49.634433	0.016118	245134.259261	1.596548e+11	399568.281071
4	Decision Tree Regressor	74.373282	74.108579	0.073529	143620.442342	8.147098e+10	285431.209631
5	KNeighbors Regressor	74.761909	71.831772	0.037407	152245.250104	8.023548e+10	283258.675692
6	RandomForestRegressor	77.578361	77.119771	0.053823	143078.180636	7.128157e+10	266986.094827
7	AdaBoostRegressor	52.939289	50.707491	0.051023	247539.505973	1.496127e+11	386797.992642
8	GradientBoostingRegressor	70.246173	68.762676	0.015560	179608.295180	9.459164e+10	307557.542669

We can see that Random Forest Regressor and KNeighbors Regressor are performing well compared to other algorithms. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase the scores.

Hyperparameter Tuning

Random Forest Regressor

```
#Creating parameter list to pass in GridSearchCV
parameters={'criterion':['mse','mae'],'n_estimators':[50,100,500],'max_features':['auto','sqrt','log2']}
```

```
#Using GridSearchCV to run the parameters and checking final accuracy
rf=RandomForestRegressor()
grid=GridSearchCV(rf,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)  #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'criterion': 'mse', 'max_features': 'auto', 'n_estimators': 100}
0.7585194316770476
```

```
#Using the best parameters obtained
RF=RandomForestRegressor(random_state=48, n_estimators=100, criterion='mse', max_features='auto')
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(RF,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(RF,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2_score:  78.02640255481647
Cross validation score:  77.0128999250369
Standard deviation:  0.0525850724267253


Mean absolute error:  142169.8222155508
Mean squared error:  69857187763.80247
Root Mean squared error:  264305.10355231975
```

KNeighbors Regressor

```
#Creating parameter list to pass in GridSearchCV
parameters={'n_neighbors':(1,10),'weights':['uniform','distance'],'algorithm':['auto','ball_tree','brute','kd_tree']}
```

```
#Using GridSearchCV to run the parameters and checking final r2_score
from sklearn.model_selection import GridSearchCV
knr=KNeighborsRegressor()
grid=GridSearchCV(knr,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)  #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'algorithm': 'brute', 'n_neighbors': 10, 'weights': 'distance'}
0.7453869635463433
```

```
#Using the best parameters obtained
knr=KNeighborsRegressor(algorithm='brute', n_neighbors=10, weights='distance')
knr.fit(x_train,y_train)
pred=knr.predict(x_test)
print('Final r2_score after tuning is: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(knr,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(knr,x,y,cv=5,scoring='r2').std())
print('\n')
print('Mean absolute error: ',mean_absolute_error(y_test,pred))
print('Mean squared error: ',mean_squared_error(y_test,pred))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Final r2_score after tuning is:  77.66397076547491
Cross validation score:  76.63929919814726
Standard deviation:  0.05760230307407942


Mean absolute error:  139198.6305152985
Mean squared error:  71009409907.79892
Root Mean squared error:  266475.90868181485
```

After applying Hyperparameter Tuning, we can see that RandomForestRegressor is the best performing algorithm among all other algorithms. It has also the less amount of error values obtained. Lesser the RMSE score, the better the model. Now we will finalize the model. The scores are improved;

FINAL THE MODEL

Comparing original and predicted price

```
rf_prediction=RF.predict(x)
print('Predictions of Random Forest Regressor: ',rf_prediction)
```

```
Predictions of Random Forest Regressor:  [523000.         644522.88752914 760124.50877554 ... 320000.
 276868.90269236 896053.87848263]
```

```
#Comparing actual and predicted values with the help of a dataframe
predictions=pd.DataFrame({'Original_price':y, 'Predicted_price':rf_prediction})
predictions
```

	Original_price	Predicted_price
0	525000.0	523000.000000
1	595000.0	644522.887529
2	775000.0	760124.508776
3	400000.0	592402.068286
4	230000.0	211955.678041
...	...	...
9995	325000.0	269123.641775
9996	290000.0	305584.444444
9997	320000.0	320000.000000
9998	185000.0	276868.902692
9999	875000.0	896053.878483

9660 rows × 2 columns

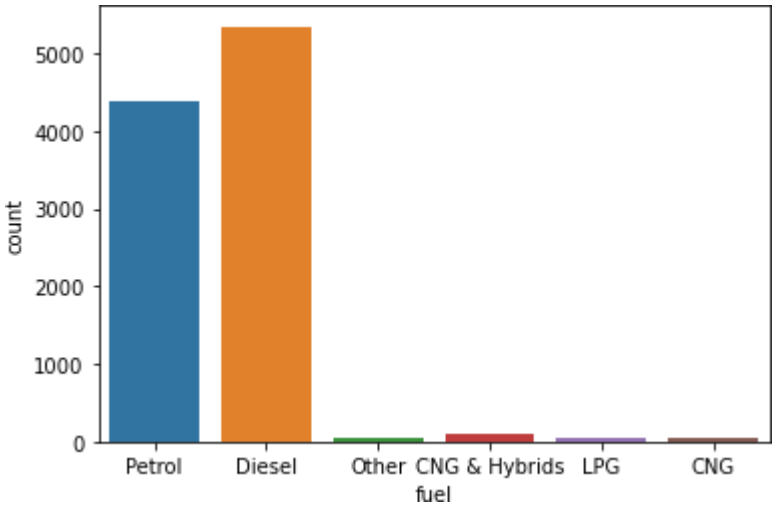
Saving the model

```
#Saving the model
import pickle
filename='Car_Price_Project.pkl'   #Specifying the filename
pickle.dump(RF,open(filename,'wb'))
```

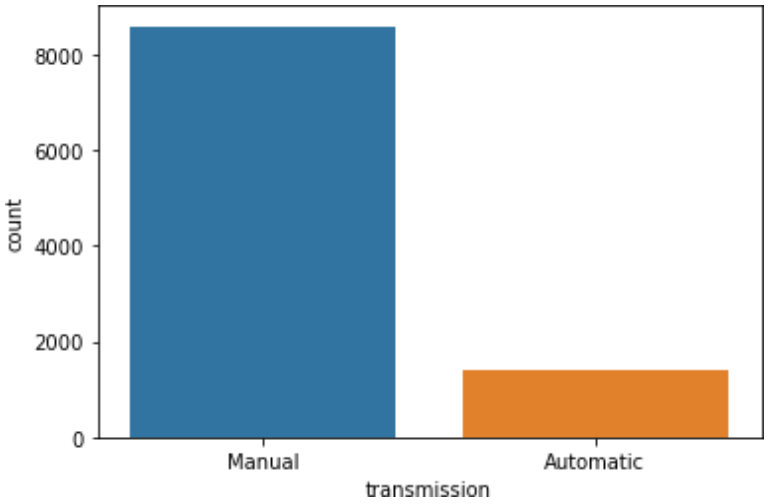
```
#Saving the predicted values
results=pd.DataFrame(rf_prediction)
results.to_csv('Car_Price_Prediction_Results.csv')
```

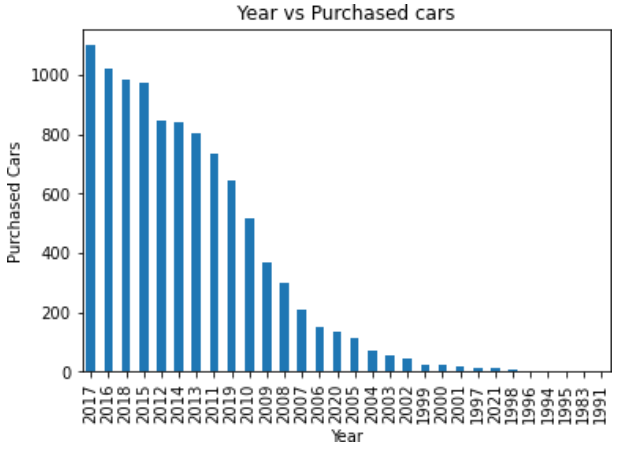
DATA VISUALIZATION

Fuel count



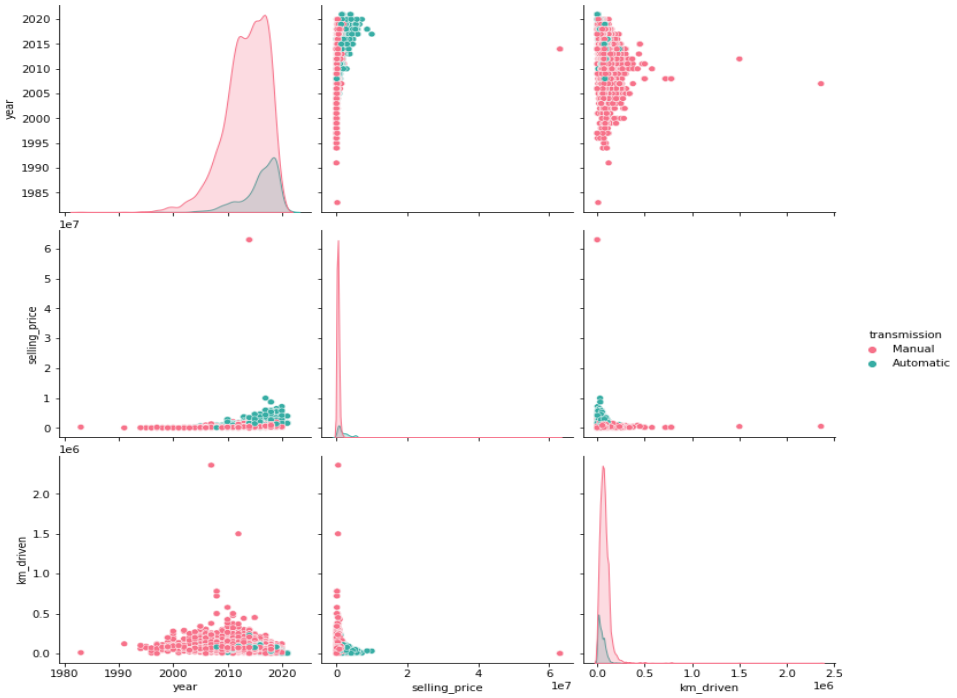
Transmission count



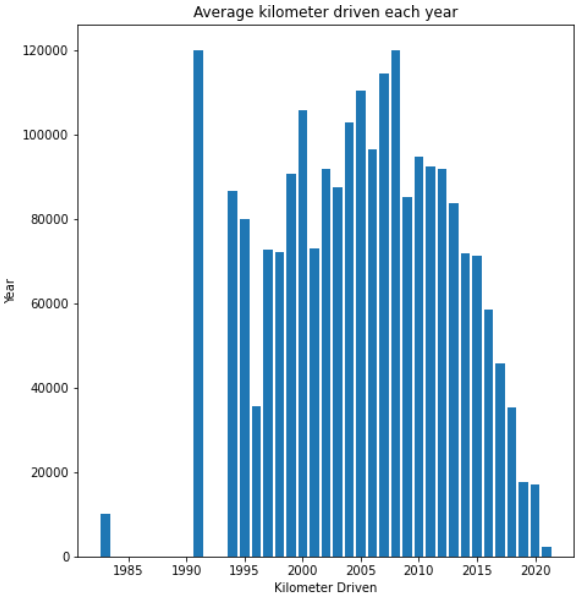


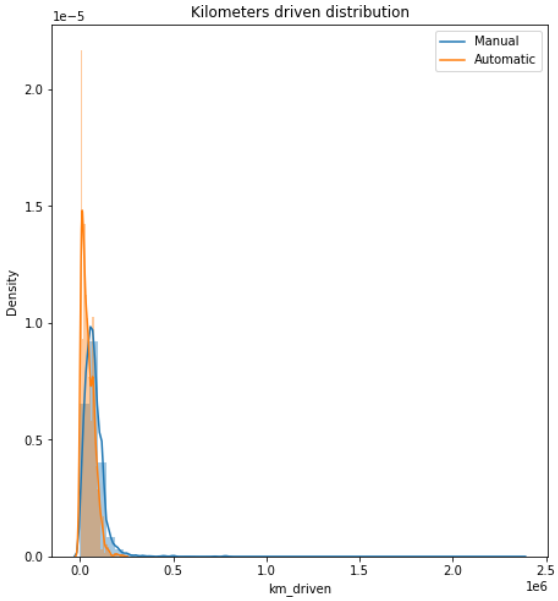
Year vs Purchased cars

Pairplot for transmission

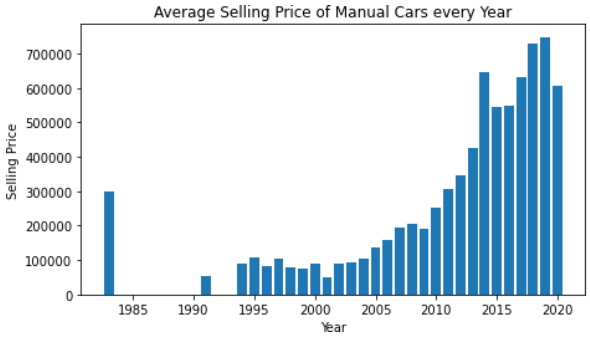


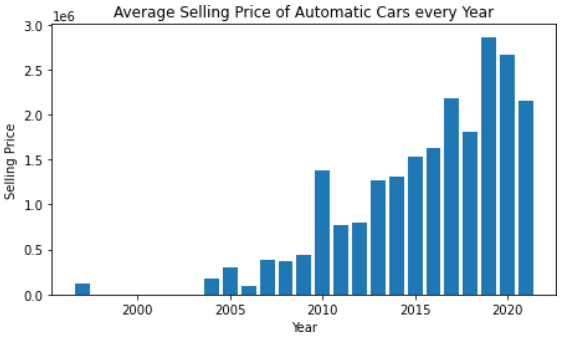
Year vs km



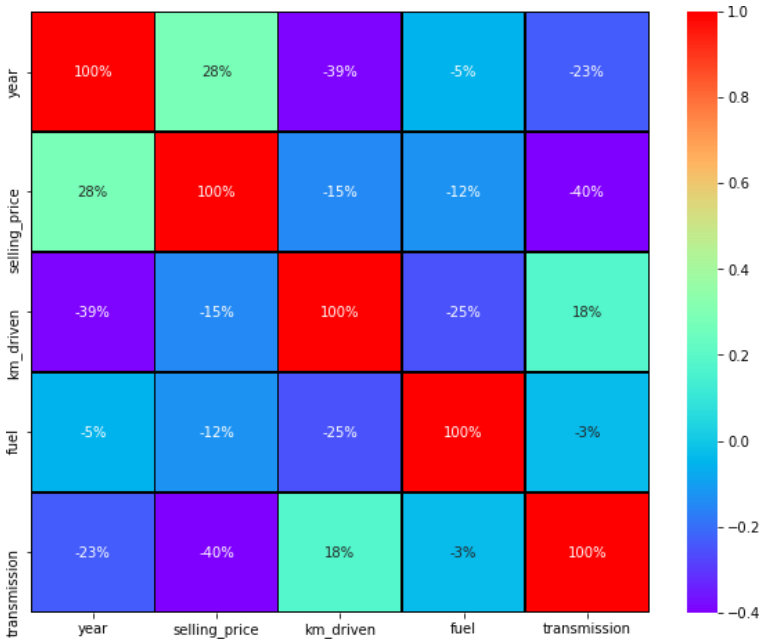


Year vs Selling price

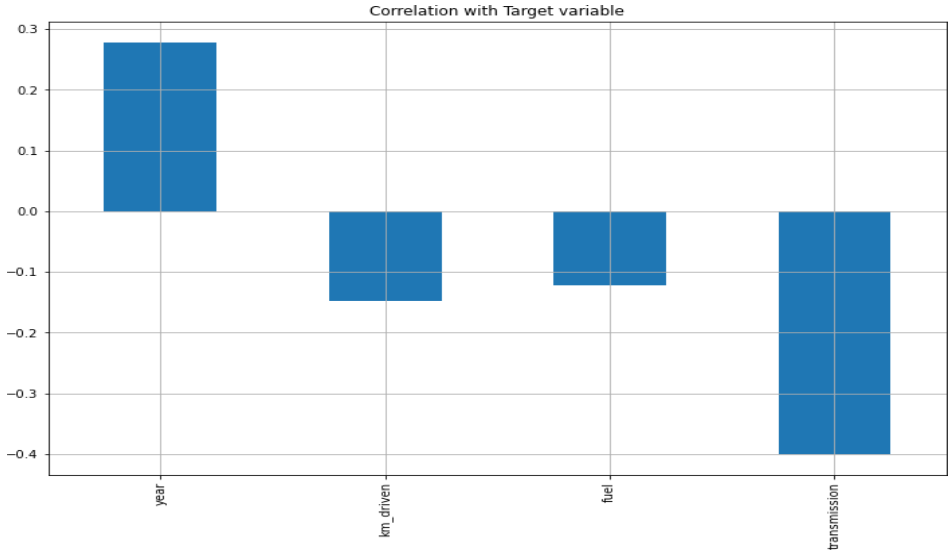




Correlation heatmap



Correlation with target variable



CONCLUSION

Key Findings and Conclusions of the Study

-> After the completion of this project, we got an insight of how to collect data, pre-processing the data, analysing the data and building a model.

-> First, we collected the used cars data from different websites like olx, cardekho and it was done by using Web scraping. The framework used for web scraping was Selenium, which has an advantage of automating our process of collecting data.

-> We collected almost 10000 of data which contained the selling price and other related features.

-> Then, the scrapped data was combined in a single data frame and saved in a csv file so that we can open it and analyse the data.

-> We did data cleaning, data-preprocessing steps like finding and handling null values, removing words from numbers, converting object to int type, data visualization, handling outliers, etc.

-> After separating our train and test data, we started running different machine learning classification algorithms to find out the best performing model.

-> We found that RandomForest and KNeighbors Algorithms were performing well, according to their r2_score and cross val scores.

-> Then, we performed Hyperparameter Tuning techniques using GridSearchCV for getting the best parameters and improving the scores. In that, RandomForestRegressor performed well and we finalised that model.

-> We saved the model in pkl format and then saved the predicted values in a csv format.

The problems we faced during this project were:

- ➔Website was poorly designed because the scrapping took a lot of time and there were many issues in accessing to next page.
- ➔More negative correlated data were present.
- ➔Many outliers were removed as some of the selling price and km values were out of range.
- ➔No information for handling these fast-paced websites were informed so that we were consuming more time in web scraping itself.