CAR PRICE PREDICTION PROJECT

Submitted by:

RAVINDER SINGH

ACKNOWLEDGMENT

1. Prashant Mundepi Senior Manager Cars24 (Refurb Department)
2. https://www.kaggle.com/datasets/hellbuoy/car-price-prediction
3. https://towardsdatascience.com/predicting-car-price-using-machine-learning-8d2df3898f16

INTRODUCTION

- ## Business Problem Framing

  Scrape the data for used cars Platform and build a model to predict price of the car.

- ## Conceptual Background of the Domain Problem

  Used cars are basic valued over the age of the vehicle,model, variant, transmission,kms covered.

- ## Review of Literature

  As in the Discussion with Cars 24 refurbSenior Manager. The cars are valued on the bases of the brand it belongs to, kms covered, Transmission (automatic or manual) , fuel variant (petrol, diesel,etc) and some of the other factors.

- ## Motivation for the Problem Undertaken

  The problem helps us to understand the current Used car market and the effect of Corona and Semi- Conductor shortage on the car market , the Price of Electric vehicle are on the Higher side

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  It is Problem based Multiple linear Regression. Every column was related to the vehicle price with a positive co-relation or a negative – negative co-relation. Achieving a high accuracy is not possible as the Used car price is determined on the factor of damage which was not mentioned on the website And whether a Car is accidental or not that can affect the price of the can drastically

- ## Data Sources and their formats

  I used a single source (cars 24.com) to scrap the Data as different companies uses different algorithms and different parameters to fix the price of the car. Ex: Spinny doesn't deal in the car manufactured after 2015.

- ## Data Preprocessing Done

  Columns with Numerical values were taken as the Object. Price column contained various special characters needed to be removed. For categorical variable I preferred Dummies over label Encoding. Age of the vehicle was calculated.

- ## Data Inputs- Logic- Output Relationships

  Each had a linear relationship with the Target column the dummy values in the column helped in attaining a high R- Squared. Columns like Age have a negative relation with the Price column.

- ## State the set of assumptions (if any) related to the problem under consideration

  While building this project I have assumed that the car is not demaged or demage doesn't effect the price of the car as its minimal .

- Hardware and Software Requirements and Tools Used

The hardware used

- Processor – Intel i3 7th gen
- RAM – 4GB
- Hard Disk -512 Gb

Software used

- Jupyter Notebook.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The Data was collected from the various cities in India. Apporach toward the statement is quite simple to apply various linear model in term to find the best suited for prediction.(Assuming the data is linearly related to the target variable

- Testing of Identified Approaches (Algorithms)

  Linear Regression, Ridge regression, LassoRegression, Random Forest Regression, Xgboost Regression.

- Run and Evaluate selected models

  Refer to Car_price_prediction.ipynb file

LINEAR REGRESSION

```
model_building(features,target,LinearRegression())
```

```
R-square 0.9170458064034965 Train set R Squared 0.9367935750233971 Random state 99 test_size 0.2
```

```
# creating empty list that will help us in the end to judge the best model
model_name=[]
mse=[]
mae=[]
R_sq=[]
rmse=[]
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.25,random_state=99)
lr=LinearRegression()
lr.fit(x_train,y_train)
lr_pred=lr.predict(x_test)
print(lr.score(x_train,y_train))
print('MSE:',mean_squared_error(lr_pred,y_test))
print('MAE:',mean_absolute_error(lr_pred,y_test))
print('r2_score:',r2_score(lr_pred,y_test))
print('RMSE Score',np.sqrt(metrics.mean_squared_error(y_test,lr_pred)))
```

```
0.938205055122747
MSE: 11874467741.008944
MAE: 73961.48417445632
r2_score: 0.9120526186841148
RMSE Score 108970.03138940973
```

RIDGE REGRESSION

```
model_building(features,target,Ridge())
```

```
R-square 0.91670521508989 Train set R Squared 0.9322957774214293 Random state 99 test_size 0.2
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2,random_state=99)
parameters={"alpha":[1e-15,1e-10,1e-8,1e-3,1e-2,1,5,10,20,30,35,40,45,50,55,100],'solver':['auto','svd','cholesky','lsqr']}
ridge=Ridge()
RR=GridSearchCV(ridge,parameters,scoring='neg_mean_squared_error')
RR.fit(x_train,y_train)
print(RR.best_params_)
```

```
{'alpha': 0.01, 'solver': 'auto'}
```

```
rr=Ridge(alpha=0.01,solver='auto')
rr.fit(x_train,y_train)
rr_pred=rr.predict(x_test)
print('R2_score:',r2_score(y_test,rr_pred))
print('mse:',metrics.mean_squared_error(y_test,rr_pred))
print('mae:',metrics.mean_absolute_error(y_test,rr_pred))
print('rmse:',np.sqrt(metrics.mean_squared_error(y_test,rr_pred)))
mse.append(mean_squared_error(rr_pred,y_test))
mae.append(mean_absolute_error(rr_pred,y_test))
R_sq.append(r2_score(y_test,rr_pred))
model_name.append("ridge_regression")
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,rr_pred)))
plt.scatter(x=y_test,y=rr_pred)
```

```
R2_score: 0.9170897898944385
mse: 11519900478.867067
mae: 72941.3592562054
rmse: 107330.79930228353
```

Lasso Regression

```
model_building(features,target,Lasso())
```

```
R-square 0.9171109297616915 Train set R Squared 0.9367927607431981 Random state 99 test_size 0.2
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2,random_state=99)
parameters={'alpha':[0.0001,0.001,0.01,0.1,1,10]}
lasso=Lasso()
LS=GridSearchCV(lasso,parameters,scoring='neg_mean_squared_error')
LS.fit(x_train,y_train)
print(LS.best_params_)
```

```
{'alpha': 10}
```

```
ls=Lasso(alpha=10)
ls.fit(x_train,y_train)
ls_pred=ls.predict(x_test)
print('R2_score:',r2_score(y_test,ls_pred))
print('mse:',metrics.mean_squared_error(y_test,ls_pred))
print('mae:',metrics.mean_absolute_error(y_test,ls_pred))
print('rmse:',np.sqrt(metrics.mean_squared_error(y_test,ls_pred)))

mse.append(mean_squared_error(ls_pred,y_test))
mae.append(mean_absolute_error(ls_pred,y_test))
R_sq.append(r2_score(y_test,ls_pred))
model_name.append("lasso_regression")
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,ls_pred)))
plt.scatter(x=y_test,y=ls_pred)
```

```
R2_score: 0.9175332326411839
mse: 11458286640.184477
mae: 72591.632929028
rmse: 107043.3867185847
```

Xgboost Regressor

```
model_building(features,target,XGBRegressor(objective ='reg:linear', verbosity = 0, random_state=42))
```

```
R-square 0.9289885657785146 Train set R Squared 0.9784743039298281 Random state 99 test_size 0.2
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2,random_state=99)
xgb1 = XGBRegressor(verbosity = 0, random_state=42)
parameters ={
        'objective':['reg:linear'],
         'max_depth': [1,2,3,4,5,6,7,8,9,10],
         'colsample_bylevel':[0.2,0.5,0.6,1],
         'learning_rate':[0.01,0.1,0.3,0.2,1],
         'n_estimators':[100,150,200,500]
                  }

xgb_grid = GridSearchCV(xgb1,parameters,cv=3)

xgb_grid.fit(x_train, y_train)
print(xgb_grid.best_params_)
```

```
{'colsample_bylevel': 0.5, 'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 500, 'objective': 'reg:linear'}
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2,random_state=99)
Xg=XGBRegressor(objective ='reg:linear',verbosity= 0, random_state=42, colsample_bylevel=0.5,learning_rate=0.2,max_depth= 5,n_estimators=500)
Xg.fit(x_train,y_train)
Xg_pred=Xg.predict(x_test)
print('R2_score:',r2_score(y_test,Xg_pred))
print('mse:',metrics.mean_squared_error(y_test,Xg_pred))
print('mae:',metrics.mean_absolute_error(y_test,Xg_pred))
print('rmse:',np.sqrt(metrics.mean_squared_error(y_test,Xg_pred)))

mse.append(mean_squared_error(Xg_pred,y_test))
mae.append(mean_absolute_error(Xg_pred,y_test))
R_sq.append(r2_score(y_test,Xg_pred))
model_name.append("XGboost_regression")
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,Xg_pred)))
plt.scatter(x=y_test,y=Xg_pred)
```

```
R2_score: 0.945101704035153
mse: 7627804888.797454
mae: 54075.23846053547
rmse: 87337.30525266654
```

RANDOM FOREST

```
model_building(features,target,RandomForestRegressor())
```

```
R-square 0.928440693145844 Train set R Squared 0.9893723709466444 Random state 99 test_size 0.2
```

```
x_train, x_test, y_train, y_test = train_test_split(features, target, test_size=0.2,random_state=99)
model=RandomForestRegressor()
param_grid = { "n_estimators"    : [500,1000,1500],
               "min_samples_split" : [2,4,8],
              "bootstrap": [True, False]
             }
grid = GridSearchCV(model, param_grid, n_jobs=-1, cv=5)
grid.fit(x_train,y_train)
print(grid.best_params_)
```

```
{'bootstrap': True, 'min_samples_split': 2, 'n_estimators': 1500}
```

```
RF=RandomForestRegressor(n_estimators= 1500,bootstrap= True,min_samples_split=2,)
RF.fit(x_train,y_train)
RF_pred=RF.predict(x_test)
print('R2_score:',r2_score(y_test,RF_pred))
print('mse:',metrics.mean_squared_error(y_test,RF_pred))
print('mae:',metrics.mean_absolute_error(y_test,RF_pred))
print('rmse:',np.sqrt(metrics.mean_squared_error(y_test,RF_pred)))

mse.append(mean_squared_error(RF_pred,y_test))
mae.append(mean_absolute_error(RF_pred,y_test))
R_sq.append(r2_score(y_test,RF_pred))
model_name.append("Random_forest_regression")
rmse.append(np.sqrt(metrics.mean_squared_error(y_test,RF_pred)))
plt.scatter(x=y_test,y=RF_pred)
```

```
R2_score: 0.929418638480339
mse: 9806877335.54411
mae: 60269.63495224849
rmse: 99029.67906412759
```

- Key Metrics for success in solving problem under consideration

  MSE, MAE, R-squared, Adjusted R-squared, and RMSE.

  Mean Squared Error (MSE)
  The most common metric for regression tasks is MSE. It has a convex shape. It is the average of the squared difference between the predicted and actual value. Since it is differentiable and has a convex shape, it is easier to optimize.

  Mean Absolute Error (MAE)
  This is simply the average of the absolute difference between the target value and the value predicted by the model. Not preferred in cases where outliers are prominent.

  R-squared or Coefficient of Determination
  This metric represents the part of the variance of the dependent variable explained by the independent variables of the model. It measures the strength of the relationship between your model and the dependent variable.

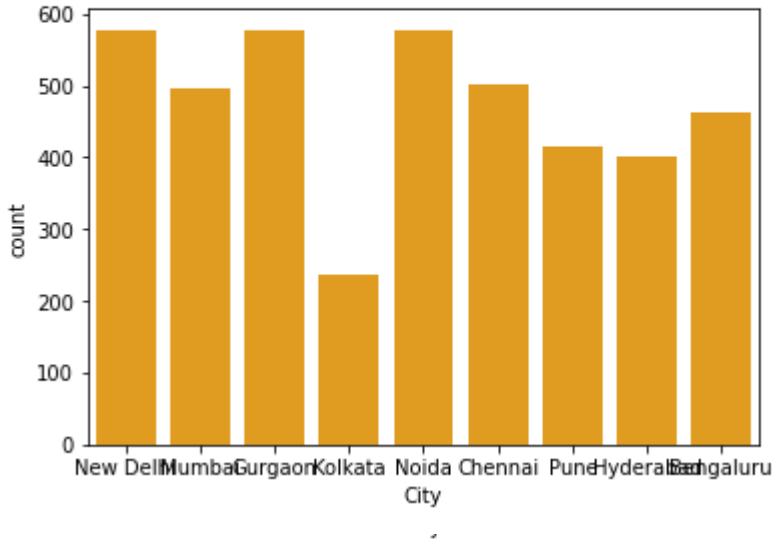
  Root Mean Squared Error (RMSE)
  This is the square root of the average of the squared difference of the predicted and actual value.

- Visualizations

  Univariate analysis of the categorical column.

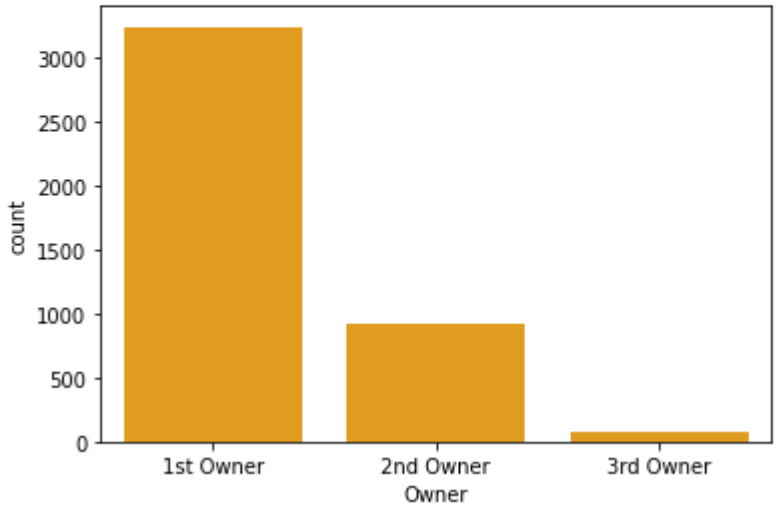
```
sns.countplot(data.City, color = 'Orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f26ee318c10>
```



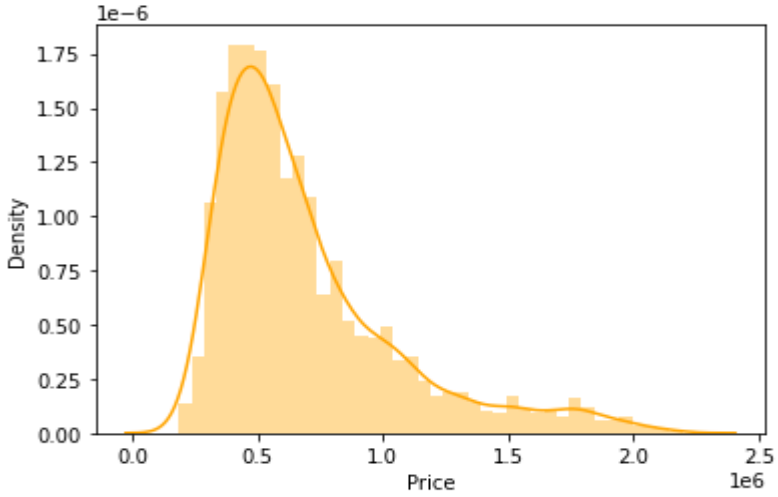

```
sns.countplot(data.Owner, color = 'Orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f26ee203290>
```



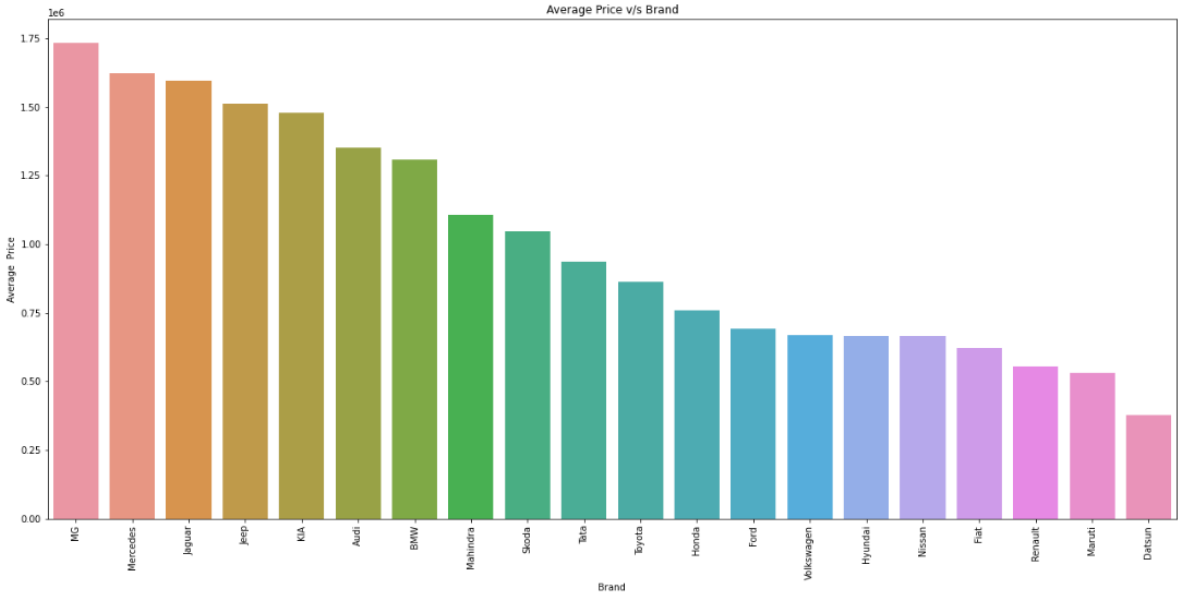
```
sns.distplot(data.Price, color = 'Orange')
```

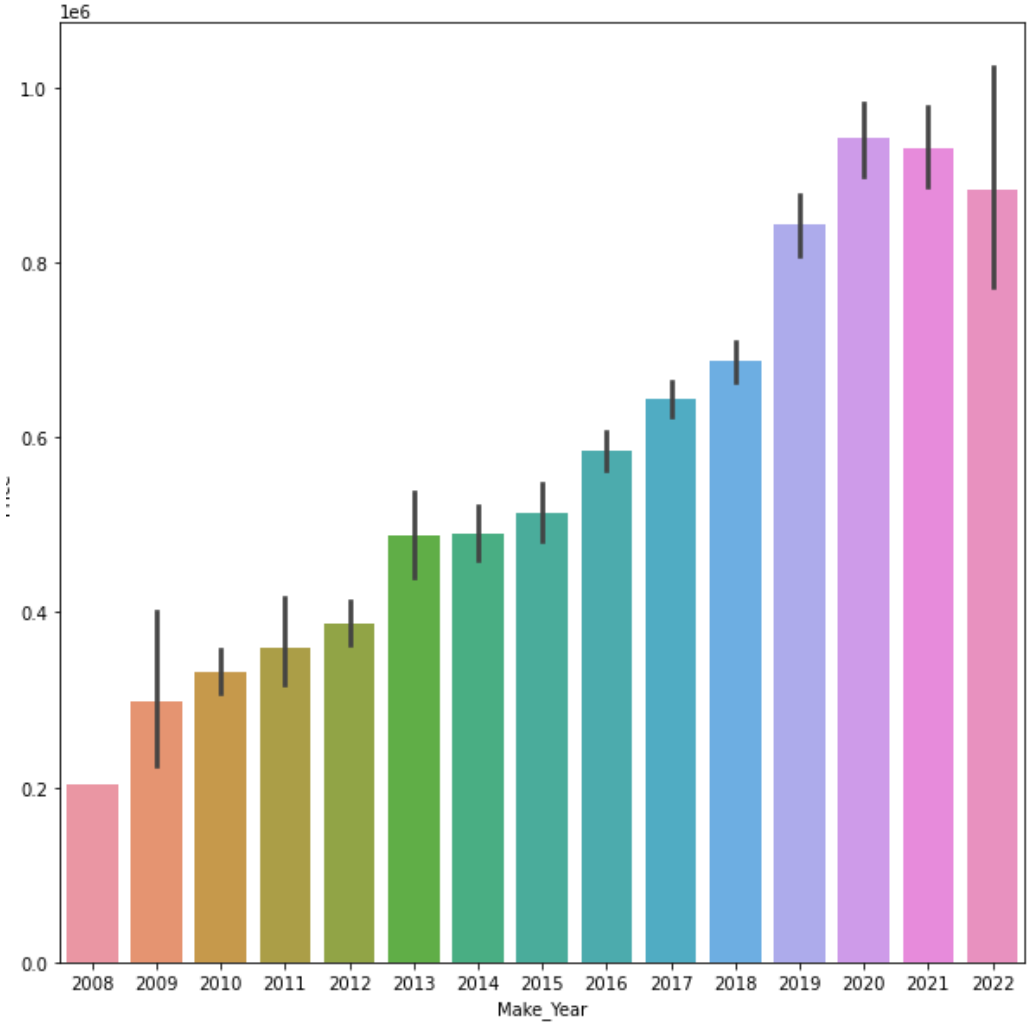
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```

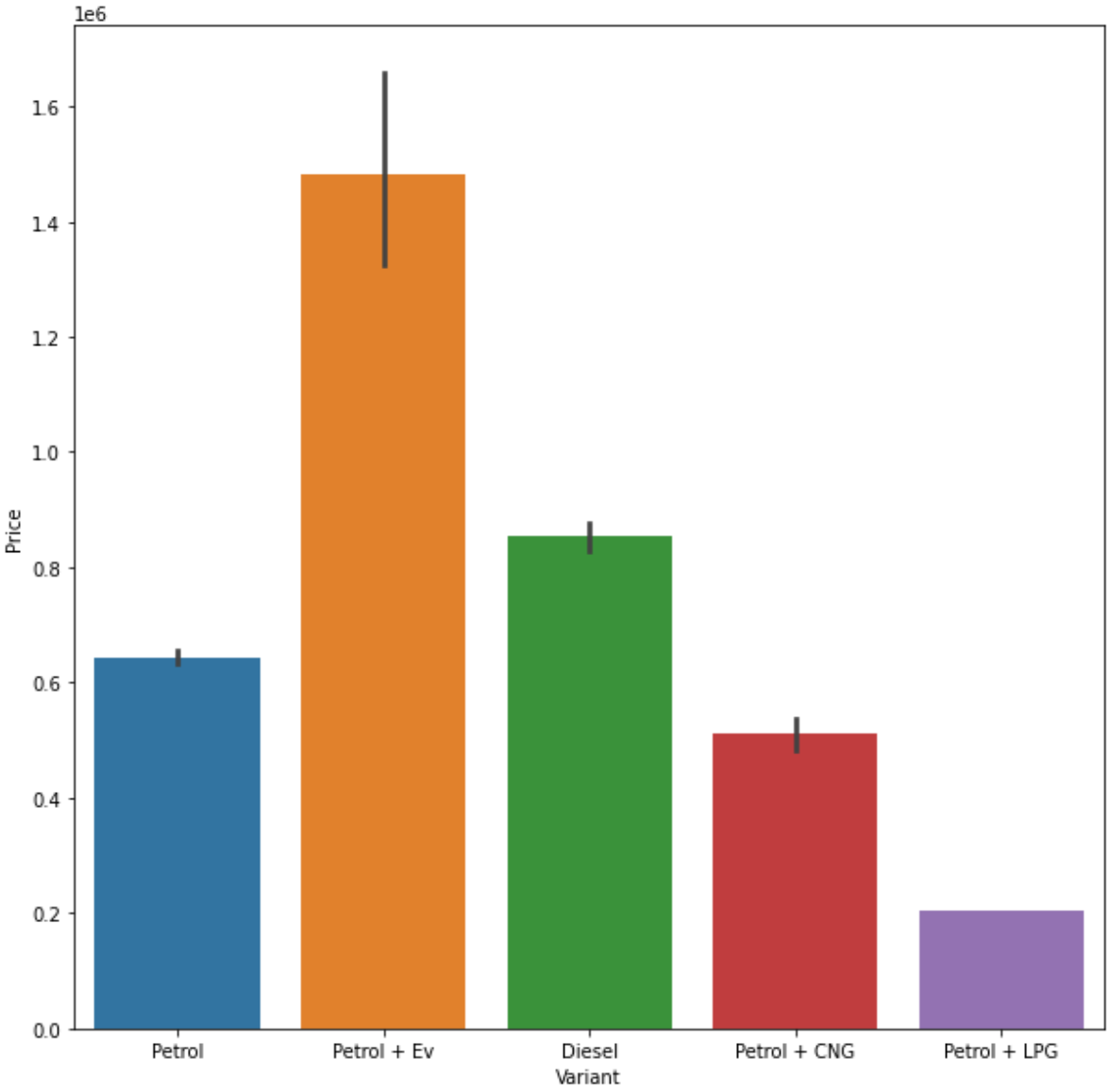


Bi- variate analysis of Year vs Price

```
Text(0.5, 1.0, 'Average Price v/s Brand')
```







- Interpretation of the Results

  Was able to get a R- squared of 90% from all the models .

  Picked Xgboost as on hyper parameteric tunning it was able get a result of 96%.

  Pickled the model for the future predictions

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The price of car is inversely proportional to the age of the vehicle.

  The Brand like Bmw and Audi sells at a higher price.

  The large portion of used car Business mostly operate in metropolitan cities

  Automatic Transmission Cars cost on higher Side

- ## Learning Outcomes of the Study in respect of Data Science

  Data collected was assumed to have linear relation with each and every attribute thus Linear Regression should be the best model.

  Contrary the Xgboost performs better than linear regression means that one or more column has quadratic or cubic relation

- ## Limitations of this work and Scope for Future Work

  Any Used car seller Quote the price after Accessing the condition of the car on their Quality Parameter which are usually not enter on their website and hence not Available for scrapping so we predicting the price on a Simple hypothesis that the car has never involved in an accident or has a minimal surface demage.