FLIP ROBO

PROJECT REPORT ON:

"Car Price Prediction"

SUBMITTED BY

Ajit Madame

ACKNOWLEDGMENT

I would like to express my special gratitude to "Flip Robo" team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. And I want to express my huge gratitude to Ms.Gulshana Chaudhary (SME Flip Robo), she is the person who has helped me to get out of all the difficulties I faced while doing the project.

A huge thanks to my academic team "Data trained" who are the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank you for many other persons who has helped me directly or indirectly to complete the project

Contents:

1.INTRODUCTION

1.1Business 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. With the change in market due to covid 19 impact. For this I am collecting data from cars24.com and make data frame and used this data for building the model. The project Car Price Prediction deals with providing the solution to this problem. Through this project, we will get to know which of the factors are significant and tell us how they affected the cars market.

1.2 Conceptual Background of the Domain Problem

A good knowledge of after sales market of cars is necessary. What makes a car valuable will be key. As the mobile internet improves by leaps and bounds, the model traditional offline used car trading has gradually lost the ability to lives up to the needs of customers, and online used car trading platforms have emerged as the times require. Second-hand car price prediction is the premise of second-hand car trading, and reasonable price can reflect the objective, fair and true nature of the second-hand car market.

1.3 Review of Literature

The first paper is Predicting the price of Used Car Using Machine Learning Techniques. In this paper, they investigate the application of supervised machine learning techniques to predict the price of used cars in Mauritius. The predictions are based on historical data collected from daily newspapers. Different techniques like multiple linear regression analysis, Random forest regressor, Gradient Boosting Regression, XGBoost regressor have been used to make the predictions. The Second paper is Car Price Prediction Using Machine Learning Techniques. Considerable number of distinct attributes are examined for the reliable and accurate prediction.

1.4 Motivation for the Problem Undertaken

The goal of this project is to create machine learning models that can properly forecast the price of a used car based on its attributes so that buyers can make educated decisions. On a dataset containing the sale prices of various brands and models, we build and analyses several learning approaches. Due to covid-19 the car market has changed a lot, some cars have shot up in popularity and some gone down in price.

2.Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

Inbuilt function such as standardising and log will be used in tackling this problem.

R-square is a comparison of residual sum of squares *(SS_{res})* with total sum of squares*(SS_{tot})*. Total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Where SS_{res} is the residual sum of squares and SS_{tot} is the total sum of squares.

R-square is the main metric which I will use in this regression analysis.

Concordance index was also used. The concordance index or c-index is a metric to evaluate the predictions made by an algorithm. It is defined as the proportion of concordant pairs divided by the total number of possible evaluation pairs.

2.2 Data Sources and their formats

The data was scraped from cars24 website; data was scraped for different cities where prices differ.

	Unnamed: 0	name	selling_price	km_driven	fuel	transmission	owner
0	0	2019 Maruti Swift	5,34,399	11,404	Petrol	Manual	2nd Owner
1	1	2018 Hyundai Grand i10	5,46,599	6,875	Petrol	Manual	2nd Owner
2	2	2021 Maruti Swift	5,55,899	13,174	Petrol	Manual	1st Owner
3	3	2020 Maruti Swift	5,57,199	16,633	Petrol	Manual	2nd Owner
4	4	2009 Hyundai i10	1,70,699	45,140	Petrol	Manual	1st Owner

2.3 Data Pre-processing Done

- First step I have imported required libraries and I have imported the dataset which was in csv format.
- Then I did all the statistical analysis like checking shape, nunique, value counts, info etc.
- I found that, few features are containing null value so fill it with mean value.
- Numerical variables were converted to integer type (form string) so I could perform deeper analysis on them.
- Then doing some EDA and Building Models.

2.4 Data Inputs - Logic - Output Relationships

The main assumption is that there is no selection bias in the data which we have.

This is because we have cars from varying years and varying city; each city doesn't have equal amount of data. Here we can see the count of data per city.

2.5 Hardware, Software and Tool Used

Hardware Used:

Processor – Intel core i3

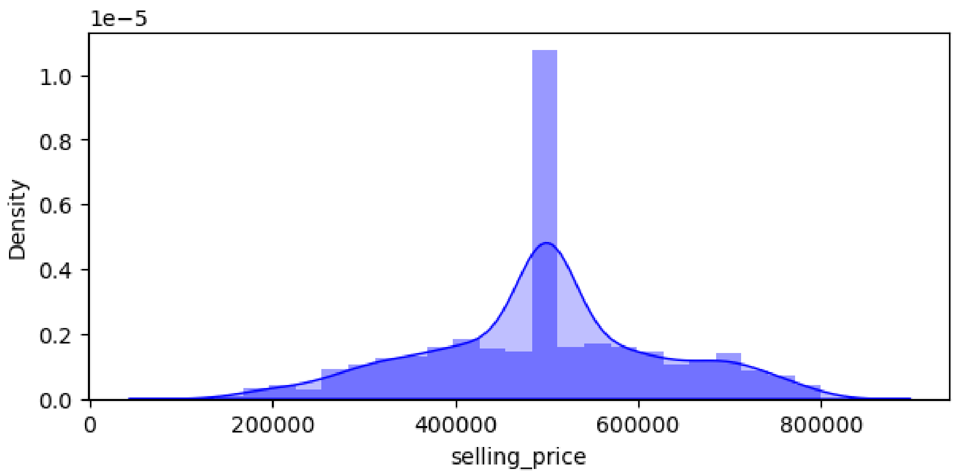
Physical Memory – 8 GB

Software Used:

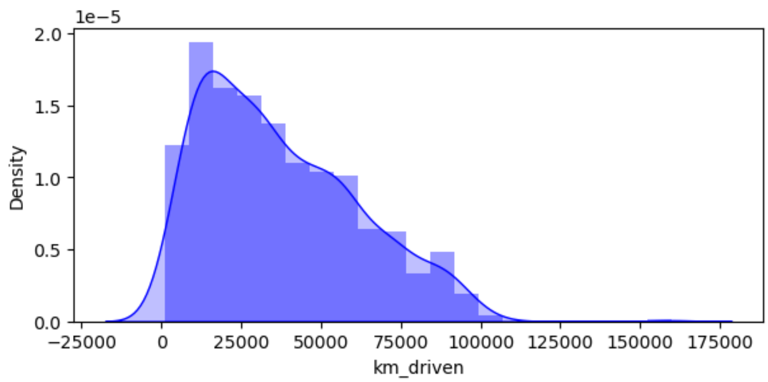
- Windows 10 Operating System
- Anaconda Package and Environment Manager
- Jupyter Notebook
- Python Libraries used: In Which Pandas, Seaborn, Matplotlib, Numpy and Scipy
- sklearn for Modelling Machine learning algorithms, Data Encoding, Evaluation metrics, Data Transformation, Data Scaling, Component analysis, Feature selection etc.

3.Data Analysis and Visualization

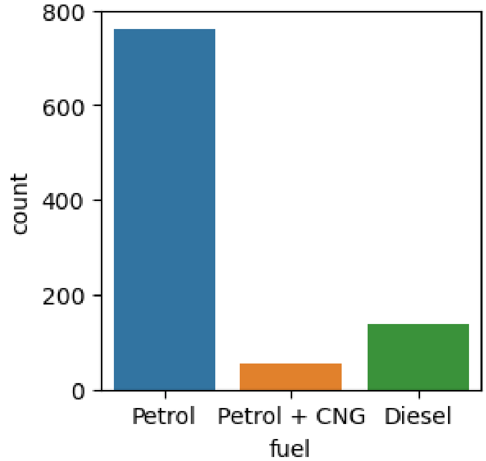
3.1 Univariate Visualization



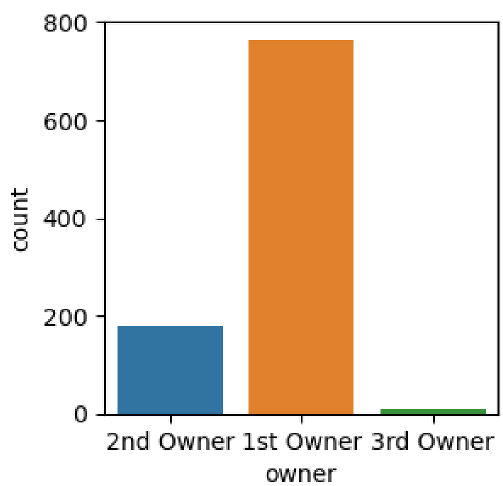
- We can see selling price is look like that data has normally distributed.
- We can see, maximum selling price lies in the range of 4 to 6 Lakh.



- We can see, km driven has not a normally distributed but maximum data lies in the range of 0 to 100000 km.

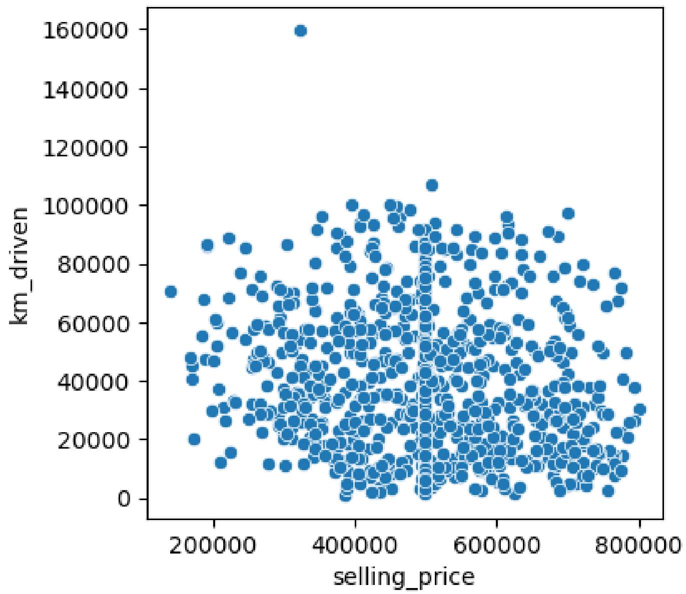


- We can see, Fuel as petrol has maximum count followed by Diesel.
- Petrol + CNG has lowest count than others.
- It means that maximum people are used petrol vehicle than others. But petrol vehicles are more expensive than others.

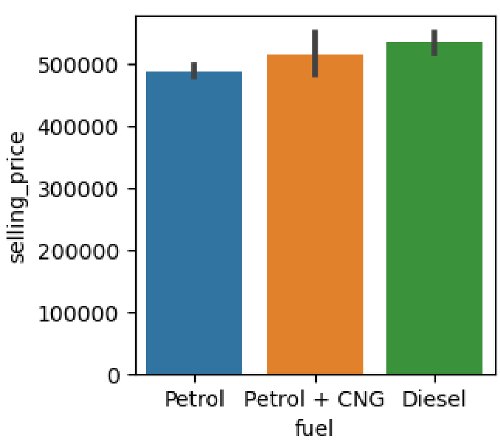


- We can see, 1st Owner owned vehicle are highest for selling followed by 2nd Owner.
- It simple because 1st owner vehicles are having more selling than others.

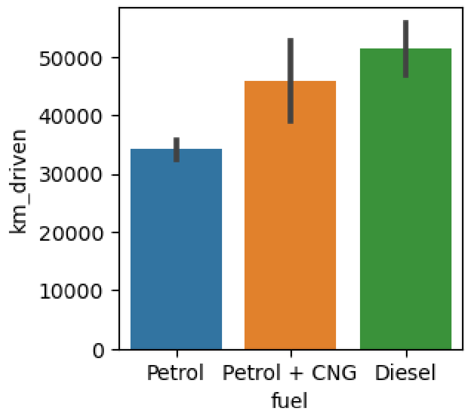
3.2 Bivariate Visualization

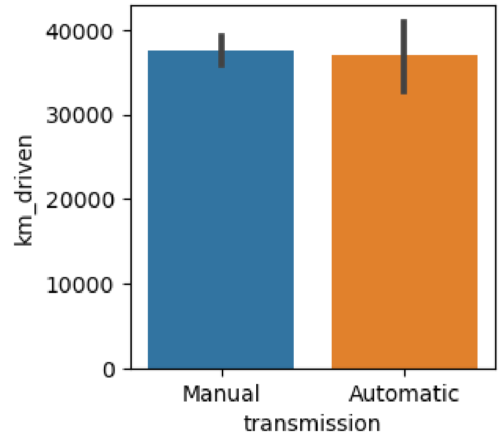


- We can see, there are no direct relationship of both feature to each other's.
- But the vehicle km driven are in 0 to 80000km those are having good price. And we can see maximum vehicle km driven lies in this range.

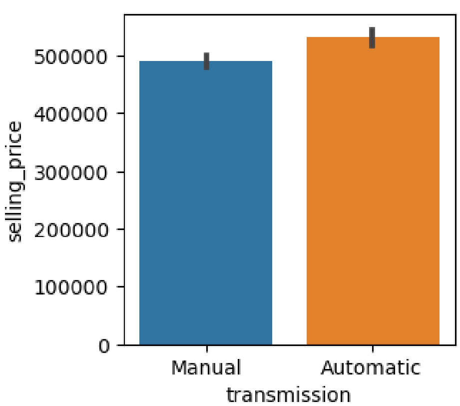


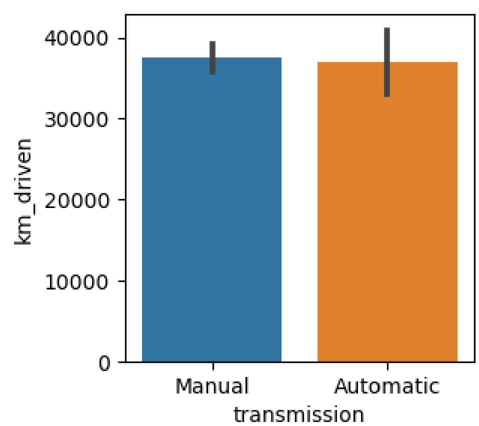
- We can see, Diesel vehicles are having maximum selling price than others.
- Whereas petrol vehicles are having lowest selling price than others. But all are having very little difference in their selling price.



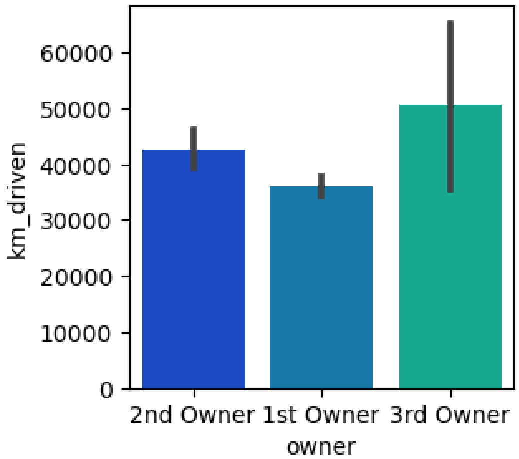


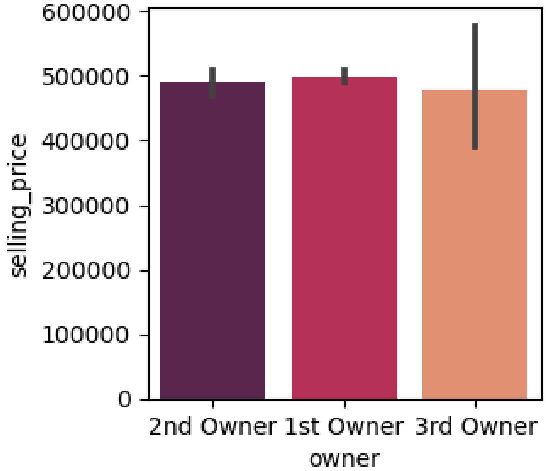
- We can see, Diesel cars are having maximum km_driven. It means that, diesel cars runing km is high they used more. Petrol cars are having lowest km_driven. But here is little contrast that, generally those cars driven less km they are having high price but here is opposite.
- We can see, manual and automatic both type of transmission is having almost same km_driven.
- But Automatic transmission little lower than Manual transmission cars.





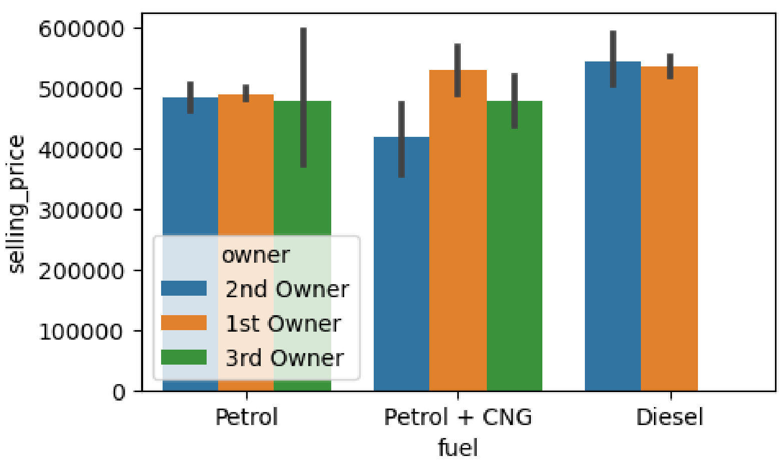
- We can see, Automatic transmission cars are having higher selling price than Manual transmission cars. But there is no major difference in selling price of both transmission cars. We can see, up to the range 0 to 450000 both are having same price.
- We can see, both transmissions are having almost same km driven.





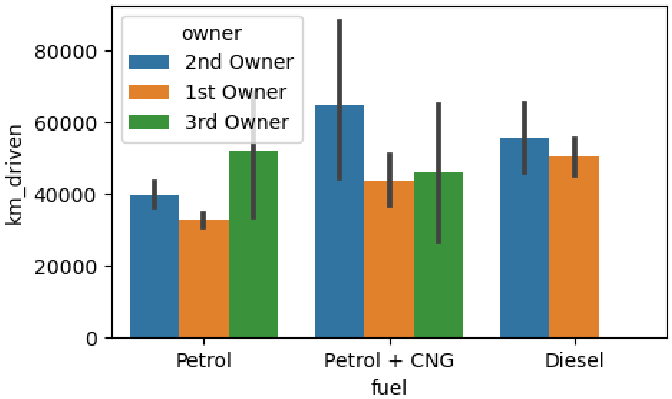
- We can see, 3rd owner cars are having maximum km_driven. It obviously because these are cars owned by multiple owners and they drive these cars more. 1st owner cars are having less km driven.
- We can see, 1st owner cars are having maximum selling price than others. But all are having almost same range of selling price.

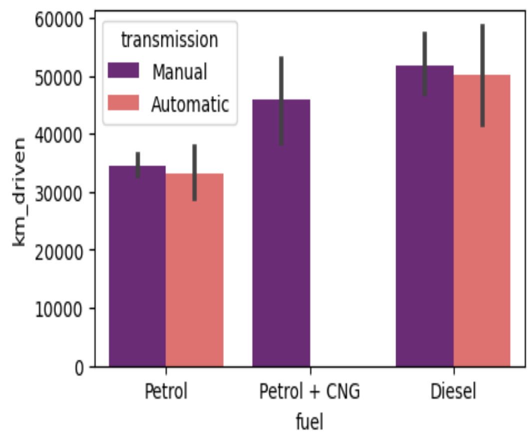
3.2 Multivariate Visualization



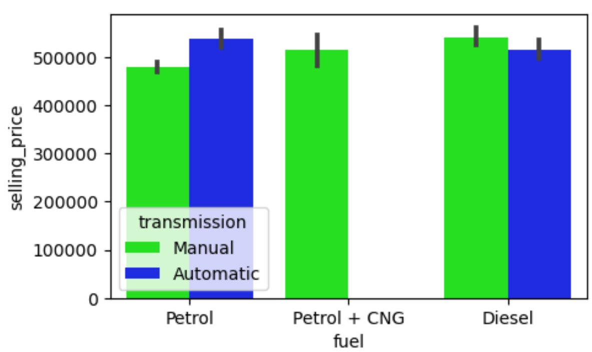
- We can see, in all variants of fuel, 1st owner are having maximum cars selling price than others owner.

- But in Petrol fuel variant, all owners are having almost same selling price there is no major difference in selling price.



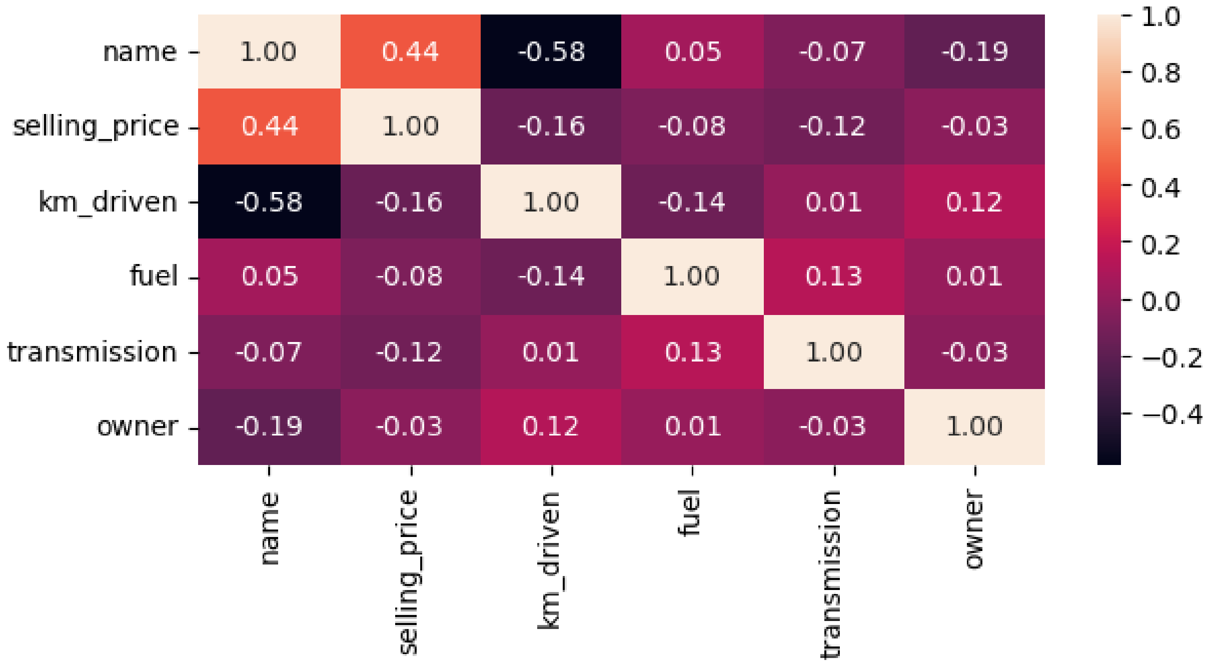


- 2nd owner is top in petrol + CNG and diesel variants of car. It means in these two variants of fuel in which 2nd owner are drive a cars maximum km than others. But in Petrol fuel cars, in which 3rd owner cars are having maximum km driven.
- We can see, in all fuel type, Manual transmission cars are having maximum km_driven. But in Petrol+CNG variants cars, there is Automatic transmission cars.



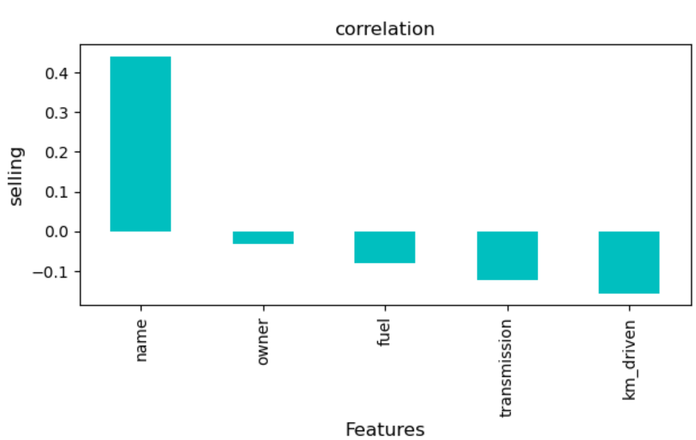
- We can see, In Petrol fuel type, In which automatic transmission cars are having maximum selling price than others.
- But petrol+CNG fuel type, there is no automatic transmission cars.

Correlation of the features with target columns



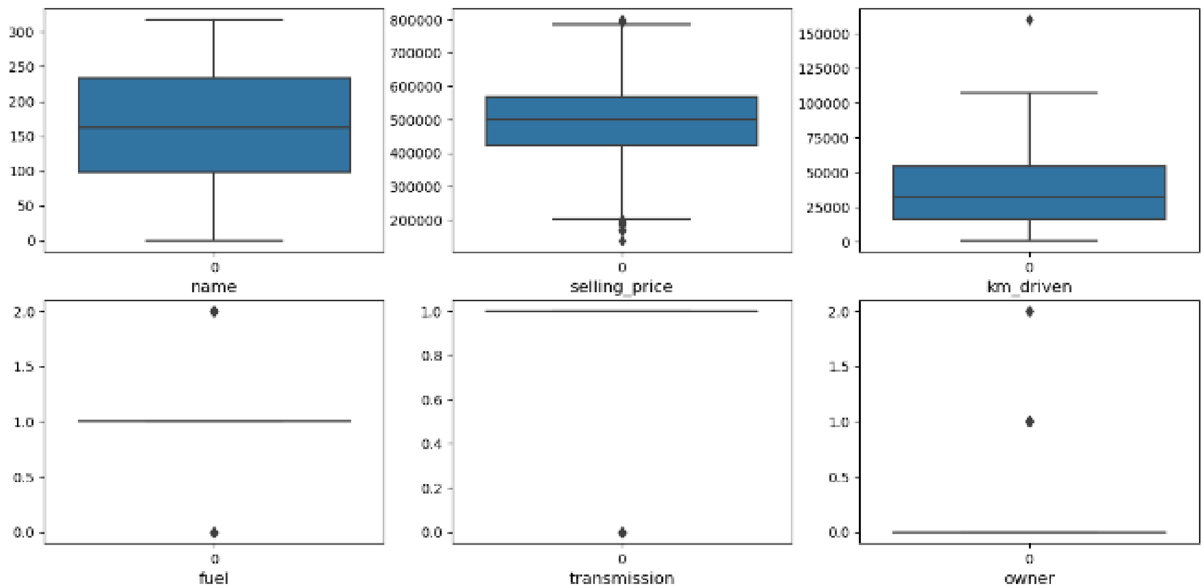
- name has 44% positive correlation with target column.
- km_driven has 16% negative correlation with target column.
- fuel has 8% negative correlation with target column.
- transmission has 12% negative correlation with target column.
- owner has 3% negative correlation with target columns.

Visualizing correlation of feature columns with label column.



- Name have the strongest positive correlation with Selling Price.
- While km_dricven, transmission, fuel have the strongest negative correlation with Selling Price.

Checking Outliers



- We can see, selling price and km_driven are having some outliers.

Removing Outliers

```
from scipy.stats import zscore

z_score = zscore(data[['km_driven']])
abs_z_score = np.abs(z_score)    # Apply the formula and get the scaled data

filtering_entry = (abs_z_score  < 3).all(axis=1)

df = data[filtering_entry]
```

Percentage of data loss

```
data_loss = ((952 - 951)/952*100)
print(data_loss,'%')
```

```
0.10504201680672269 %
```

4. Models Development and Evaluation

4.1 The model algorithms used

Checking Multicollinearity

```
x = df.drop(columns=['selling_price'],axis=1)
y = df['selling_price']
```

```
from sklearn.preprocessing import StandardScaler
```

```
scaler= StandardScaler()
scaled_X = scaler.fit_transform(x)
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
vif = pd.DataFrame()
```

```
vif["Features"] = x.columns
vif['vif'] = [variance_inflation_factor(scaled_X,i) for i in range(scaled_X.shape[1])]
vif
```

	Features	vif
0	name	1.564957
1	km_driven	1.546311
2	fuel	1.037484
3	transmission	1.026349
4	owner	1.039353

Linear Regression:

Finding Best Random State

```
# finding Best Random state
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=i)
    lr=LinearRegression()
    lr.fit(X_train, y_train)
    pred = lr.predict(X_test)
    r2 = r2_score(y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best r2 score is", maxAccu,"on Random State", maxRS)
```

```
Best r2 score is 0.32928243359120646 on Random State 105
```

Train and Test

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=105)
```

```
regressors = {
    'Linear Regression' : LinearRegression(),

}

results=pd.DataFrame(columns=['MAE','MSE', 'RMSE', 'R2-score'])

for method,func in regressors.items():
    model = func.fit(X_train,y_train)
    pred = model.predict(X_test)
    results.loc[method]= [np.round(mean_absolute_error(y_test,pred),3),
                          np.round(mean_squared_error(y_test,pred),3),
                          np.sqrt(mean_squared_error(y_test,pred)),
                          np.round(r2_score(y_test,pred),3)

                         ]
```

```
results
```

	MAE	MSE	RMSE	R2-score
Linear Regression	89860.304	1.198599e+10	109480.545129	0.329

Cross Validation of the Model

```
y_pred = lr.predict(X_test)
from sklearn.model_selection import cross_val_score
lss  = r2_score(y_test,y_pred)
```

```
for j in range(4,10):
    isscore = cross_val_score(lr,x,y,cv=j)
    lsc = isscore.mean()
    print("At cv:-",j)
    print('Cross validation score is:-',lsc*100)
    print('accuracy_score is:-',lss*100)
    print('\n')
```

```
At cv:- 4
Cross validation score is:- 16.855127519323293
accuracy_score is:- 34.069082997475334


At cv:- 5
Cross validation score is:- 16.259158175192717
accuracy_score is:- 34.069082997475334


At cv:- 6
Cross validation score is:- 15.471078887020115
accuracy_score is:- 34.069082997475334


At cv:- 7
Cross validation score is:- 18.04443605028853
accuracy_score is:- 34.069082997475334


At cv:- 8
Cross validation score is:- 17.526336158085495
accuracy_score is:- 34.069082997475334


At cv:- 9
Cross validation score is:- 16.751168951789193
accuracy_score is:- 34.069082997475334
```

```
lsscore_selected = cross_val_score(lr,x,y,cv=7).mean()
print("The cv score is: ",lsscore_selected,"\nThe accuracy score is: ",lss)
```

```
The cv score is:  0.1804443605028853
The accuracy score is:  0.34069082997475334
```

Random Forest Regressor

Finding Best Random State

```
# finding Best Random state
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=i)
    rf = RandomForestRegressor()
    rf.fit(X_train, y_train)
    pred = rf.predict(X_test)
    r2 = r2_score(y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best r2 score is", maxAccu,"on Random State", maxRS)
```

```
Best r2 score is 0.664022980094553 on Random State 542
```

Train and Test

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=542)
```

```
regressors = {

    'Random Forest' : RandomForestRegressor(),

}

results=pd.DataFrame(columns=['MAE','MSE', 'RMSE', 'R2-score'])

for method,func in regressors.items():
    model = func.fit(X_train,y_train)
    pred = model.predict(X_test)
    results.loc[method]= [np.round(mean_absolute_error(y_test,pred),3),
                          np.round(mean_squared_error(y_test,pred),3),
                          np.sqrt(mean_squared_error(y_test,pred)),
                          np.round(r2_score(y_test,pred),3)

                         ]
```

```
results
```

	MAE	MSE	RMSE	R2-score
Random Forest	59172.31	5.849114e+09	76479.502182	0.633

Cross Validation of the Model

```
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
lss  = r2_score(y_test,y_pred)
```

```
for j in range(4,10):
    isscore = cross_val_score(rf,x,y,cv=j)
    lsc = isscore.mean()
    print("At cv:-",j)
    print('Cross validation score is:-',lsc*100)
    print('accuracy_score is:-',lss*100)
    print('\n')
```

```
At cv:- 4
Cross validation score is:- 46.673356656461806
accuracy_score is:- 65.14786396013046


At cv:- 5
Cross validation score is:- 46.323078437931
accuracy_score is:- 65.14786396013046


At cv:- 6
Cross validation score is:- 48.63362656825655
accuracy_score is:- 65.14786396013046


At cv:- 7
Cross validation score is:- 53.83785794067488
accuracy_score is:- 65.14786396013046


At cv:- 8
Cross validation score is:- 53.42495358398573
accuracy_score is:- 65.14786396013046


At cv:- 9
Cross validation score is:- 52.776514495211735
accuracy_score is:- 65.14786396013046
```

```
lsscore_selected = cross_val_score(rf,x,y,cv=7).mean()
print("The cv score is: ",lsscore_selected,"\nThe accuracy score is: ",lss)
```

```
The cv score is:  0.5353572566331872
The accuracy score is:  0.6514786396013046
```

Gradient Boost Regressor

Finding Best Random State

```
# finding Best Random state
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=i)
    gbr = GradientBoostingRegressor()
    gbr.fit(X_train, y_train)
    pred = gbr.predict(X_test)
    r2 = r2_score(y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best r2 score is", maxAccu,"on Random State", maxRS)
```

```
Best r2 score is 0.6128833242011187 on Random State 30
```

Train and Test

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=30)
```

```
regressors = {

    'Gradient Boost Regressor' : GradientBoostingRegressor(),

}

results=pd.DataFrame(columns=['MAE','MSE', 'RMSE', 'R2-score'])

for method,func in regressors.items():
    model = func.fit(X_train,y_train)
    pred = model.predict(X_test)
    results.loc[method]= [np.round(mean_absolute_error(y_test,pred),3),
                          np.round(mean_squared_error(y_test,pred),3),
                          np.sqrt(mean_squared_error(y_test,pred)),
                          np.round(r2_score(y_test,pred),3)

                         ]
```

```
results
```

	MAE	MSE	RMSE	R2-score
Gradient Boost Regressor	63310.362	6.953908e+09	83390.095749	0.616

Cross Validation of the Model

```
gbr = GradientBoostingRegressor()
gbr.fit(X_train,y_train)
y_pred = gbr.predict(X_test)
from sklearn.model_selection import cross_val_score
lss  = r2_score(y_test,y_pred)
```

```
for j in range(4,10):
    isscore = cross_val_score(gbr,x,y,cv=j)
    lsc = isscore.mean()
    print("At cv:-",j)
    print('Cross validation score is:-',lsc*100)
    print('accuracy_score is:-',lss*100)
    print('\n')
```

```
At cv:- 4
Cross validation score is:- 43.54856016836865
accuracy_score is:- 61.4268252375739


At cv:- 5
Cross validation score is:- 42.369607307845015
accuracy_score is:- 61.4268252375739


At cv:- 6
Cross validation score is:- 43.66046694954064
accuracy_score is:- 61.4268252375739


At cv:- 7
Cross validation score is:- 45.828478228894035
accuracy_score is:- 61.4268252375739


At cv:- 8
Cross validation score is:- 45.3616157244546
accuracy_score is:- 61.4268252375739


At cv:- 9
Cross validation score is:- 45.297572721404855
accuracy_score is:- 61.4268252375739
```

```
lsscore_selected = cross_val_score(gbr,x,y,cv=7).mean()
print("The cv score is: ",lsscore_selected,"\nThe accuracy score is: ",lss)
```

```
The cv score is:  0.4580247534879835
The accuracy score is:  0.614268252375739
```

XGBRegressor

Finding Best Random State

```
# finding Best Random state
maxAccu=0
maxRS=0

for i in range(1, 1000):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=i)
    xgb = XGBRegressor()
    xgb.fit(X_train, y_train)
    pred = xgb.predict(X_test)
    r2 = r2_score(y_test, pred)

    if r2>maxAccu:
        maxAccu=r2
        maxRS=i

print("Best r2 score is", maxAccu,"on Random State", maxRS)
```

```
Best r2 score is 0.6861039395010371 on Random State 90
```

Train and Test

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=90)
```

```python
regressors = {

    'XG Boost Regressor' : XGBRegressor()

}

results=pd.DataFrame(columns=['MAE','MSE', 'RMSE', 'R2-score'])

for method,func in regressors.items():
    model = func.fit(X_train,y_train)
    pred = model.predict(X_test)
    results.loc[method]= [np.round(mean_absolute_error(y_test,pred),3),
                          np.round(mean_squared_error(y_test,pred),3),
                          np.sqrt(mean_squared_error(y_test,pred)),
                          np.round(r2_score(y_test,pred),3)

                         ]
```

```python
results
```

	MAE	MSE	RMSE	R2-score
XG Boost Regressor	52167.006	5.300535e+09	72804.775196	0.686

Cross Validation of the Model

```python
xgb = XGBRegressor()
xgb.fit(X_train,y_train)

y_pred = xgb.predict(X_test)
lss  = r2_score(y_test,y_pred)
```

```python
for j in range(4,10):
    isscore = cross_val_score(xgb,x,y,cv=j)
    lsc = isscore.mean()
    print("At cv:-",j)
    print('Cross validation score is:-',lsc*100)
    print('accuracy_score is:-',lss*100)
    print('\n')
```

```
At cv:- 4
Cross validation score is:- 46.59685942584823
accuracy_score is:- 68.61039395010371


At cv:- 5
Cross validation score is:- 48.999130625931784
accuracy_score is:- 68.61039395010371


At cv:- 6
Cross validation score is:- 47.274640915976775
accuracy_score is:- 68.61039395010371


At cv:- 7
Cross validation score is:- 55.136781748027296
accuracy_score is:- 68.61039395010371


At cv:- 8
Cross validation score is:- 51.980030269951925
accuracy_score is:- 68.61039395010371


At cv:- 9
Cross validation score is:- 53.13968174882427
accuracy_score is:- 68.61039395010371
```

```python
lsscore_selected = cross_val_score(xgb,x,y,cv=7).mean()
print("The cv score is: ",lsscore_selected,"\nThe accuracy score is: ",lss)
```

```
The cv score is:  0.551367817480273
The accuracy score is:  0.6861039395010371
```

Regularization

```python
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
```

```python
parameters = {'alpha':[0.0001,0.001,0.01,0.1,1 ,10],
              'random_state':list(range(0,10))}

ls = Lasso()
clf = GridSearchCV(ls,parameters)
clf.fit(X_train,y_train)
clf.best_params_
```

```
{'alpha': 10, 'random_state': 0}
```

```python
ls = Lasso(alpha=10,random_state=0)
ls.fit(X_train,y_train)
ls_score_training = ls.score(X_train,y_train)
pred_ls = ls.predict(X_test)
ls_score_training*100
```

```
24.972222899017915
```

```python
pred = r2_score(y_test,pred_ls)
pred*100
```

```
21.436761446284834
```

```python
pred = r2_score(y_test,pred_ls)
pred*100
```

```
21.436761446284834
```

```python
cv_score = cross_val_score(ls,x,y,cv = 4)
cv_mean = cv_score.mean()
cv_mean*100
```

```
16.859385486345232
```

4.3 Interpretation of the results

Based on comparing Accuracy Score results with Cross Validation results, it is determined XGboost Regressor is the best model. It has least difference between accuracy score and cross validation score.

4.4 Hyperparameter Tuning

```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

```python
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.7, test_size=0.3, random_state=90)
```

```python
param = {'learning_rate':[0.1,0.2,0.3],
        'n_estimators':[150,200,300],
        'max_depth':[5,10,15],
        'min_child_weight':[7,9,11],
        'gamma':[0,0.1,0.2,0.3],
        'colsample_bytree':[0.3,0.4,0.5]}
```

```python
grd = GridSearchCV(xgb,param_grid=param)
```

```python
grd.fit(X_train,y_train)

grd.best_params_
```

```
{'colsample_bytree': 0.3,
 'gamma': 0,
 'learning_rate': 0.3,
 'max_depth': 15,
 'min_child_weight': 9,
 'n_estimators': 300}
```

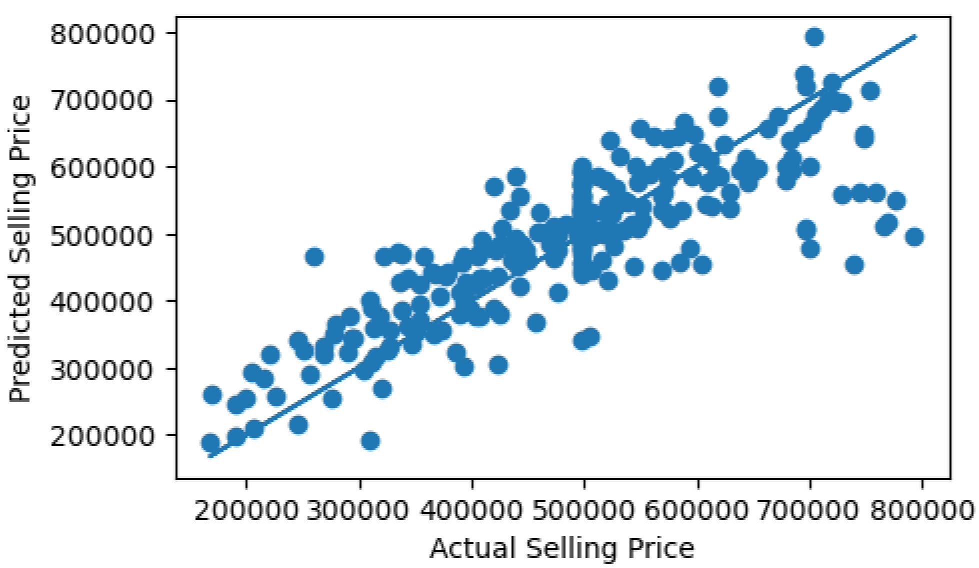
```
xgb = XGBRegressor(colsample_bytree=1, gamma=0, max_depth=6, min_child_weight=1, n_estimators=100,learning_rate=0.3)
xgb.fit(X_train,y_train)

y_pred = xgb.predict(X_test)
r2_score(y_test,y_pred)
```

```
0.6861039395010371
```

Based on the input parameter values and after fitting the train datasets The XGBoost Regressor model was further tuned based on the parameter values yielded from GridSearchCV. The XGBoost Regressor model displayed an accuracy of 68.62%.

Scatter plot Between Actual and Predicted Selling Price of Car



The Model Saving and Testing

```python
import joblib
joblib.dump(xgb,"car_price_prediction.pkl")
```

```
['car_price_prediction.pkl']
```

Loading The Model

```python
mod=joblib.load("car_price_prediction.pkl")
```

```python
print(mod.predict(x))
```

```
[505247.8  508688.62 565316.8  554702.56 179277.72 615746.2  628873.56
 639082.4  290217.3  508320.72 492352.1  496635.62 666270.2  270391.06
 404146.66 572504.25 268726.44 500658.97 259172.34 595106.56 491993.38
 499692.2  590961.   302134.28 504404.75 514221.7  604334.2  724937.9
 496275.34 742477.9  596583.6  793772.5  457253.5  261143.12 511697.53
 457253.5  697010.   600113.44 275317.4  766250.44 651443.56 190787.64
 389142.56 519053.84 543740.9  187674.88 586346.44 499101.62 437522.3
 497497.66 498132.6  604972.06 318435.8  551047.7  650402.4  289433.66
 381667.97 509927.4  496461.88 440487.94 608507.56 514095.03 506498.94
 516997.2  654065.25 285201.8  392083.78 515357.   599626.3  451221.25
 498307.56 376189.3  502381.38 438388.4  268777.38 711850.06 268932.72
 579217.   496580.2  506425.47 219020.27 519644.2  498544.7  255730.77
 264741.88 659197.6  405748.47 501339.1  367508.66 618576.06 562280.2
 497354.22 382474.2  578322.1  193096.53 731969.9  502535.94 503146.5
 543251.5  602611.56 292189.97 395816.12 499543.78 492286.34 511855.38
 259993.28 504435.44 498498.7  450555.16 768636.1  505812.56 539538.2
 365862.34 529113.4  591282.4  582267.25 393936.6  721603.56 340173.16
 360150.2  520935.1  504422.   500545.53 535478.7  613063.9  446166.53
```

```python
Prediction_accuracy = pd.DataFrame({'Predictions': mod.predict(x), 'Actual Values': y})
Prediction_accuracy.head(30)
```

	Predictions	Actual Values
0	505247.81250	534399.0
1	508688.62500	546599.0
2	565316.81250	555899.0
3	554702.56250	557199.0
4	179277.71875	170699.0
5	615746.18750	625999.0
6	628873.56250	631199.0
7	639082.37500	523099.0
8	290217.31250	256699.0
9	508320.71875	509899.0
10	492352.09375	498299.0
11	496635.62500	491499.0
12	666270.18750	588899.0
13	270391.06250	281499.0
14	404146.65625	401199.0

5. Conclusions

5.1 Key Finding and Conclusions

The main component on which the price of a car depends is the engine size, the year which car was bought, the mileage on the car etc.

The price also depends on which city the car was registered, as some cities have different tax rates and restrictions.

XGBRegressor works best for this particular data set, hyper parameter tuning was performed, and optimal parameters were found.

EDA is very powerful in understanding the data and pre-processing it before feeding it to the algorithm. Statistical methods work the best.

5.2 Limitation of this works and scope for future works

Post covid-19 car market is still evolving, and it will keep evolving for the foreseeable future. The algorithms will need to keep changing to keep up with the evolution.