
# Used Car Price Prediction

Submitted by:

Swati Pandey

# ACKNOWLEDGMENT

I have taken efforts in this project. I am highly indebted to (Flip Robo Technologies) for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

A huge thanks to “DataTrained”, because of that I got this opportunity to work with FlipRobo. Lastly, I would like to thank my family, who have always been my support in every step of my life.

Based on the business requirements of the Client, I have scraped the Data from the well-known e-commerce websites such as cars 24, OLX and cardekho.com. Based on the Data collected, I will be predicting the prices of used cars. We will be building various Machine Learning models. In the end, we will see how all the machine learning models performs. And based on which we will sort the best machine learning model and hyperparameter tune the same to get the improved performance.

# Content

## Business Problem Framing

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

## Conceptual Background of the Domain Problem

The Indian automotive sector was already struggling in FY20. before the Covid-19 crisis. It saw an overall degrowth of nearly 18 per cent. This situation was worsened by the onset of the Covid-19 pandemic and the ongoing lockdowns across India and the rest of the world. These two years (FY20 and FY21) are challenging times for the Indian automotive sector on account of slow economic growth, negative consumer sentiment, BS-VI transition, changes to the axle load norms, liquidity

crunch, low-capacity utilization and potential bankruptcies.

The return of daily life and manufacturing activity to near normalcy in China and South Korea, along with extended lockdown in India, gives hope for a U- shaped economic recovery. Our analysis indicates that the Indian automotive sector will start to see recovery in the third quarter of FY21. We expect the industry demand to be down 15-25 per cent in FY21. With such degrowth, OEMs, dealers and suppliers with strong cash reserves and better access to capital will be better positioned to sail through.

**Postponement** of new vehicle purchases by customers and reduced exports are likely to result in the aftermarket receiving increased attention from auto component suppliers.

Auto sector has been under pressure due to a mix of demand and supply factors. However, there are also some positive outcomes, which we shall look at.

- With India's GDP growth rate for FY21 being downgraded from 5% to 0% and later to (-5%), the auto sector will take a hit. Auto demand is highly sensitive to job creation and income levels and both have been impacted. CII has estimated the revenue impact at $2 billion on a monthly basis across the auto industry in India.
- Supply chain could be the worst affected. Even as China recovers, supply chain disruptions are likely to last for some more time. The problems on the Indo-China border at Ladakh are not helping matters. Domestic suppliers are chipping in but they will face an inventory surplus as demand remains tepid.
- The Unlock 1.0 will coincide with the implementation of the BS-VI norms and that would mean heavier discounts to dealers and also to customers. Even as auto companies are managing costs, the impact of discounts on profitability is going to be fairly steep.
- The real pain could be on the dealer end with most of them struggling with excess inventory and lack of funding options in the post COVID-19 scenario. The BS-VI price increases are also likely to hit auto demand.

There are two positive developments emanating from COVID- 19. The China supply chain shock is forcing major investments in the "Make in India" initiative. The COVID-19 crisis has exposed chinks in the automobile business model and it could catalyze a big move towards electric vehicles (EVs). That could

be the big positive for auto sector.

## Review of Literature

- As per the requirement of our client, I have scrapped data from different used cars selling merchants websites, and so based on the data collected, I have tried to analyze that based on what factors the used car price is decided? What is the relationship between cost of the used cars and other factors like Fuel type, Brand and Model, Variant Manufacturing year and No. Of owners, Transmission, Driven Km, and before selling? And so based on all the above consideration I have developed a model that will predict the price of the used cars.

## Motivation for the Problem Undertaken

I have taken this problem based on the requirement of the client and also, with a curiosity to know how the used cars markets are at the time of pandemic.

# Analytical Problem Framing

## Mathematical/ Analytical Modelling of the Problem

Since target or output variable for prediction is known, so this problem comes under Supervised learning, since label contains continuous data, so this a Regression analysis problem.

I have considered following algorithms for analytical modelling:

**Linear Regression**: Linear regression is one of the most basic **types of regression in machine learning**. The linear regression model consists of a variable and a dependent variable related linearly to each other. In case the data involves more than one independent variable, then linear regression is called multiple linear regression models.

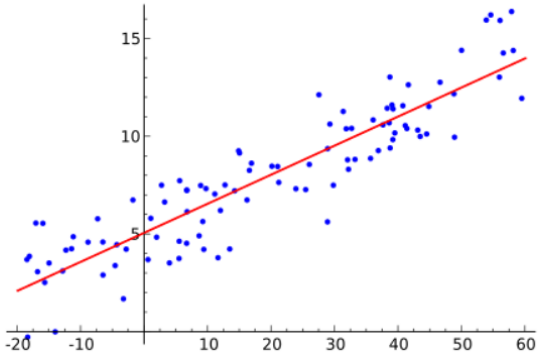
*The below-given equation is used to denote the linear regression model:*

$$y=mx+c+e$$

where m is the slope of the line, c is an intercept, and e represents the error in the model.

The best fit line is determined by varying the values of m and c. The predictor error is the difference between the observed values and the predicted value. The values of m and c get selected in such a way that it gives the minimum predictor error. It is important to note that

a simple linear regression model is susceptible to outliers. Therefore, it should not be used in case of big size data.




**Ridge Regression:** This is another one of the **types of regression in machine learning** which is usually used when there is a high correlation between the independent variables. This is because, in the case of multi collinear data, the least square estimates give unbiased values. But, in case the collinearity is very high, there can be some bias value. Therefore, a bias matrix is introduced in the equation of Ridge Regression. This is a powerful regression method where the model is less susceptible to overfitting.

Below is the equation used to denote the Ridge Regression, where the introduction of λ (lambda) solves the problem of multicollinearity:

$$\beta = (X^{T}X + \lambda*I) ^{-1}X^{T}y$$

**Lasso:** Lasso Regression is one of the **types of regression in machine learning** that performs regularization along with feature selection. It prohibits the absolute size of the regression coefficient. As a result, the coefficient value gets nearer to zero, which does not happen in the case of Ridge Regression.
Due to this, feature selection gets used in Lasso Regression, which allows selecting a set of features from the dataset to build the model. In the case of Lasso Regression, only the required features are used, and the other ones are made zero. This helps in avoiding the overfitting in the model. In case the independent variables are highly collinear, then Lasso regression picks only one variable and makes other variables to shrink to zero.

$$N^ {-1} \Sigma^{N}_{i=1} f(x_{i}, y_{I}, \alpha, \beta)$$

**Decision Tree Regressor:** **Decision Tree** is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility.

Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

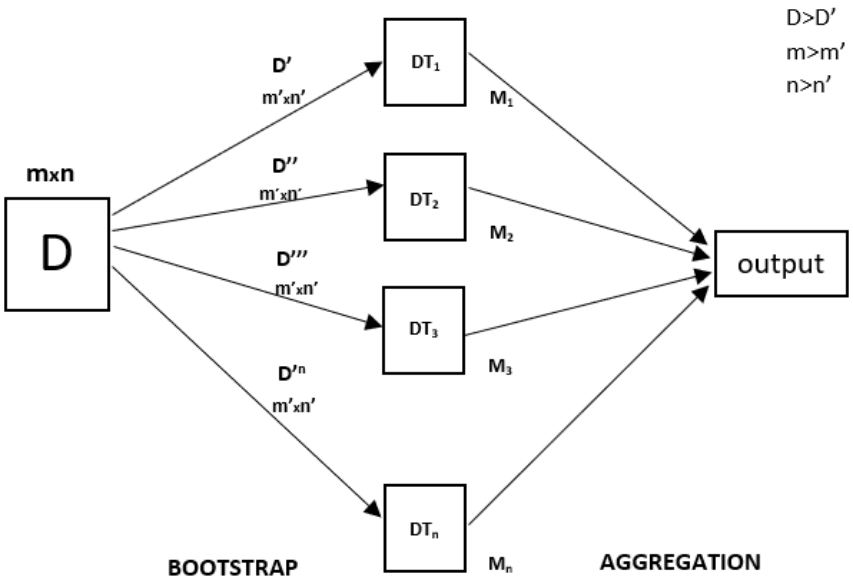
The branches/edges represent the result of the node and the nodes have either:

1. Conditions [Decision Nodes]
2. Result [End Nodes]

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

**Random Forest Regressor:**

Every decision tree has high variance, but when we combine all of them together in parallel then the resultant variance is low as each decision tree gets perfectly trained on that particular sample data and hence the output doesn't depend on one decision tree but multiple decision trees. In the case of a classification problem, the final output is taken by using the majority voting classifier. In the case of a regression problem, the final output is the mean of all the outputs. This part is Aggregation.




**Ada Boost Regressor:** Boosting is a class of ensemble machine learning algorithms that involve combining the predictions from many weak learners.

A weak learner is a model that is very simple, although has some skill on the dataset. Boosting was a theoretical concept long before a practical algorithm could be developed,

and the AdaBoost (adaptive boosting) algorithm was the first successful approach for the idea.

AdaBoost combines the predictions from short one-level decision trees, called decision stumps, although other algorithms can also be used. Decision stump algorithms are used as the AdaBoost algorithm seeks to use many weak models and correct their predictions by adding additional weak models.

## Data Sources and their formats

The Data is scrapped from multiple ecommerce websites that sells used cars in India, Websites like cars 24, OLX and car Dekho. These data are scapped and stored in a CSV format. Data contains following columns.

1. 'LOCATION' – It will tell which location the car is sold.
2. 'MNF_YEAR' – At what year the car is manufactured
3. 'BRAND' – Brand is manufacturer or which company made
4. 'MODEL' – It is basically the model of the car.
5. 'Variant' - Variant is Description of Model. For ex:
6. TRANSMISSION – Gear shift variant is (Automatic, Manual, Semi-Automatic)
7. 'DRIVEN_KM' – no of Kms driven before selling
8. 'FUELTYPE' – Petrol, diesel, CNG, LPG, Electric
9. 'NO_OF_OWNERS' – 1st, 2nd or 3rd owner
10. 'PRICE' – our target variable that tells what is the price of the used car.

## Data Pre-processing:

Steps:

a) I have imported all the necessary libraries.
b) Loaded and read the csv file using pandas Data Frame.
c) I have checked dataset size, columns datatype, unique values of each column datatype.
d) I have checked for null values. There are 81 null values present in Transmission column.

```
In [44]: ## Checking null values:
         df.isnull().sum()

Out[44]: LOCATION        0
         MNF_YEAR        0
         BRAND           0
         MODEL           0
         VARIANT         0
         TRANSMISSION    81
         DRIVEN_KM       0
         FUELTYPE        0
         NO_OF_OWNERS    0
         PRICE           0
         dtype: int64
```

Since TRANSMISSION is a categorical column.so filling null values with mode in TRANSMISSION column:

```
In [176]: ## Since  TRANSMISSION is a categorical column.so filling null values with mode in TRANSMISSION column:
          df['TRANSMISSION'].fillna(df['TRANSMISSION'].mode()[0],inplace=True)

In [177]: df.isnull().sum()

Out[177]: LOCATION        0
          MNF_YEAR        0
          BRAND           0
          MODEL           0
          VARIANT         0
          TRANSMISSION    0
          DRIVEN_KM       0
          FUELTYPE        0
          NO_OF_OWNERS    0
          PRICE           0
          dtype: int64

          Now There is not any null values are present in dataset.
```

e) During analysis I found that DRIVEN_KM and PRICE have km and Rs. appended before, So I have formatted that and convert their datatype to int.

```
df['DRIVEN_KM'] = df['DRIVEN_KM'].astype(int)
```

```
df['DRIVEN_KM'].dtypes
```

```
df['PRICE'] = pp
df['PRICE'] =df['PRICE'].astype(int)
```

```
df['PRICE'].dtypes
```

```
dtype('int32')
```

f) Encode categorical features using LabelEncoder.

```
## Label Encoding:
columns = ['MNF_YEAR','LOCATION','BRAND','MODEL','VARIANT','TRANSMISSION','FUELTYPE','NO_OF_OWNERS']
lec = LabelEncoder()
for i in columns:
    df[i] = df[i].astype('str')
    df[i]=  lec.fit_transform(df[i])

```

```
df
```

| | LOCATION | MNF_YEAR | BRAND | MODEL | VARIANT | TRANSMISSION | DRIVEN_KM | FUELTYPE | NO_OF_OWNERS | PRICE |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7 | 8 | 12 | 94 | 497 | 1 | 101529 | 0 | 0 | 655999 |
| **1** | 7 | 8 | 12 | 94 | 424 | 1 | 59773 | 0 | 0 | 666399 |
| **2** | 7 | 9 | 12 | 94 | 424 | 1 | 35231 | 0 | 0 | 684399 |
| **3** | 7 | 5 | 4 | 30 | 101 | 1 | 80946 | 0 | 1 | 409899 |
| **4** | 7 | 8 | 4 | 30 | 86 | 1 | 16212 | 1 | 0 | 741699 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **8663** | 0 | 11 | 12 | 31 | 172 | 1 | 26880 | 1 | 0 | 459599 |
| **8664** | 0 | 11 | 12 | 31 | 172 | 1 | 11882 | 1 | 0 | 461999 |
| **8665** | 0 | 9 | 4 | 30 | 101 | 1 | 71900 | 0 | 1 | 616799 |
| **8666** | 0 | 10 | 12 | 94 | 419 | 1 | 27460 | 0 | 0 | 816199 |
| **8667** | 0 | 12 | 4 | 30 | 101 | 1 | 15951 | 0 | 0 | 1039699 |

8668 rows × 10 columns

g) Checked correlation using heatmap.

## Correlation

```
plt.figure(figsize =(30,30))
sns.heatmap(df.corr(),annot = True,cmap ='RdYlGn')
plt.show()
```

h) Detected outliers using **distplot** and **boxplot.**

i) Removed skewness using Power Transformer, method is ‘yeo-johnson’.

```
df.skew()
```

```
LOCATION        -0.403131
MNF_YEAR        -0.462373
BRAND            0.190042
MODEL            0.241976
VARIANT         -0.330499
TRANSMISSION    -1.594060
DRIVEN_KM       -1.402540
FUELTYPE        -0.220277
NO_OF_OWNERS     2.083213
PRICE            1.811769
dtype: float64
```

```
#Except DRIVEN_KM and PRICE ,all are categorical columns. PRICE is target variable,so will remove skewness from DRIVEN_KM.
from sklearn.preprocessing import PowerTransformer
skewed_features = ['DRIVEN_KM']
scaler = PowerTransformer(method='yeo-johnson')
```

```
df[skewed_features] = scaler.fit_transform(df[skewed_features].values)
df[skewed_features].head()
```

| | DRIVEN_KM |
|---|---|
| 0 | 1.400642 |
| 1 | 0.552685 |
| 2 | -0.163707 |
| 3 | 1.021060 |
| 4 | -1.012395 |

j) Separated Features and Target

k) Standardize data using Standard Scaler.

## Generalizing data using Standard Scaler

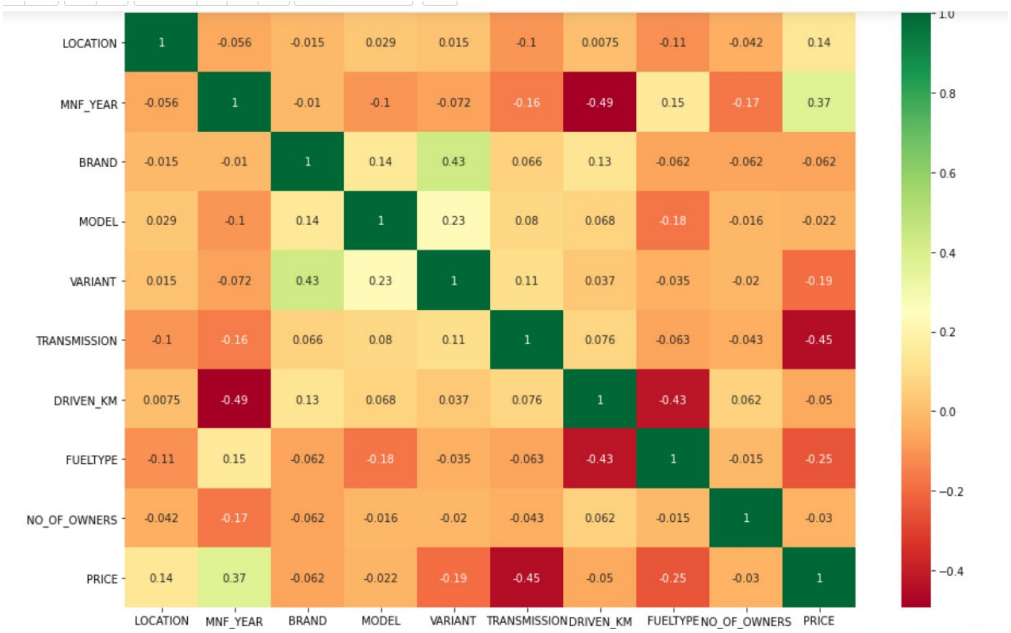
```
scaler = StandardScaler()
x=pd.DataFrame(scaler.fit_transform(x),columns=x.columns)
x
```

| | LOCATION | MNF_YEAR | BRAND | MODEL | VARIANT | TRANSMISSION | DRIVEN_KM | FUELTYPE | NO_OF_OWNERS |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.536716 | -0.152979 | 0.447871 | 1.324209 | 1.264121 | 0.481794 | 1.400642 | -1.214091 | -0.469389 |
| 1 | 0.536716 | -0.152979 | 0.447871 | 1.324209 | 0.793740 | 0.481794 | 0.552685 | -1.214091 | -0.469389 |
| 2 | 0.536716 | 0.291995 | 0.447871 | 1.324209 | 0.793740 | 0.481794 | -0.163707 | -1.214091 | -0.469389 |
| 3 | 0.536716 | -1.487899 | -1.337846 | -0.687676 | -1.287537 | 0.481794 | 1.021060 | -1.214091 | 1.743043 |
| 4 | 0.536716 | -0.152979 | -1.337846 | -0.687676 | -1.384191 | 0.481794 | -1.012395 | 0.754435 | -0.469389 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 8663 | -1.894292 | 1.181941 | 0.447871 | -0.656240 | -0.830043 | 0.481794 | -0.485204 | 0.754435 | -0.469389 |
| 8664 | -1.894292 | 1.181941 | 0.447871 | -0.656240 | -0.830043 | 0.481794 | -1.292932 | 0.754435 | -0.469389 |
| 8665 | -1.894292 | 0.291995 | -1.337846 | -0.687676 | -1.287537 | 0.481794 | 0.832749 | -1.214091 | 1.743043 |
| 8666 | -1.894292 | 0.736968 | 0.447871 | 1.324209 | 0.761522 | 0.481794 | -0.460878 | -1.214091 | -0.469389 |
| 8667 | -1.894292 | 1.626915 | -1.337846 | -0.687676 | -1.287537 | 0.481794 | -1.027831 | -1.214091 | -0.469389 |

8668 rows × 9 columns

## Data Inputs- Logic- Output Relationships

Following is the Heatmap which shows inputs -output Relationship:




| | PRICE |
|---|---|
| **MNF_YEAR** | 0.373988 |
| **LOCATION** | 0.139629 |
| **MODEL** | -0.021679 |
| **NO_OF_OWNERS** | -0.030471 |
| **DRIVEN_KM** | -0.050042 |
| **BRAND** | -0.062152 |
| **VARIANT** | -0.194721 |
| **FUELTYPE** | -0.253148 |
| **TRANSMISSION** | -0.447728 |

MNF Year and Location have positive relationship with Target. Model,No of owners,driven km,brand, variant, Fuel Type,Transmission have negative co with Target.

## Hardware and Software Requirements and Tools Used

**Hardware**:

- Process: Intel core i5 and above.
- RAM:16GB and above.
- SSD:250GB and above.

**Software & Tools used:**

- Python3.0,
- Jupyter Notebook

**Libraries:**

- **Pandas**: pandas is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.

- **Numpy:** NumPy is the fundamental package for package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation,

sorting, selecting, I/O, discrete Fourier transforms,
basic linear algebra, basic statistical operations,
random simulation and much more.

- **Seaborn**: Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.
- **Matplotlibt**: matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some linesin a plotting area, decorates the plot with labels, etc.

ι. from sklearn.preprocessing import StandardScaler
ιι. from sklearn.preprocessing import PowerTransformer
ιιι. from sklearn.tree import DecisionTreeRegressor
ιϖ. from sklearn.ensemble import RandomForestRegressor
ϖ. from sklearn.linear_model import LinearRegression,Lasso,LassoCV,Ridge,RidgeCV
ϖι. from sklearn.ensemble import AdaBoostRegressor
ϖιι. from sklearn.neighbors import KNeighborsRegressor
ϖιιι. from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
ιξ. from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error ## to get score of our model.
ξ. from sklearn.preprocessing import LabelEncoder ## To encode our categorical variable into numerical variable.
ξι. from sklearn.preprocessing import StandardScaler

# Model/s Development and Evaluation

**Identification of possible problem-solving approaches**: Since this is a regression problem, So I have first tried to find best random states for my regression models.

Finding Best random state

## Model Building

In [201]:
```
## Finding Best Random State
score = 0
for i in range(0,200):
    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = .30, random_state = i)
    rf = RandomForestRegressor()
    rf.fit(x_train,y_train)
    y_pred = rf.predict(x_test)
    tempscore = r2_score(y_test,y_pred)
    if tempscore > score:
        score= tempscore
        best_rstate=i

print(f"Best Accuracy {score*100} found on randomstate {best_rstate}")

```

```
Best Accuracy 98.98878329931033 found on randomstate 42
```

In [202]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = .30, random_state = best_rstate)
```

b) Initialize models and train our models and then predict our test target data and checked all metrics ,so that I can know that how my models are performing and which is best for this data set.

```
lr=LinearRegression()
dtr=DecisionTreeRegressor()
knn=KNeighborsRegressor()
rf=RandomForestRegressor()
AdaBoost=AdaBoostRegressor()
lasso = Lasso()
ridge  = Ridge(alpha=1, random_state=42)
```

```
algo=[lr,dtr,rf,knn,AdaBoost,lasso,ridge]
acc_models={}
for model in algo:
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
    print("-"*60)
    acc_models[model]=round(r2_score(y_test,y_pred)*100,1)
    print(f"The model {model} has:: \n\t Accuracy :: {round(r2_score(y_test,y_pred)*100,1)}% \n\t Mean Absolute Error is ::
    print("-"*100)
    print("\n")
```

## Key Metrics for success in solving problem under consideration

r2 Scores of our models:

```
acc_models
```

```
{LinearRegression(): 42.1,
 DecisionTreeRegressor(): 99.0,
 RandomForestRegressor(): 99.0,
 KNeighborsRegressor(): 82.5,
 AdaBoostRegressor(): 56.4,
 Lasso(): 42.1,
 Ridge(alpha=1, random_state=42): 42.1}
```

```
Maximum r2 score is for DecisionTreeRegressor and RandomForestRegressor.
```

## Testing of Identified Approaches (Algorithms)

Cross Validation of the models

```
for model in algo:
    CVscore_={}
    print(f"\n{model}")
    print("-"*25)
    print("\n")
    for i in range(2,10):
        cvs=cross_val_score(model,x,y,cv=i)
        CVscore_[i]=cvs.mean()
        print(f"Mean CV Score of model {model}:: {cvs.mean()} at k-fold::{i}\n")
    CVdata=pd.DataFrame(CVscore_,index=[""])
    CVmodel[str(model)]=CVdata.max(axis=1).tolist()
```

```
Mean CV Score of model LinearRegression():: 0.3349268715580617 at k-fold::4

Mean CV Score of model LinearRegression():: 0.34380219091641234 at k-fold::5

Mean CV Score of model LinearRegression():: 0.3505076895723003 at k-fold::6

Mean CV Score of model LinearRegression():: 0.36713955047258745 at k-fold::7

Mean CV Score of model LinearRegression():: 0.36374123008508064 at k-fold::8

Mean CV Score of model LinearRegression():: 0.37679186275494286 at k-fold::9


DecisionTreeRegressor()
------------------------


Mean CV Score of model DecisionTreeRegressor():: 0.8083727543806798 at k-fold::2
```

Difference between r2 score and CV score of each model

```
acc_value=list(acc_models.values ( ))

m=list(CVmodel.keys())

print("The least difference between the r2-score and CV score of each model is::\n")
for i in range(5):
    print(f"{m[i]}::{round(np.abs(CVmodel[m[i]][0]*100-acc_value[i]),2)}")
```

```
The least difference between the r2-score and CV score of each model is::

LinearRegression()::3.48
DecisionTreeRegressor()::7.68
RandomForestRegressor()::3.31
KNeighborsRegressor()::8.72
AdaBoostRegressor()::11.56
```

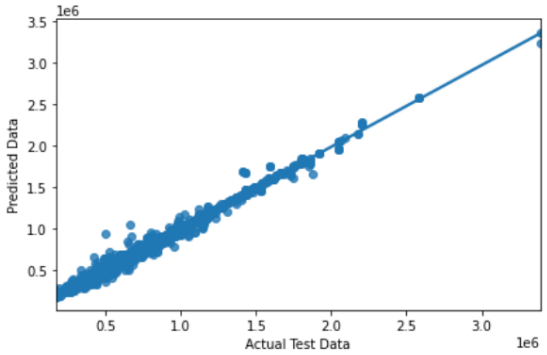
Here for model RandomForest Regressor we get the least value i.e. the difference between the r2-score and cvScore of this model is 0.25 . So,

# Run and evaluate selected models

Regplot for Random Forest Model:

```
# Plotting regplot graph for Random Forest Regressor model
```

```
rf=RandomForestRegressor()
rf.fit(x_train,y_train)
y_pred=rf.predict(x_test)
sns.regplot(y_test,y_pred)
plt.xlabel("Actual Test Data")
plt.ylabel("Predicted Data")
plt.tight_layout()
```




Here we can see that the data points are very close to the best fit line. That means the residual is less.So I'll do hyperparameter tuning for Random Forest.

## g) HyperParameter Tuning of the model:

```
HyperParameter Tuning
```

```
param_grid = [
    {"bootstrap": [True, False],
     "criterion": ["mse", "mae"],
     "n_estimators"      : [10,20,30,50,100],
     "max_features"      : ["auto", "sqrt", "log2"],
     "min_samples_split" : [2,4,8],
    }
]

rf = RandomForestRegressor(random_state=best_rstate)

grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 5, n_jobs = -1, verbose = 2)

grid_search.fit(x_train,y_train)
final_model = grid_search.best_estimator_
final_model
```

```
Fitting 5 folds for each of 180 candidates, totalling 900 fits
```

```
RandomForestRegressor(bootstrap=False, max_features='sqrt', random_state=42)
```

```
## Providing hyperparameters to model:-
rfr = RandomForestRegressor(bootstrap=False,max_features='sqrt', random_state=42 )
rfr.fit(x_train, y_train)
y_pred = rfr.predict(x_test)
score =  r2_score(y_test,y_pred)
score
```

```
After Hyperparameter tuning score has increased.
```

# CONCLUSION

## Conclusion

```
a = np.array(y_test)
predicted = np.array(rfr.predict(x_test))
new_df = pd.DataFrame({"Original":a,"Predicted":predicted},index= range(len(a)))
new_df

```

| | Original | Predicted |
|---|---|---|
| **0** | 585799 | 581552.0 |
| **1** | 951399 | 892896.0 |
| **2** | 723599 | 723599.0 |
| **3** | 672199 | 672199.0 |
| **4** | 669999 | 669999.0 |
| **...** | ... | ... |
| **2596** | 599999 | 616431.0 |
| **2597** | 666999 | 660395.0 |
| **2598** | 495299 | 495299.0 |
| **2599** | 303199 | 303199.0 |
| **2600** | 642099 | 618441.0 |

2601 rows × 2 columns

The above model will help our seller to predict the Price of the used cars, and also will helps them to understand based on factors that affect car's pricing.

The above model will help our seller to predict the Price of the used cars, and also will helps them to understand based on factors that affects used car’s pricing.

## Learning Outcomes

1. Applied quantitative modelling and data analysis techniques to the solution of real- world business problems, communicate findings, and effectively resent results using data visualization techniques.

2. Demonstrate knowledge of statistical data analysis techniques utilized in business decision making.

3. Apply principles of Data Science to the analysis of business problems.

4. Use data mining software to solve real-world problems.

5. Apply algorithms to build machine intelligence.

*** I have found Random Forest Model is giving highest R2 score, among the models, which I have used for predicting the label.

## Limitations

Scrapping from different sites are very time consuming and tedious process, because I have experienced difficulty, after running scrapping scripts, sometimes for whole day or whole

Night, time out error came and , then again run whole scrapping script. So I didn't use large dataset. One more challenge that data is present is different sites in different formats, So make them in one format is also a tedious process.

**Scope for Future Work:**

One can scrap from different sites and collect more and more data with more different parameters.