CAR PRICE PREDICTION PROJECT

SUBMITTED BY:
ATISH KALANGUTKAR

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Shwetank Mishra as well as Flip Robo Technologies who gave me the opportunity to do this project on Car Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things.

References:

I have also used few external resources that helped me to complete this project successfully. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. https://github.com/
4. https://www.analyticsvidhya.com/
5. www.researchgate.net/

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.

Conceptual Background of the Domain Problem:

Used car price prediction is a thing which is very hard to guess as there are many factors which influence the price of the car in the second-hand market. As the prices changes according to the brand, model, fuel type, km, gear type and the age of the car. All this factors plays an important role in deciding the price of the used car. Beside this will have to look upon the car history also whether the car has been in the accident or not, whether there any external damage or not, have to check the working condition of the car. And this days mos t of the people are going for automatic cars and electric cars.

So here we are trying to help the client who sell used cars to understand the price of the used cars by deploying machine learning models. These models would help the client/sellers to understand the used car market and accordingly they would be able to sell the used car in the market.

Review of literature:

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict the Used car prices in the market. In this we have discussed about various applications and methods which inspired us to build our supervised Machine Learning techniques to predict the price of the used cars. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of data information by doing web scraping from www.cars24.com website which is a web platform where customers can buy used cars.

This project is more about data exploration, feature engineering and pre processing that can be done on this data. As we have scrapped the data that includes car related features, we can do better data explorations and derive some interesting features using the available columns.

Different techniques like ensemble techniques have been used to make the predictions.

Motivation for the Problem Undertaken:

Deciding whether a used car is worth the posted price when you see any car on a particular website. Several factors such as Brand, Model, Fuel type, Gear type, kms driven, age of the car plays an important role in deciding the worth of the used car.

So i was more interested so that we can help the sellers as well as the clients to decide the worth of the car based on the above factors.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling of the Problem:

We need to develop an efficient and effective machine learning model which predicts the price of the used cars. So 'Price' is our target variable which is continuous in nature. So it is a Regression problem where we need to use regression algorithms to predict the results.This project is done on three phases:

Data Collection Phase: I have done web scraping to collect the data of used cars from the website www.cars24.com. As per the requirement we need to build the model to predict the prices of the used cars.

Data Analysis: After cleaning the data I have done some analysis on the data by using different types visualizations.

Model Building Phase: After collecting the data, I built a machine learning model. Before that, have done all data per-processing steps. The complete life cycle of data science that I have used in this project are as follows:

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-Processing
- Model Building
- Model Evaluation
- Selecting the best model

Data Sources and their formats:

We have collected the dataset from the website www.cars24.com. The data is scrapped using web scraping technique and the framework used is selenium. We have scraped about 6034 rows and 9 columns for different locations including target variable 'Price'. The dataset contains both categorical and numerical data type. The data description is as follows:

Variables	Definition
Brand	Name of the cars with manufacturing year
Model	Model or name of the car
Year	Age of the car
Transmission	Type of gear transmission used in car
Kms Driven	Car running in kms till the date
Owner	Number of owners of the car
Fuel Type	Type of fuel used for car engine
Location	Location/place of the car
Price	Price of the Car

Data Pre-Processing Done:

Data pre-processing is the process of converting raw data into a well readable format to be used by machine learning model. I have used following pre-processing steps:

- Importing required libraries and loading the dataset.
- Checked some statistical information such as shape of the dataset,unique values and number of unique values present, type of data present in the columns etc.

- Checked for null values and found that there were about 553 null values present in the column Transmission. Further dropped the null values from the dataset.
- Also there was one unwanted column named Unnamed:0 which was dropped as it was just having numerical values which was irrelevant at the time of prediction.
- Done feature engineering on target variable that is 'Price' and also on features that is 'kms driven' as it was having commas in between the numbers and rupees sign was there, because of that it was showing data type as object. So replaced them by empty spaces and converted the data type to integer.
- Also done feature engineering on column Model in which square brackets and single quotation was present. So replaced them by empty spaces.
- While checking for duplicates it was found that there were 18 duplicates present in the dataset. Further it was dropped from the dataset.
- While checking for unique values of column Transmission it was found that it was having unique values Manual, Automatic and MAnual. As Manual and MAnual are same. So replaced MAnual with Manual.
- While checking for unique values of column owners it was found that it was having values 1st owner, 2nd owner and 3rd owner. As it was an string data type column replaced 1st with First, 2nd with Second and 3rd with Third.
- Than separated categorical and numeric features according to the data types.

- Performed uni-variate, bi-variate and multi-variate analysis by plotting Dist plot, Count plot, Pie plot, Bar plot and Line plot.
- Than checked for Top 50 highest priced cars from the dataset.
- Encoded the categorical features by using Label encoded method.
- Used bar plot to check the correlation between the features and the label and after with the help of heatmap checked whether there is multicollinearity present or not among the features. So found that there was no multicollinearity problem within the features.
- Checked whether there are any outliers present or not using box plot on numerical data columns and found that in column year and kms driven was having outliers so by using z-score method removed outliers.
- Checked for skewness on numerical data columns and found that column year was having skewness, so by using power transform method treated the skewness of that particular columns.
- Separated feature and label data and than feature scaling was done by using power transform method to avoid any kind of data biasness.
- After plotting heatmap, it was seen than there was no multicollinearity problem within the features, but just to cross verify vif method was used. After using vif method it was found that all the columns values were below 5. So it was confirmed that there was no multicollinearity problem within the features.

Data Inputs-Logic-Output Relationships:

The dataset consists of label and features. The features are independent and label is dependent.

I have checked the correlation between the label and the features using heatmap and bar plot in which both positive and negative correlation between the label and features were there.

After that different data mining model were designed to predict the prices of the used cars. In this 4 regression models were used namely, XGB, Gradient Boosting, Ada Boost, Random forest, Decision Tree and KNeighbors.

Hardware and Software Requirements and Tools Used:

To build a machine learning projects it is important to have the following hardware and software requirements tools.

Hardware Required:

- RAM: 8GB
- CPU: AMD RYZEN 5 4600H
- GPU: AMD Radeon ™ Vega 8 Graphics and NVIDIA GeForce GTX 1650 Ti

Software Required:

- Distribution: Anaconda Navigator
- Programming Language: Python
- Browser based language shell: Jupyter Notebook
- Chrome: To scrap the data

Libraries Required:

Pre-processing and Visualization

```
#importing required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

- Encoding

```
#Importing required libraries
from sklearn.preprocessing import LabelEncoder
```

- To remove outliers

```
#Importing required libraries
from scipy.stats import zscore
```

- To remove skewness

```
#Importing required libraries
from sklearn.preprocessing import power_transform
```

- To standardize the data

```
#Standardiziing the data
#Importing require libraries
from sklearn.preprocessing import StandardScaler
```

- Vif method

```
#Importing require libraries
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

- Evaluation metrics and machine learning Algorithms

```
#Importing required libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost as xgb
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

MODEL/S DEVELOPMENT AND EVALUATION

Identification of Possible Problem-solving Approaches (Methods):

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of features and label.

Encoded data using label encoder, removed outliers and and skewness using z-score method and power transform method. Scaled the data using standard scaler. Also, before building the model, I made sure that input data is cleaned and scaled.

Than checked for the best random state to be used on our regression machine learning model pertaining to the feature importance details. And than created multiple regression models along with evaluation metrics. In this data 'Price' is our target variable which is continuous in nature so this is a regression problem. So I have used 6 regression algorithms and predicted the prices of the used cars.

By going through each and very aspect of the machine learning models I have selected Xtreme Gradient Boosting(XGB) as best suitable algorithm to create our final model as it is giving highest testing score(R2 SCORE) and less root mean squared error(RMSE) among all the algorithms used. Performed hyper parameter tuning on best model and at last saved the final model.

Testing of Identified Approaches(Algorithms):

Since 'Price' is my target variable which is continuous in nature, so that we came to know that it is regression type problem and I have used following regression models, they are:

1. Xtreme Gradient Boosting Regressor (XGB)
2. Gradient Boosting Regressor
3. Ada Boost Regressor
4. Random Forest Regressor
5. Decision Tree Regressor
6. KNeighbors Regressor

Run and Evaluate Selected Models:

I have used 6 regression algorithms after choosing random state amongst 0-200 number. I have used XGB Regressor to find the best random forest state and code is as below:

```
xgb=xgb.XGBRegressor()
```

```
#using range fucntion to find the best random state using xgb regressor
for i in range(0,200):
    x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=i)
    xgb.fit(x_train,y_train)
    pred=xgb.predict(x_train)
    y_pred=xgb.predict(x_test)
    print(f'at random state {i}, training accuracy is {metrics.r2_score(y_train,pred)*100}')
    print(f'at random state {i}, testing accuracy is {metrics.r2_score(y_test,y_pred)*100}')
    print('\n')
```

Maximum testing score (R2 score) is 94.49% on random state 149.

Model Building:

Xtreme Gradient Boosting

```
#since random state  has highest testing score,so taking random state as 149
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=149)
```

```
#Training the Model
xgb.fit(x_train,y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=xgb.predict(x_train)
y_pred=xgb.predict(x_test)
```

```
#Printing the training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 98.84929815934103

 Testing Score: 94.49732052564546
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for xgb regressor
for j in range(2,6):
    cv_score=cross_val_score(xgb,x_scaler,y,cv=j)
    cv_mean=cv_score.mean()
    print(f'at cross fold {j} the cv score is {cv_mean}and accuracy for the testing is {test_accuracy}')
    print('\n')
```

```
at cross fold 2 the cv score is 0.9044273442843995and accuracy for the testing is 0.9449732052564547

at cross fold 3 the cv score is 0.9049160353052571and accuracy for the testing is 0.9449732052564547

at cross fold 4 the cv score is 0.9084054670358213and accuracy for the testing is 0.9449732052564547

at cross fold 5 the cv score is 0.9079656846215315and accuracy for the testing is 0.9449732052564547
```

taking cv=4

```
#mean absolute error
mean_absolute_error(y_test,y_pred)
```

```
48583.36346453446
```

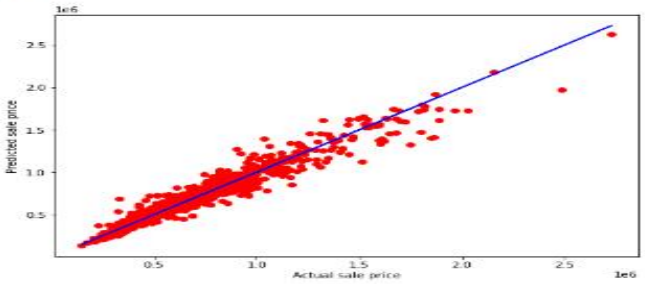
```
#mean squared error
mean_squared_error(y_test,y_pred)
```

```
5569222494.054324
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
74627.22354512678
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual sale price',fontsize=10)
plt.ylabel('Predicted sale price',fontsize=10)
plt.show()
```



- Created XGB Regressor model and checked for its evaluation metrics. The model is giving testing score as 94.49%.
- From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Gradient Boosting

```
#Instantiating Gradient Boosting
gb=GradientBoostingRegressor()
```

```
#since random state  has highest testing score,so taking random state as 249
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=149)
```

```
#Training the Model
gb.fit(x_train,y_train)
```

```
GradientBoostingRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=gb.predict(x_train)
y_pred=gb.predict(x_test)
```

```
#Printing Training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 84.73012129348727

 Testing score: 85.28219512805292
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for gradient boosting
cv_score=cross_val_score(gb,x_scaler,y,cv=4).mean()
cv_score
```

```
0.8156993113748674
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
81681.88410727576
```

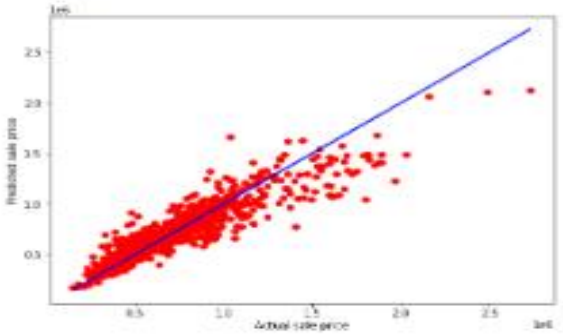
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
14976752351.206367
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
122379.54220868113
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual sale price',fontsize=10)
plt.ylabel('Predicted sale price',fontsize=10)
plt.show()
```



- Created Gradient Boosting Regressor model and checked for its evaluation metrics. The model is giving testing score as 85.28%.
- From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Ada Boost

```
#Instantiating Ada Boost Regressor
ada=AdaBoostRegressor()
```

```
#since random state  has highest testing score,so taking random state as 140
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=140)
```

```
#Training the data
ada.fit(x_train,y_train)
```

```
AdaBoostRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=ada.predict(x_train)
y_pred=ada.predict(x_test)
```

```
#Printing Training and Testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 36.623427615601

 Testing Score: 32.99639497839249
```

```
train_accuracy=metrics.r2_score(y_train,pred)
test_accuracy=metrics.r2_score(y_test,y_pred)
```

```
#Checking cross validation score for Ada boost
cv_score=cross_val_score(ada,x_scaler,y,cv=4).mean()
cv_score
```

```
0.33473999620989863
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
324833.21533175858
```

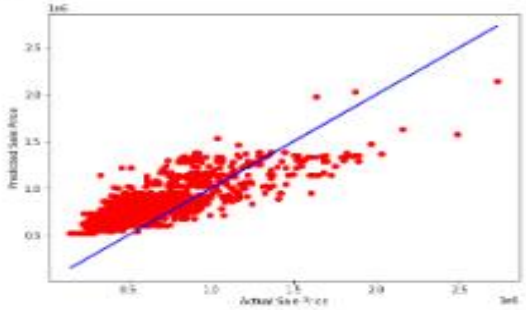
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
67813868863.81226
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
260430.96348782267
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



- Created Ada Boost Regressor model and checked for its evaluation metrics. The model is giving testing score as 32.99%.
- From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Random Forest

```
#Instantiating Random forest regressor
rf=RandomForestRegressor()
```

```
#since random state  has highest testing score,so taking random state as 149
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=149)
```

```
#Training data
rf.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=rf.predict(x_train)
y_pred=rf.predict(x_test)
```

```
#Printing Training and Testing Score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 98.61807385508776

 Testing Score: 91.81510279297117
```

```
#Checking cross validation score for Random forest
cv_score=cross_val_score(rf,x_scaler,y,cv=4).mean()
cv_score
```

```
0.8820331869500468
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
51404.231549364624
```

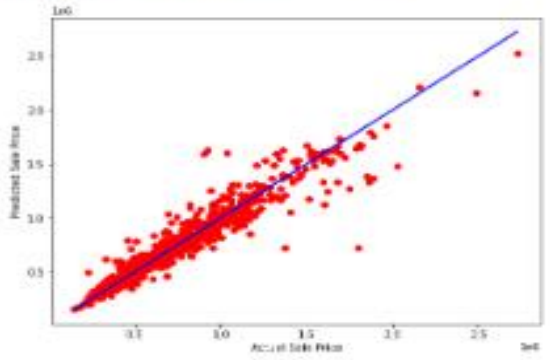
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
8283875855.272178
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
91015.80003894066
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



- Created Random Regressor model and checked for its evaluation metrics. The model is giving testing score as 91.81%.
- From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Decision Tree

```
#Instantiating Decision Tree regressor
dt=DecisionTreeRegressor()
```

```
#since random state  has Highest testing score,so taking random state as 149
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=149)
```

```
#Training data
dt.fit(x_train,y_train)
```

```
DecisionTreeRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=dt.predict(x_train)
y_pred=dt.predict(x_test)
```

```
#Printing Training and Testing Score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 99.99999715473129

 Testing Score: 83.79043867688837
```

```
#Checking cross validation score for Random forest
cv_score=cross_val_score(dt,x_scaler,y,cv=4).mean()
cv_score
```

```
0.8004509433374896
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
59638.29901282346
```

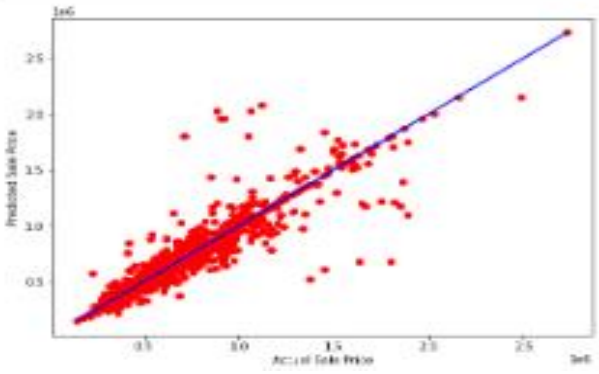
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
16405599164.232874
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
128084.34394656855
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



- Created Decision Tree Regressor model and checked for its evaluation metrics. The model is giving testing score as 83.79%.
- From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

KNeighbors Regressor

```
#Instantiating KNeighbors regressor
knn=KNeighborsRegressor()
```

```
#since random state  has highest testing score,so taking random state as 149
x_train,x_test,y_train,y_test=train_test_split(x_scaler,y,test_size=0.25,random_state=149)
```

```
#Training data
knn.fit(x_train,y_train)
```

```
KNeighborsRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
pred=knn.predict(x_train)
y_pred=knn.predict(x_test)
```

```
#Printing Training and Testing Score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 76.08093872753636

 Testing Score: 65.77767550005349
```

```
#Checking cross validation score for Random forest
cv_score=cross_val_score(knn,x_scaler,y,cv=5).mean()
cv_score
```

```
0.5780473410547444
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
126813.00586510203
```

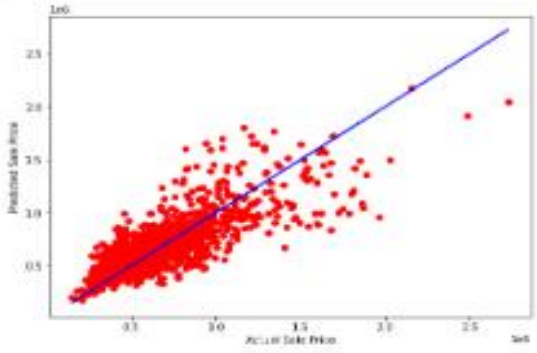
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
34036169577.41936
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
184587.95140836772
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Sale Price',fontsize=10)
plt.ylabel('Predicted Sale Price',fontsize=10)
plt.show()
```



Created KNeighbors Regressor model and checked for its evaluation metrics. The model is giving testing score as 65.77%.

From the graph we can observe how our model is mapping and also we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

Model Selection:

From the above created models, we can conclude that 'Xtreme Gradient Boosting Regressor' as the best fitting model as it is giving high testing score(R2 SCORE) and low MAE, MSE and RMSE values. Then we used hyper parameter tuning to tuned the model and increase our model score.

Hyper Parameter Tuning:

```
#Using Grid Search cv for hyperparameter tuning for XGB
from sklearn.model_selection import GridSearchCV
```

```
param_grid={{
    'n_estimators': [100,200],
    'max_depth' : [2,3,4],
    'eta':[0.3,0.1,0.01],
    'subsample':[0.1,0.2,0.3],
    'colsample_bytree' :[0.4,0.5,0.6]
}}
```

```
grid_search=GridSearchCV(xgb,param_grid=param_grid,cv=5)
```

```
#Training
grid_search.fit(x_train,y_train)
```

```
GridSearchCV(cv=5,
             estimator=XGBRegressor(base_score=0.5, booster='gbtree',
                                    callbacks=None, colsample_bylevel=1,
                                    colsample_bynode=1, colsample_bytree=1,
                                    early_stopping_rounds=None,
                                    enable_categorical=False, eval_metric=None,
                                    gamma=0, gpu_id=-1, grow_policy='depthwise',
                                    importance_type=None,
                                    interaction_constraints='',
                                    learning_rate=0.300000012, max_bin=256,
                                    max_cat...ot=4, max_delta_step=0,
                                    max_depth=6, max_leaves=0,
                                    min_child_weight=1, missing=nan,
                                    monotone_constraints='()', n_estimators=100,
                                    n_jobs=0, num_parallel_tree=1,
                                    predictor='auto', random_state=0,
                                    reg_alpha=0, reg_lambda=1, ...),
             param_grid={'colsample_bytree': [0.4, 0.5, 0.6],
                         'eta': [0.3, 0.1, 0.01], 'max_depth': [2, 3, 4],
                         'n_estimators': [100, 200],
                         'subsample': [0.1, 0.2, 0.3]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
#Printing the training and testing score
print('\n Training Score:',metrics.r2_score(y_train,pred)*100)
print('\n Testing Score:',metrics.r2_score(y_test,y_pred)*100)
```

```
 Training Score: 76.08093872753036

 Testing Score: 65.77767550865349
```

```
#Checking cross validation score for XGBoost
cv_score=cross_val_score(xgbl,x_scaler,y,cv=4).mean()
cv_score
```

```
0.8986975088882899
```

```
#Mean Absolute Error
mean_absolute_error(y_test,y_pred)
```

```
126813.08586538263
```

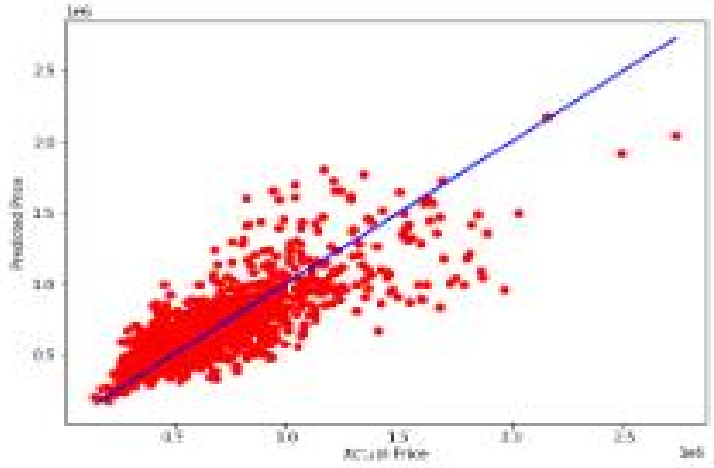
```
#Mean Squared Error
mean_squared_error(y_test,y_pred)
```

```
34036109577.41936
```

```
#Root mean squared error
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
184487.95140836772
```

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=y_pred,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Price',fontsize=10)
plt.ylabel('Predicted Price',fontsize=10)
plt.show()
```



After performing hyper parameter tuning using best parameters of XGB, the testing scores ,the cross validation score reduced. So saving the model on which hyper parameter tuning was not done.

Saving the model:

```
#saving the model
#Importing required libraries
import pickle
pickle.dump(xgb,open('Car Price Prediction Project','wb'))
```

Loading the saved model and predicting the price of the used car:

```
#Loading the saved model
model=pickle.load(open('Car Price Prediction Project','rb'))
```

```
#Prediction
prediction=model.predict(x_test)
prediction
```

```
array([ 636453.6 , 1726238.9 ,  488433.38, ...,  423070.4 ,  555018.1 ,
       1743999.6 ], dtype=float32)
```

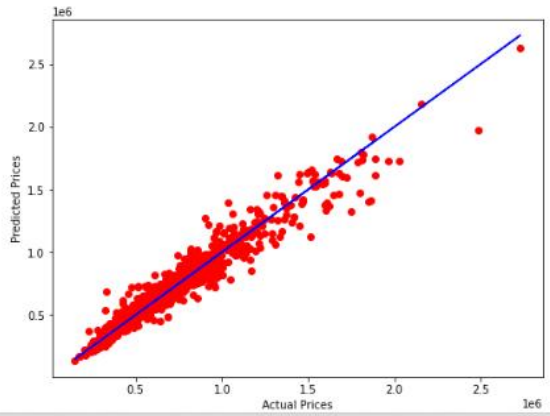
```
#This are the predicted prices of the flight tickets
Predicted_car_Price=pd.DataFrame([model.predict(x_test)[:],y_test[:]],index=['Predicted','Actual'])
Predicted_car_Price
```

	0	1	2	3	4	5	6	7	8	9	...	1354	
Predicted	636453.625	1726238.875	488433.375	718083.3125	336820.96875	686304.75	575406.8125	385789.625	1396313.0	214457.59375	...	649380.1875	752
Actual	604000.000	1963000.000	487000.000	593000.0000	396000.00000	688000.00	496000.0000	404000.000	1607000.0	218000.00000	...	539000.0000	853

2 rows × 1364 columns

Using regression model, we have got the predicted price of the car.From the above output we can observe that predicted values are almost near to the actual values.

```
#Plotting the best fit line
plt.figure(figsize=(8,6))
plt.scatter(x=y_test,y=prediction,color='red')
plt.plot(y_test,y_test,color='blue')
plt.xlabel('Actual Prices',fontsize=10)
plt.ylabel('Predicted Prices',fontsize=10)
plt.show()
```



The graph shows how our model is mapping and the plot gives the linear relation between predicted and actual price of the used cars. The blue line is the best fitting line which gives us the actual values/data and red dots gives us the predicted values/data.

Key Metrics for success in solving problem under Consideration:

The key metrics used here were r2_score, cross_val_score, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyper parameter Tuning and we will be using the GridSearchCV method.

1. Cross Validation: Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of the full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during the process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. **R2 Score:** It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE): MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

5. Hyper parameter Tuning: There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyper parameters. These hyper parameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyper parameters for a given model as it varies from model to model.We are not aware of optimal values for hyper parameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyper parameters is known as Hyper parameter Tuning. We can do tuning by using GridSearchCV.

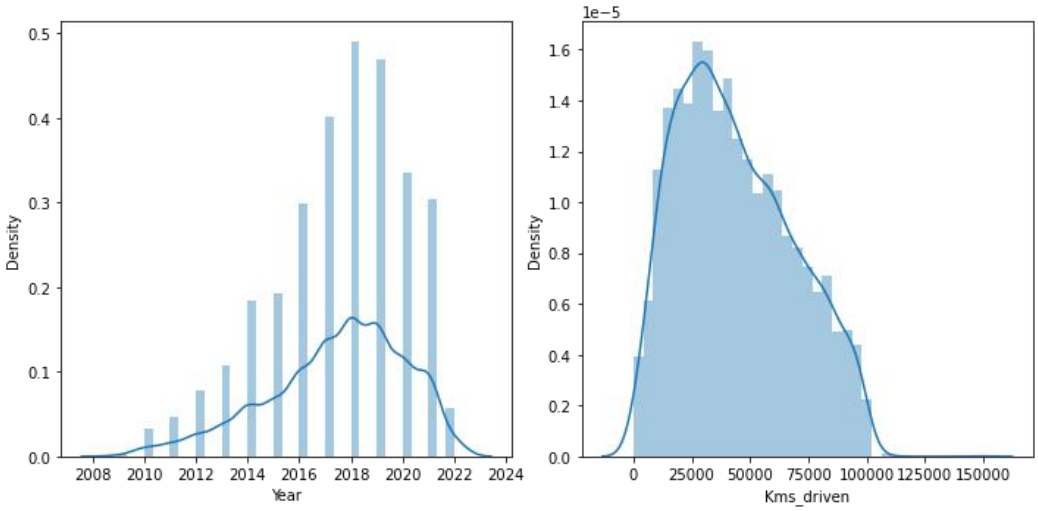
GridSearchCV is a function that comes in the Sci kit-learn (or SK-learn) model selection package. An important point here to note is that we need to have the Sci kit-learn library installed on the computer. This function helps to loop through predefined hyper parameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyper parameters.

Visualization:

I used pandas profiling to get the over viewed visualization on the pre processed data. Pandas is an open source python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable.

I have analyzed the data using distribution plot, pie plot,count plot, bar plot, line plots and box plots to get the relation between the features and label.

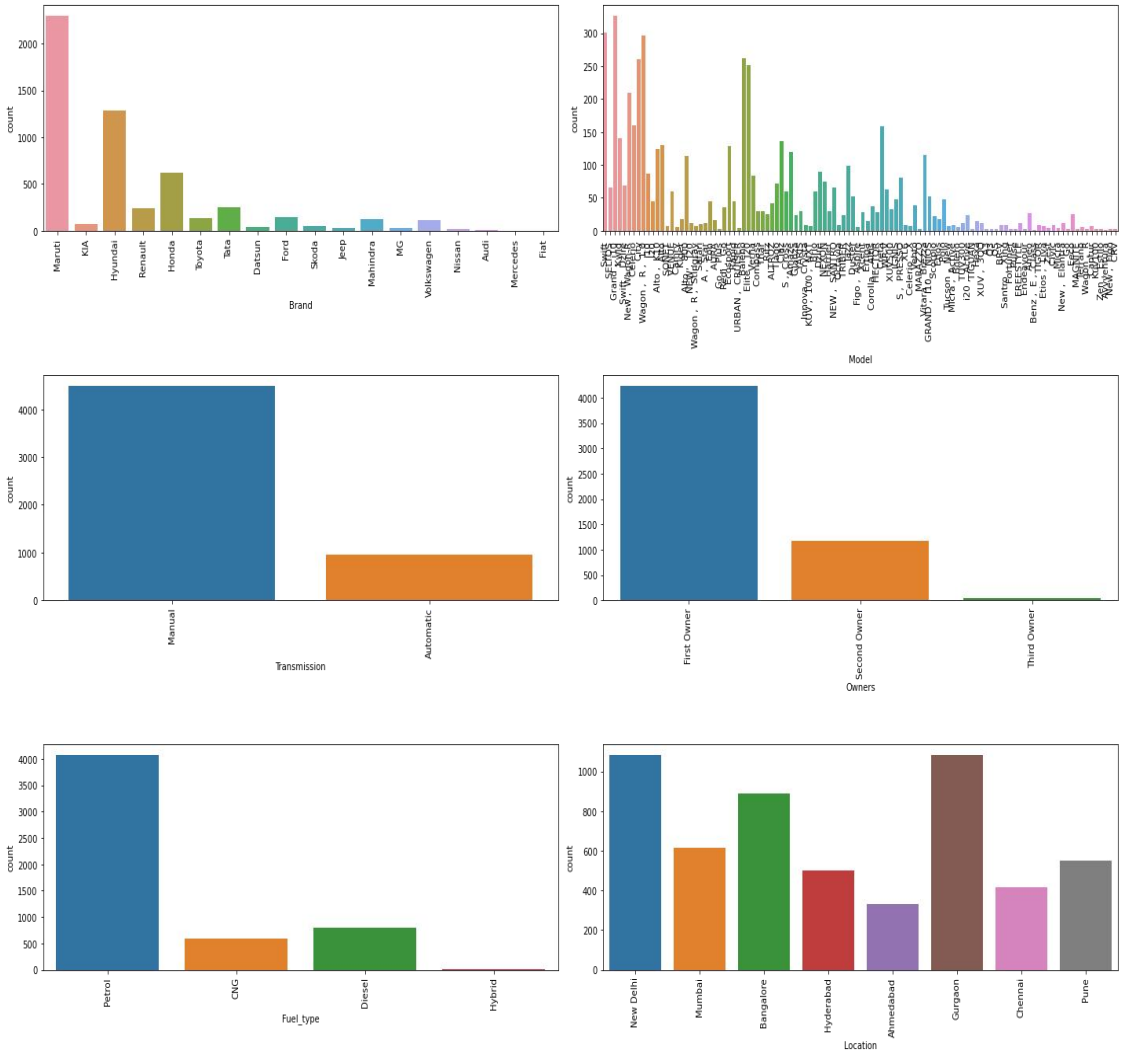
Distribution Plots:



In above plots we can see that:

- outliers may be present in both the columns and it was seen that skewness is present in column year.

Count Plots:

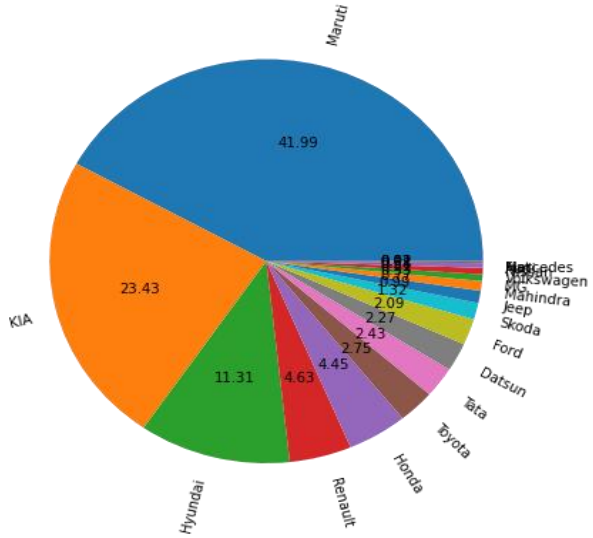


In above plots we can see that:

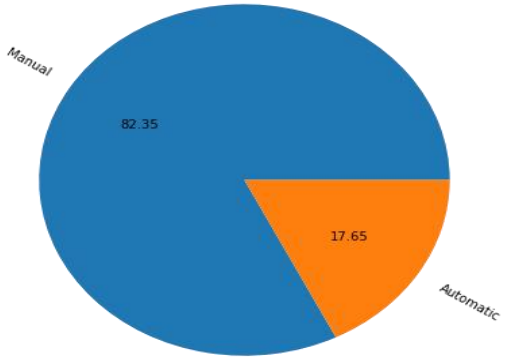
- In column Brand, we can see that most of the cars are from brand Maruti.
- In column Model, we can see that most of the models of the cars is Grand i 10.

- In column transmission, we can see that most of the cars have gear type manual.
- In column owners, we can see that most of the cars have single owners.
- In column fuel_type, we can see that most of the cars have petrol as fuel type.
- In column location, we can see that most of the cars are from location New Delhi and Gurgaon.

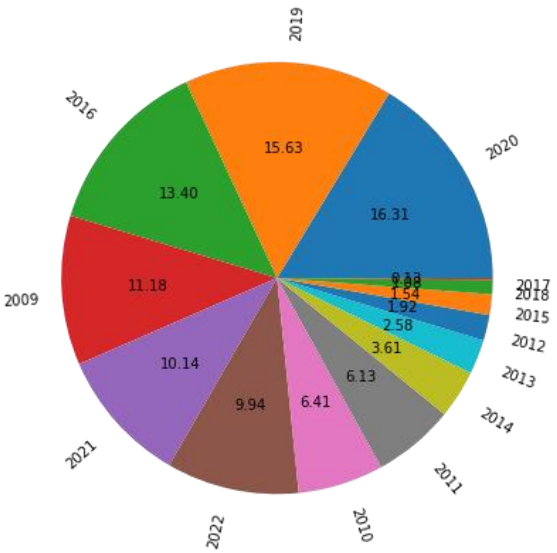
Pie Plots:



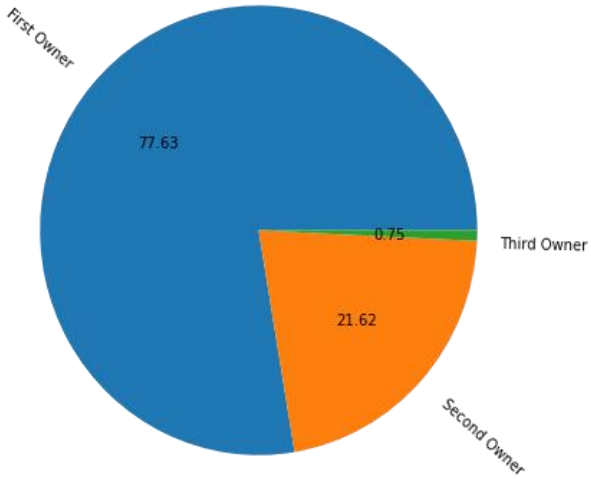
- In the above plot we can see clearly see that most of the cars from brand Maruti followed by KIA and Hyundai.



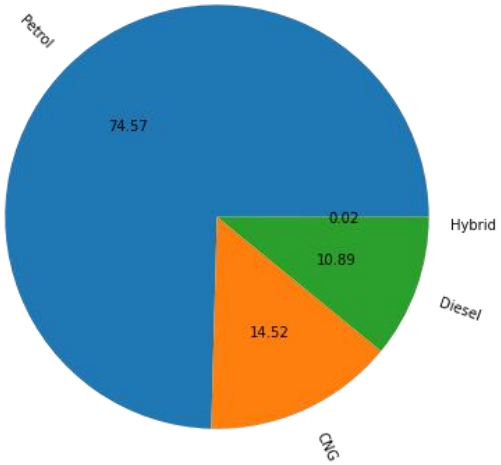
- In the above plot we can see that most of the cars was having gear type as manual.



- In the above plot we can see that most of the cars were from year 2020 followed by 2019 and 2016.

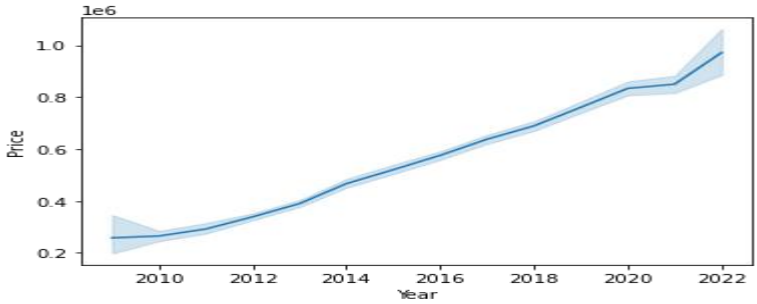


- By looking at the plot we can see that most of the cars was owned by single owners.

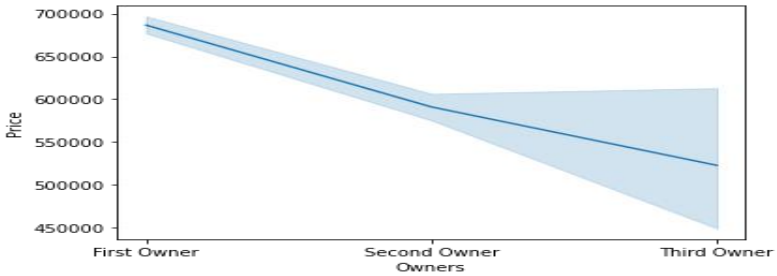


- By looking at the above plot we can see that most of the car was having fuel type as petrol followed by CNG and Diesel.

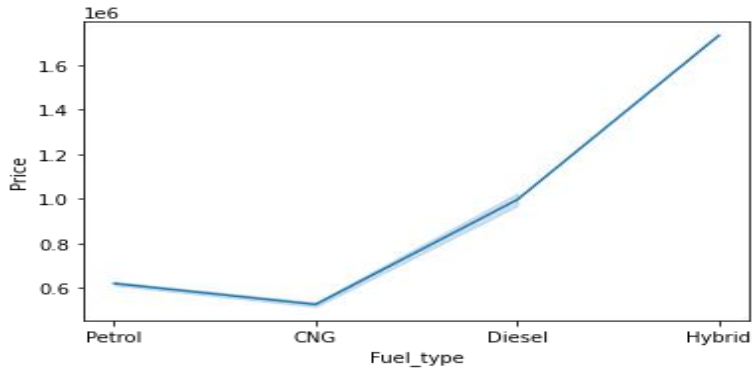
Line Plots:



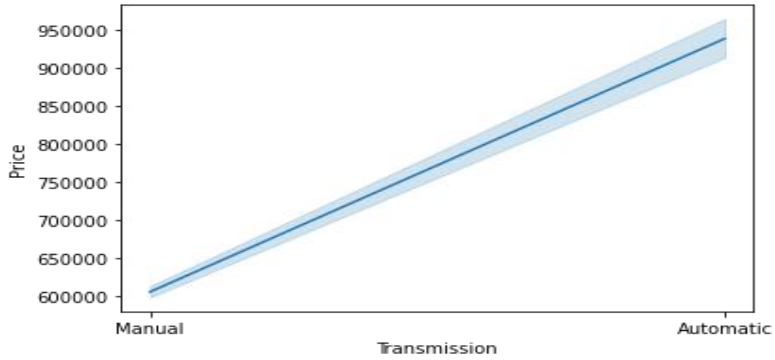
- By looking at the above plots we can clearly see that as the date/year of the car is closed to the present date/year, the price is more, so we can see in the plot that as the years starts coming close to the present year the price of the car starts increasing.



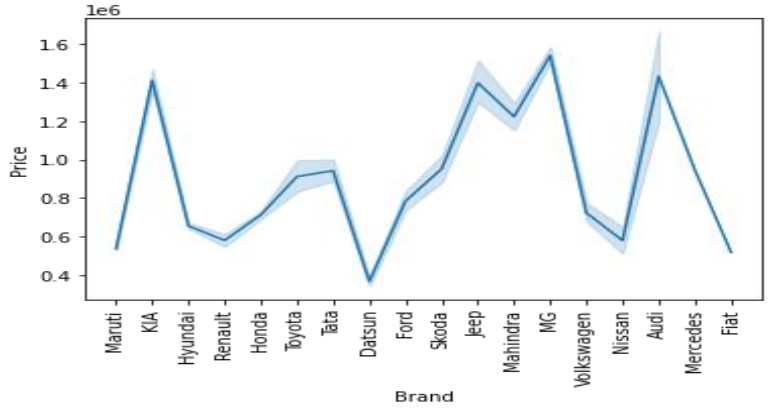
- In this plot we can see that as the owners starts increasing the price of the car starts decreasing.



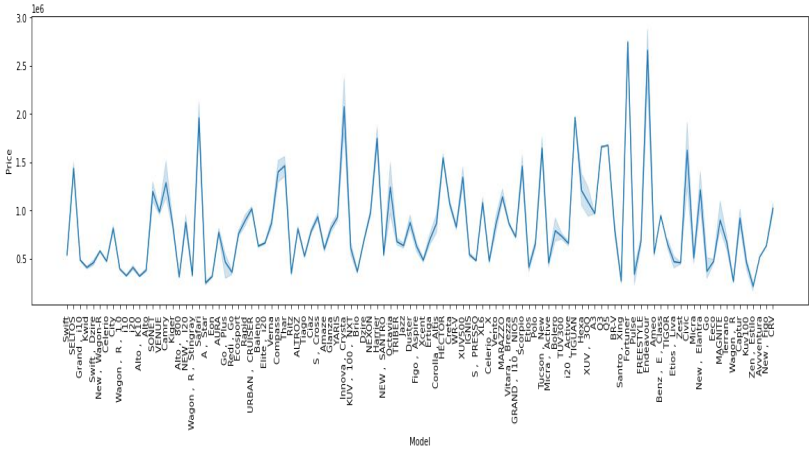
- In this plot we can see that CNG used fuel car have low price and Hybrid cars have more price compared to petrol and diesel used cars.



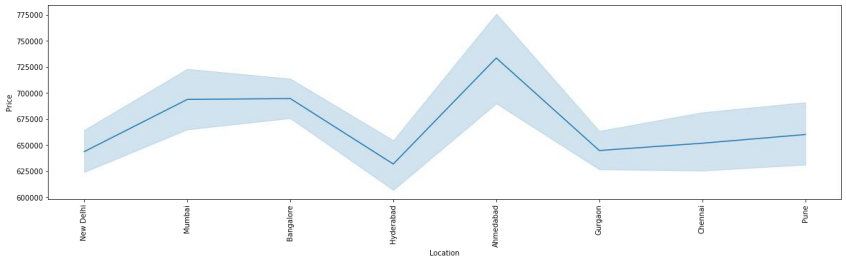
- In this plot we can see that the car having automatic gear type have more price than that of manual used gear type.



- In this plot we can see that the price fluctuates according to the brands, so in this we can see that MG brand have the highest price.

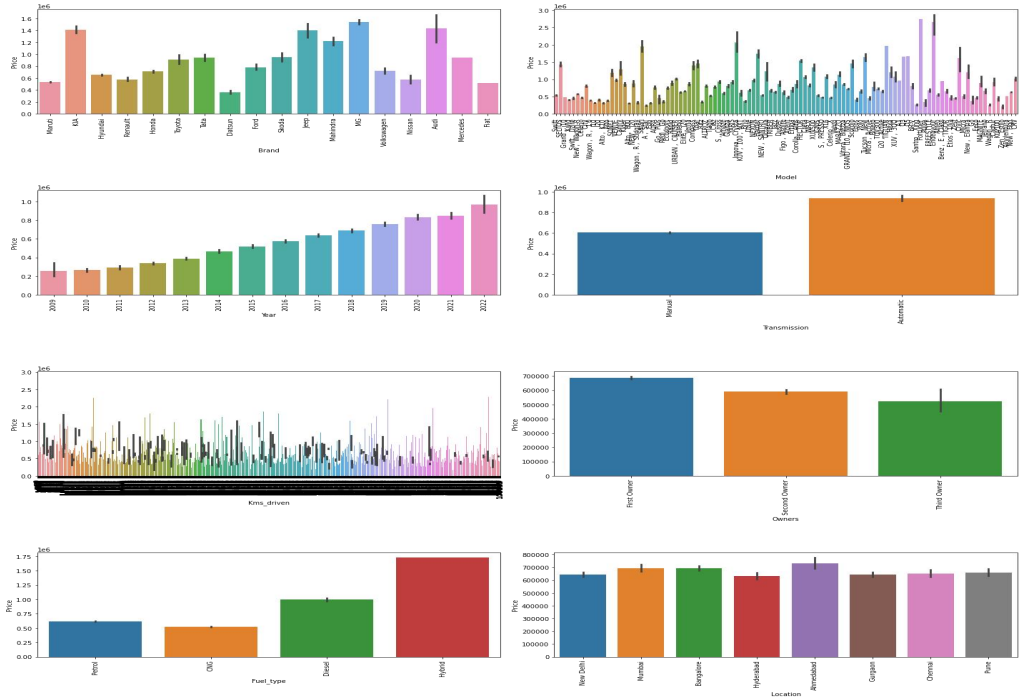


- In this plot we can see that Fortuner and Endeavour models have high price than that of other models.



- In this plot we can see that cars from location Ahmadabad have high prices as compared to other locations.

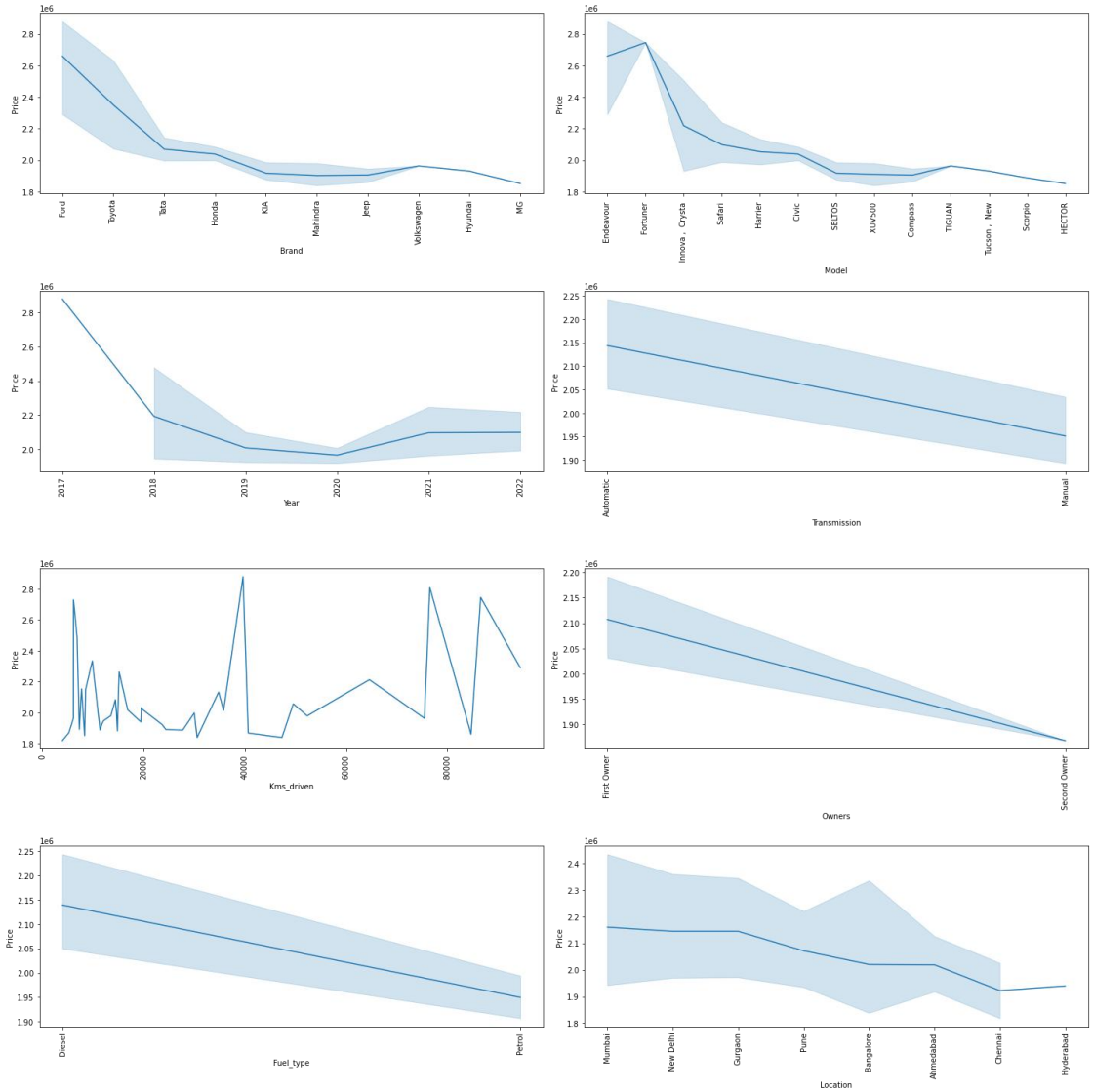
Bar Plots:



In above plots we can see that:

- In the plot brand vs price we can see that the price fluctuates according to the brands, so in this we can see that MG brand have the highest price.
- In the plot model vs price we can see that Fortuner and Endeavour models have high price than that of other models.
- By looking at the above plot year vs price we can clearly see that as the date/year of the car is closed to the present date/year, the price is more, so we can see in the plot that as the years starts coming close to the present year the price of the car starts increasing.
- In the plot transmission vs price we can see that the car having automatic gear type have more price than that of manual used gear type.
- In the plot owners vs price we can see that as the owners starts increasing the price of the car starts decreasing.
- In the plot fuel vs price we can see that CNG used fuel car have low price and Hybrid cars have more price compared to petrol and diesel used cars.
- In the plot location vs price we can see that cars from location Ahmadabad have high prices as compared to other locations.

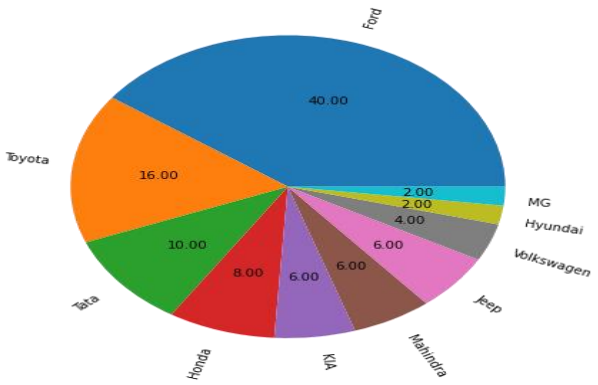
Top 50 highest priced cars Line Plots:



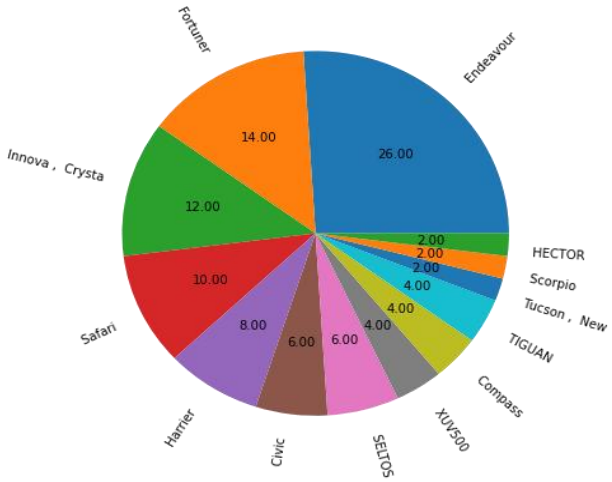
In above plots we can see that:

- Above plots are the plotted of top 50 highest priced car data
- So in plot brand vs price we can see that brand Ford has the highest price among other brands.
- In plot model vs price we can see that model Endeavour have the highest price among other models.
- In plot fuel type vs price we can see that diesel cars have more price than the petrol cars.

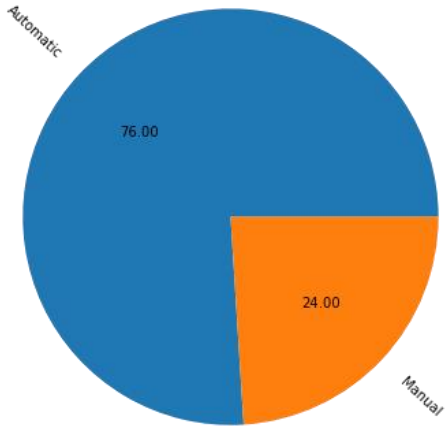
Top 50 highest priced cars Pie Plots:



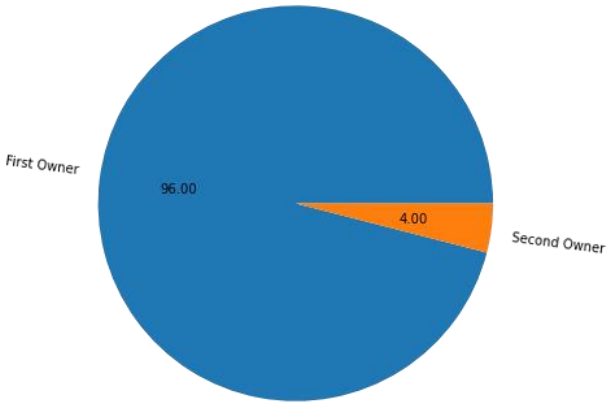
- From above plot we can see that around 40% of cars which are there in top 50 highest priced cars are from brand Ford.



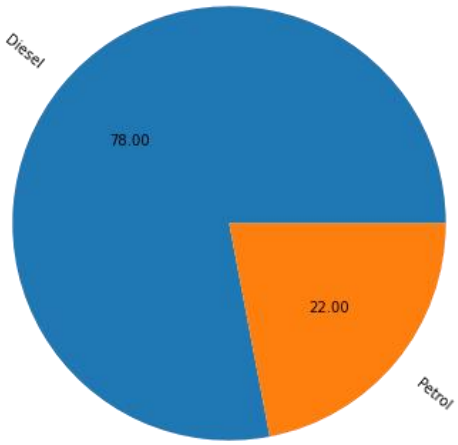
- From above plots we can see that around 26% of cars which are there in top 50 highest priced cars is of model Endeavour.



- From above plots we can see that around 76% of cars which are there in top 50 highest priced cars have Automatic gear type.



- From above plots we can see that around 96% of cars which are there in top 50 highest priced cars have single owner.



- From above plots we can see that around 78% of cars which are there in top 50 highest priced cars have fuel type diesel.

INTERPRETATION OF THE RESULTS

Visualizations: I have used distribution plot to visualize the numerical Variables So over there I found that the data was not normally distributed. Used bar plot and line plot to check the relation between label and the features. The heat map and bar plot helped me to understand the correlation between dependent and independent features. Also, heat map helped to detect the multicollinearity problem within the features. Detected outliers and skewness with the help of box plots respectively.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple regression models to get the accurate R2 score, and evaluation metrics like MAE, MSE and RMSE. I got Xtreme Gradient Boosting Regressor (XGB) as the best model which gives 94.49% R2 score. I checked the cross-validation score ensuring there will be no over fitting and under fitting. After tuning the best model, the R2 score of Extreme Gradient Boosting Regressor has been reduced. Finally, I saved my final model on which hyper parameter tuning was not done and got the good predictions results for price of used cars.

CONCLUSION

Key Findings and Conclusions of the Study:

The case study aims to give an idea of applying machine learning algorithms to predict the price of the used car. After the completion of this project, we got an insight of how to collect data, pre-processing the data, analyse the data, cleaning the data and building a model.

In this we have used 6 learning models to predict the price of the used cars.

From our detailed analysis we can determine the following:

- Year and transmission was having some strong relationship with the label that is Price.
- Car Brand is having highest price among all the other brands.
- Car model Endeavour is having high price among all the other model.
- Cars having fuel type as hybrid are having highest prices.
- Cars having gear type as Automatic are having high prices as compared to others.
- Cars which are located at Ahmadabad are having high prices as compared to other locations.

Learning Outcomes of the Study in respect of Data Science:

While working on this project I have learned many things about the features of the Cars and about the used car selling website and came to know how the machine learning models have helped to predict the price of the used cars.

I found that the project was quite interesting as I came to know about the different features of the cars which plays an important role in deciding the price of the used cars. Also came to know which brand and model have the highest prices. So overall the experience was good as I came across a new challenge.

Visualization plays an important role in data analysis as it gives you a perfect idea of what is going through the dataset and also data cleaning was one of the most important and crucial thing about the project.

Finally we achieved what we had visualized by predicting the prices of the used cars and by building the Car price evaluation model that could help the buyers to understand the future price of the used cars.

Limitations of this work and Scope for Future Work:

Limitations: The main limitation is the low number of records that have been used in each locations. Some of the values in dataset was unrealistic as the kms driven was showing as 0kms which is not possible in case of the used cars. So because of that data our models may not make the right patterns and the performance of the model also reduces.

Future Work: The greatest shortcoming of this work is the shortage of data. I think we should scrap the data from each location more so that we can have more cars from that particular location so that it will be good for us to check the trend.