CAR-PRICE-PREDICTION-PROJECT

SUBMITTED BY: - TEJENDRA SONI

ACKNOWLEDGMENT

I would like to express my thanks of gratitude to SME SWATI MAHASETH as well as Flip Robo who gave me the golden opportunity to do this wonderful project on the topic Micro Credit Defaulter, which also helped me in doing a lot of research and I came to know about so many new things. I am very thankful to them. I am making this project to increase my knowledge.

INTRODUCTION

Problem Statement:

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 must make car price valuation mode

Objective:

Building a model which can be used to predict in terms of a probability for used cars

Firstly, we will start by importing required libraries and databases.

```
In [1]: #Importing needed libraries

        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        import sklearn
        import warnings
        warnings.filterwarnings('ignore')
```

```
In [2]: #Importing the dataframe storing it into df & printing it

        df=pd.read_csv('CAR-PRICE-DATA-FINAL.csv')
        df
```

Out[2]:

	Unnamed: 0	Unnamed: 0.1	PRICE	YEAR	KM	COMPANY	LOCATION	POSTED-DATE	soucefilename
0	0	0	645000.0	2018	41,250	Maruti Suzuki Baleno	Kowdiar\nJan 03	Jan-03	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE01.csv
1	1	1	725000.0	2017	82,000	Maruti Suzuki Swift Dzire	Changubetty\n5 days ago	5 days ago	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE01.csv
2	2	2	13500000.0	2020	5,005	BMW 8 Series	Companypadi\nJan 12	Jan-12	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE01.csv
3	3	3	35000.0	1998	50,000	Maruti Suzuki 800	Pazhakutty\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE01.csv
4	4	4	185000.0	2008	1,41,000	Maruti Suzuki Swift	Palarivattom\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE01.csv
...	...	...	...	...	...	...	...	...	...
5492	495	495	1390000.0	2014	58,000	Audi Q3	Sector 80\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE9.csv
5493	496	496	280000.0	2012	80,000	Maruti Suzuki Ertiga	Gohana Part\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE9.csv
5494	497	497	45000.0	2006	1,18,000	Maruti Suzuki Alto 800	Prakash Nagar\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE9.csv
5495	498	498	325000.0	2017	88,000	Hyundai Xcent	New Grain Market\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE9.csv
5496	499	499	700000.0	2018	73,000	Hyundai i20 Active	Tagra Kali Ram\nToday	Today	E:/TEJ/FLIPROBO/CAR-PRICE/DATAA\CARS-FILE9.csv

5497 rows × 9 columns

Our dataset has 5497 rows and 9 columns.

Let’s check the datatype of all columns:

```
In [7]: #Printing the datatypes of all the columns that are presented here in our dataset

        df.dtypes

Out[7]: Unnamed: 0          int64
        Unnamed: 0.1        int64
        PRICE             float64
        YEAR                int64
        KM                 object
        COMPANY            object
        LOCATION           object
        POSTED-DATE        object
        soucefilename      object
        dtype: object
```

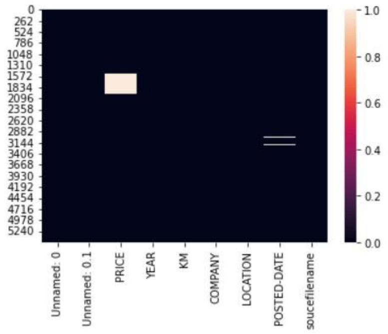
- Also, we can drop column Unnamed: 0 & Unnamed: 0.1 as it contains serial number only.
- After dropping the columns, we don’t need we can also see that there are only 2 columns are of integer type i.e., PRICE & YEAR

VISUALIZATIONS:

```
In [12]: #The white spots shows that there are null values in these columns

         sns.heatmap(df.isnull())

Out[12]: <AxesSubplot:>
```



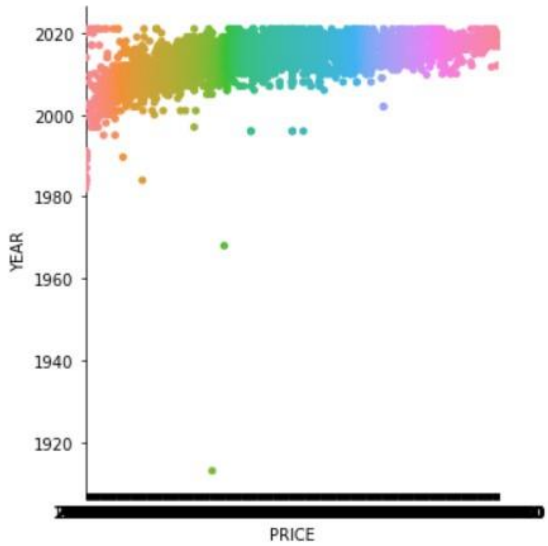
There are some null values present in our dataset showing it with the help of seaborn library & with the heatmap

All the white spots that you can see in the plot are null values shown here

```
In [41]: #Plotting catplot

         sns.catplot(x='PRICE',y='YEAR',data=df)

Out[41]: <seaborn.axisgrid.FacetGrid at 0x2081b3583a0>
```



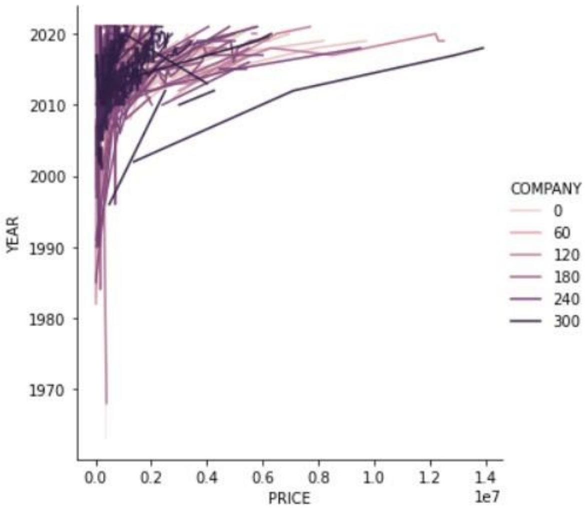
The categorical plot stats that PRICE & YEAR are in positive relationship we can here see as the year increases the Price of the cars are also increases

As a used car with low runner KM & latest as per MANUFACT-YEAR the PRICE is also increased for that

```
In [44]: #Plotting the relational plot of the PRICE & YEAR column with respect to the companies
         #The below shows us so many insights
         #Different companie model shows their unique impact we can see it in the below graph

         sns.relplot(x='PRICE',y='YEAR',hue='COMPANY',data=df,kind='line')

Out[44]: <seaborn.axisgrid.FacetGrid at 0x2081b7fa2e0>
```



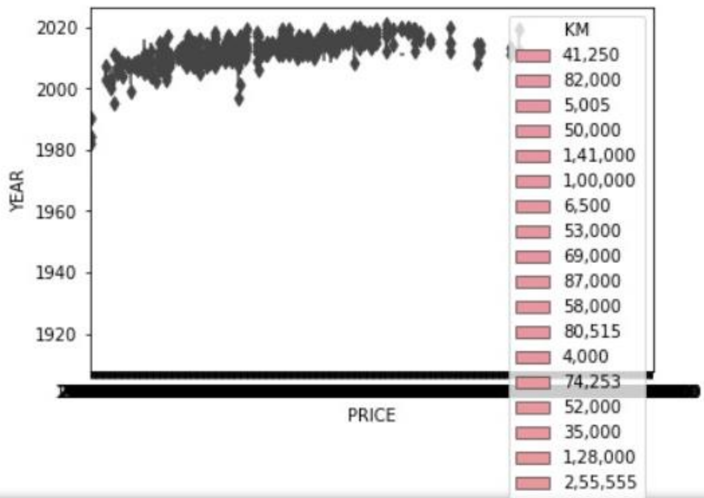
Here we have set hue=company means the plot will work with respect to company

In [45]:

```
#Plotting boxen plot of PRICE & YEAR with respect to KM
#The KMS that are used per car are showed here with different colours

sns.boxenplot(x='PRICE',y='YEAR',hue='KM',data=df)
```

Out[45]: `<AxesSubplot:xlabel='PRICE', ylabel='YEAR'>`



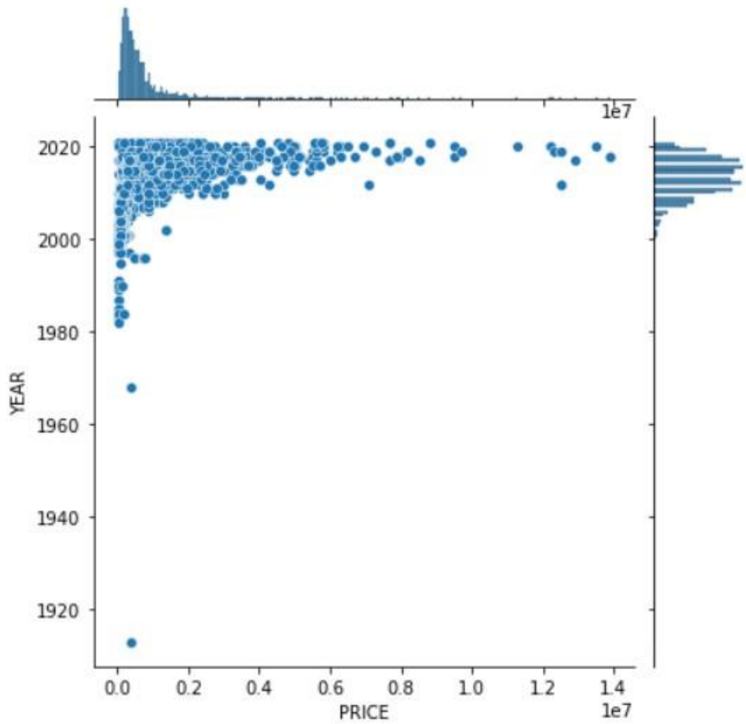
With respect to KMS also we are here plotting the boxenplot

Different lines are representing the KM runner by the cars

```
In [47]: #Plotting Jointplot

         sns.jointplot(x='PRICE',y='YEAR',data=df)

Out[47]: <seaborn.axisgrid.JointGrid at 0x20827e64a90>
```



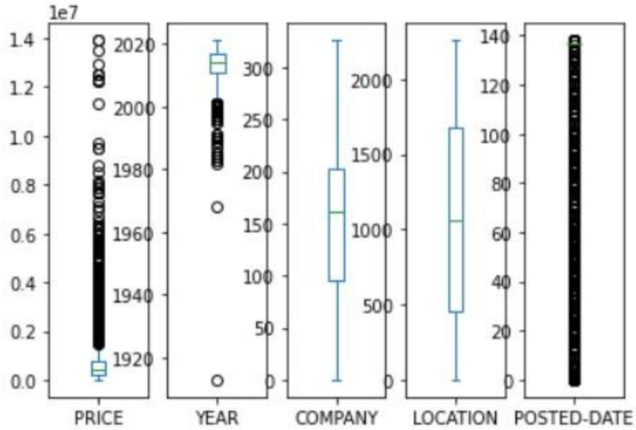
Plotting the joint plot of the variables

OUTLIERS:

```
In [50]: #Showing boxplots of all the variables

         df.plot(kind='box',subplots=True)

Out[50]: PRICE           AxesSubplot(0.125,0.125;0.133621x0.755)
         YEAR         AxesSubplot(0.285345,0.125;0.133621x0.755)
         COMPANY       AxesSubplot(0.44569,0.125;0.133621x0.755)
         LOCATION     AxesSubplot(0.606034,0.125;0.133621x0.755)
         POSTED-DATE  AxesSubplot(0.766379,0.125;0.133621x0.755)
         dtype: object
```



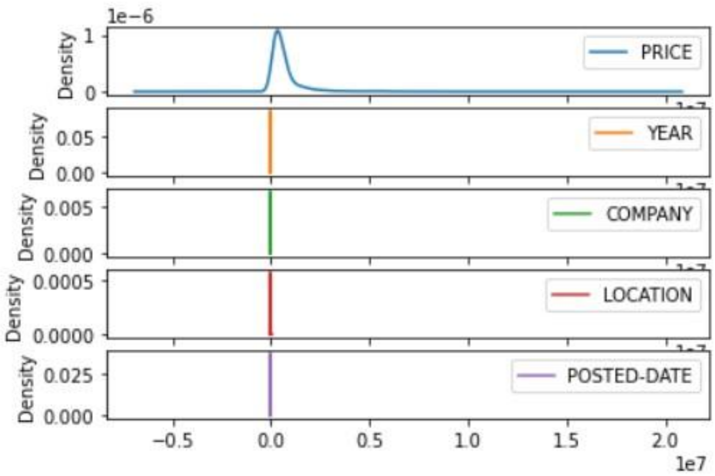
We can see outliers in many columns, we will check them & will remove them

LET'S CHECK THE SKEWNESS WITH THE HELP OF DISTRIBUTION PLOT

```
In [49]: #Determining the skewness present in all the columns
         #There is skewness present in our target variable

         df.plot(kind='kde',subplots=True)

Out[49]: array([<AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
                <AxesSubplot:ylabel='Density'>, <AxesSubplot:ylabel='Density'>,
                <AxesSubplot:ylabel='Density'>], dtype=object)
```



As we can clearly see there is some skewness present in PRICE

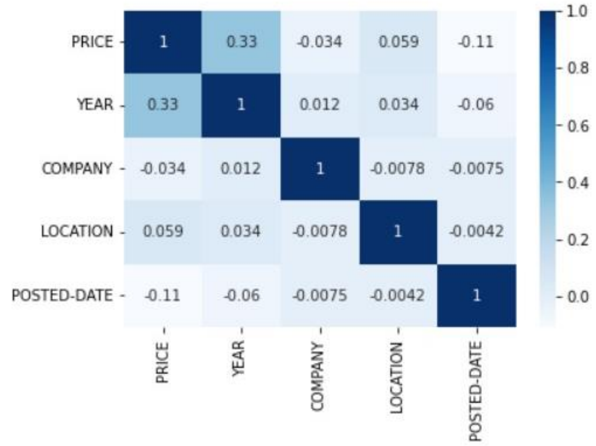
Also, in other columns as well

CORELATION: -

```
In [53]: #Showing corelation through heatmap & also printing values for better understanding
         #Here all the DARK COLUMNS are HIGHLY CORELATED

         sns.heatmap(dfcor,annot=True,cmap='Blues')

Out[53]: <AxesSubplot:>
```

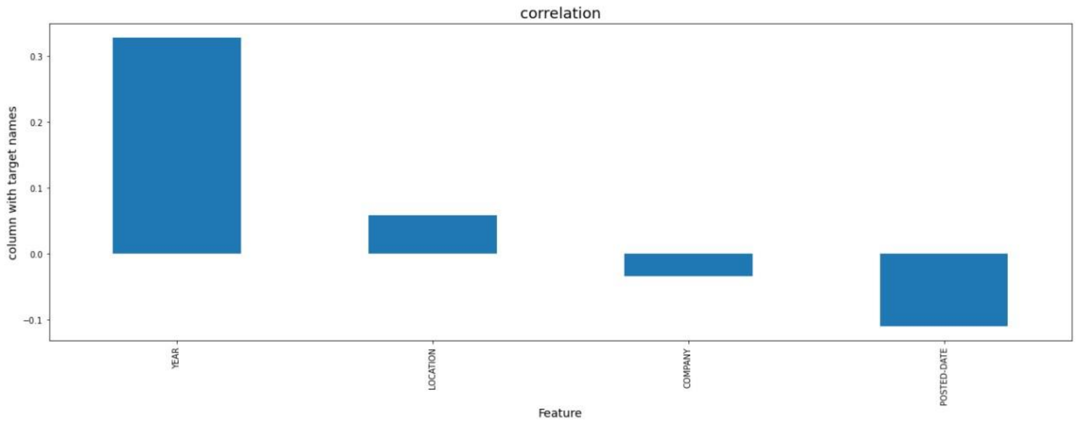


All the dark blue shades show in the heatmap

Are highly co-related to each other

```
In [54]: #Showing all the columns  positively & negatively corelated

plt.figure(figsize=(22,7))
df.corr()['PRICE'].sort_values(ascending=False).drop(['PRICE']).plot(kind='bar')
plt.xlabel('Feature',fontsize=14)
plt.ylabel('column with target names',fontsize=14)
plt.title('correlation',fontsize=18)
plt.show()
```



Only POSTED-DATE column is negatively corelated

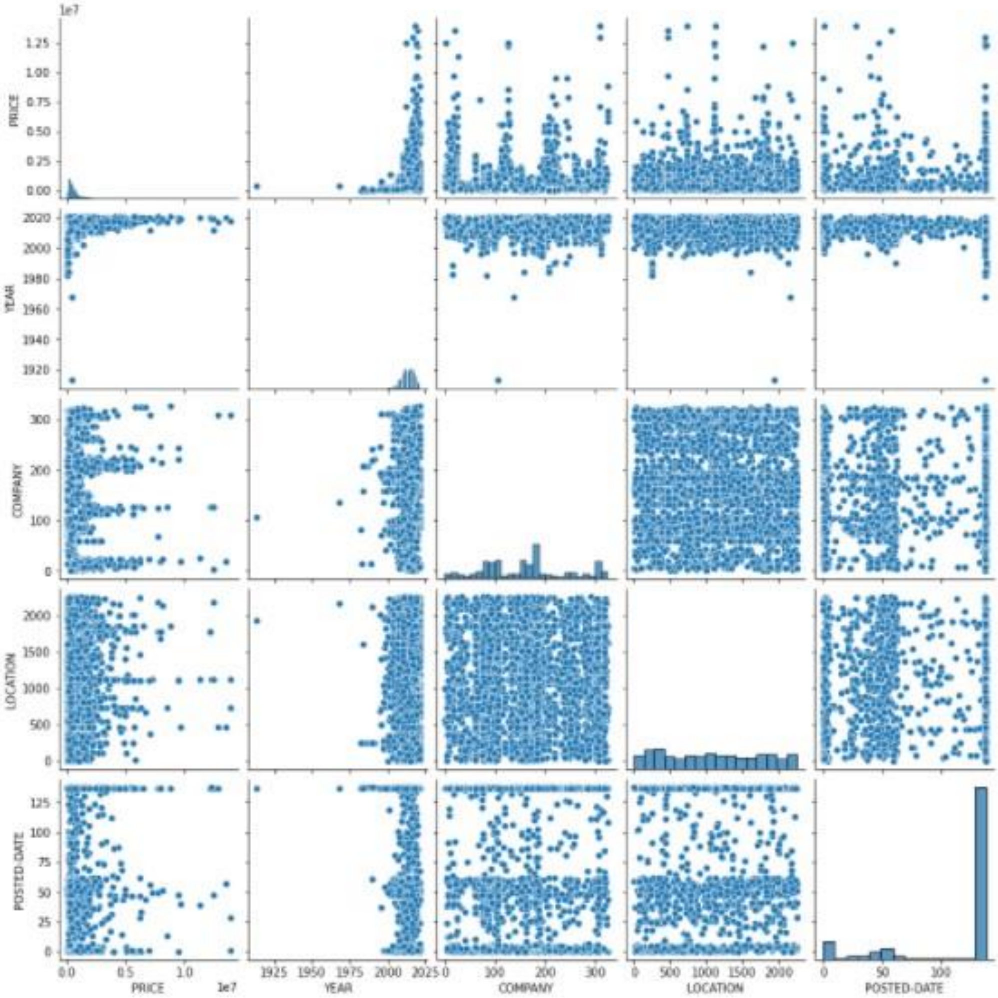
We can see clearly in the above graph

PAIRPLOT: -

```
In [48]: #Plotting all the variables in respect to all the variable to determine each relationships b/w the variables

         sns.pairplot(df)

Out[48]: <seaborn.axisgrid.PairGrid at 0x20829196bb0>
```



Plotting all the columns with respect to all the rows

OUTLIERS TREATEMENT

```
In [61]: #Removing outliers

         from scipy.stats import zscore
         z=np.abs(zscore(df))
         threshold=3
         np.where(z>3)
```

```
Out[61]: (array([   2,    3,   42,  148,  191,  220,  221,  266,  295,  303,  354,
                  365,  534,  553,  557,  580,  597,  630,  670,  693,  726,  766,
                  835,  849,  976,  977,  978,  979,  985,  987,  988,  989,  991,
                 1034, 1053, 1057, 1080, 1097, 1130, 1170, 1193, 1226, 1266, 1335,
                 1349, 1476, 1477, 1478, 1479, 1485, 1487, 1488, 1489, 1491, 1595,
                 1641, 1708, 1861, 1928, 2050, 2081, 2182, 2326, 2385, 2408, 2502,
                 2509, 2557, 2871, 2941, 2943, 2977, 3030, 3031, 3074, 3080, 3092,
                 3108, 3191, 3299, 3416, 3470, 3472, 3491, 3492, 3512, 3524, 3563,
                 3564, 3582, 3591, 3600, 3634, 3682, 3710, 3741, 3742, 3743, 3744,
                 3753, 3754, 3769, 3785, 3786, 3846, 3859, 3871, 3901, 3902, 3904,
                 3905, 3918, 3969, 3991, 4001, 4063, 4083, 4143, 4265, 4326, 4483,
                 4534, 4537, 4544, 4549, 4562, 4565, 4630, 4651, 4670, 4685, 4690,
                 4694, 4750, 4754, 4785, 4851, 4930, 4935, 4949, 4957], dtype=int64),
          array([0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
                 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64))
```

```
In [62]: #Storing the dataframe after removing outlier

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

Out[62]:

	PRICE	YEAR	COMPANY	LOCATION	POSTED-DATE
0	645000.0	2018	163	1034	48
1	725000.0	2017	184	360	3
4	185000.0	2008	183	1497	137
5	125000.0	2006	160	1700	137
6	145000.0	2015	283	1502	137
...	...	...	...	...	...
5492	1390000.0	2014	8	1882	137
5493	280000.0	2012	170	622	137
5494	45000.0	2006	161	1605	137
5495	325000.0	2017	104	1413	137
5496	700000.0	2018	107	2010	137

4828 rows × 5 columns

Removing outliers using z-score technique

Saving the cleaned data into the new data frame

As we can see the shape has also changed to 4282 rows

& 5 columns

SKEWNESS REMOVING: -

In [67]:
```
#Removnig skewness with the help of power transformation

from sklearn.preprocessing import power_transform
df_new=power_transform(x)

df_new=pd.DataFrame(df_new,columns=x.columns)
```

In [68]:
```
#Here we can see skewness has been removed from the data

df_new.skew()
```

Out[68]:
```
YEAR          -0.210969
COMPANY       -0.103469
LOCATION      -0.250172
POSTED-DATE   -1.431037
dtype: float64
```

We have removed the skewness to the maximum extent we can using power transformation technique

FINDING BEST RANDOM STATE:

```
In [70]: #Using for loop determining the best accuracy for the model at the best random state

maxAccu=0
maxRS=0
for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.10,random_state=i)
    DTC=DecisionTreeClassifier()
    DTC.fit(x_train,y_train)
    pred=DTC.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is ",maxAccu, " on Random State ",maxRS)
```

```
Best accuracy is  0.2836438923395445  on Random State  8
```

With the help of for loop we are determining the best possible accuracy at the best possible random state

MODEL/S DEVELOPMENT AND EVALUATION:

```
In [71]: #Sending the data for training & testing phase

         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.10,random_state=maxRS)
```

```
In [72]: #Created a list in which we have stored all the instances of the model
         #Using for loop we will determine the accuracy of all the models

         model=[DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier(),LogisticRegression()]

         for m in model:
             m.fit(x_train,y_train)
             #m.score(x_train,y_train)
             pred=m.predict(x_test)
             acc=accuracy_score(y_test,pred)
             print('Accuracy Score of',m,'is:', acc)
             print(confusion_matrix(y_test,pred))
             print(classification_report(y_test,pred))
             print('\n')
```

Sending the data to training & testing phase with the help of train test split

Again, with the help of for loop we are here determining the accuracies of all the models for selecting the best model for our dataset

MODEL SELECTION

MODELS	ACCCURACY
DTC	28.57
SVC	00.62
ABC	00.14
RFC	27.32
LR	00.08

Here in the table, we can see that DTC model is giving

Highest accuracy among all the different present models

So, we select the DTC best for our model

But before proceeding further we will cross validate

Each of the models so that we can finally select which model will be the best for our dataset

CROSS VALIDATION: -

```
In [90]: #Again using for loop we will now determine the cross validation of each model

         model=[DecisionTreeClassifier(),SVC(),AdaBoostClassifier(),RandomForestClassifier(),LogisticRegression()]
         for m in model:
             score=cross_val_score(m,x,y,cv=5)
             print("Score for ",m," is : ",score.mean())

         Score for  DecisionTreeClassifier()  is :  0.21226702710820752
         Score for  SVC()  is :  0.012220255527306666
         Score for  AdaBoostClassifier()  is :  0.017190916014975487
         Score for  RandomForestClassifier()  is :  0.21267981849193834
         Score for  LogisticRegression()  is :  0.007869640309378988
```

After determining the cross-validation scores of the models we will now select RFC as our model

Because the RFC model shows the least difference in all the models

HYPER PARAMETER TUNNING: -

```
In [106]: #Importing the grid search cv for hypertune the model

          from sklearn.model_selection import GridSearchCV

In [107]: #Passing the parameters for the model

          parameters = {'max_depth':np.arange(2,10),
                        'criterion':['gini','entropy'],
                        'max_features':['auto','sqrt','log2'],
                        'min_samples_split':[2,3,4]}

In [108]: #Creating the instance of the model

          GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=5)

In [109]: #Fetting the model

          GCV.fit(x_train,y_train)

Out[109]: GridSearchCV(cv=5, estimator=RandomForestClassifier(),
                       param_grid={'criterion': ['gini', 'entropy'],
                                   'max_depth': array([2, 3, 4, 5, 6, 7, 8, 9]),
                                   'max_features': ['auto', 'sqrt', 'log2'],
                                   'min_samples_split': [2, 3, 4]})

In [110]: #Getting the best parameters

          GCV.best_params_

Out[110]: {'criterion': 'entropy',
           'max_depth': 9,
           'max_features': 'sqrt',
           'min_samples_split': 3}

In [111]: #Passing the best parameters & printing the final accuracy

          Final_mod= RandomForestClassifier(criterion="entropy",max_depth=9,max_features="sqrt",min_samples_split=3)
          Final_mod.fit(x_train,y_train)
          pred=Final_mod.predict(x_test)
          acc=accuracy_score(y_test,pred)
          print(acc*100)

          23.60248447204969
```

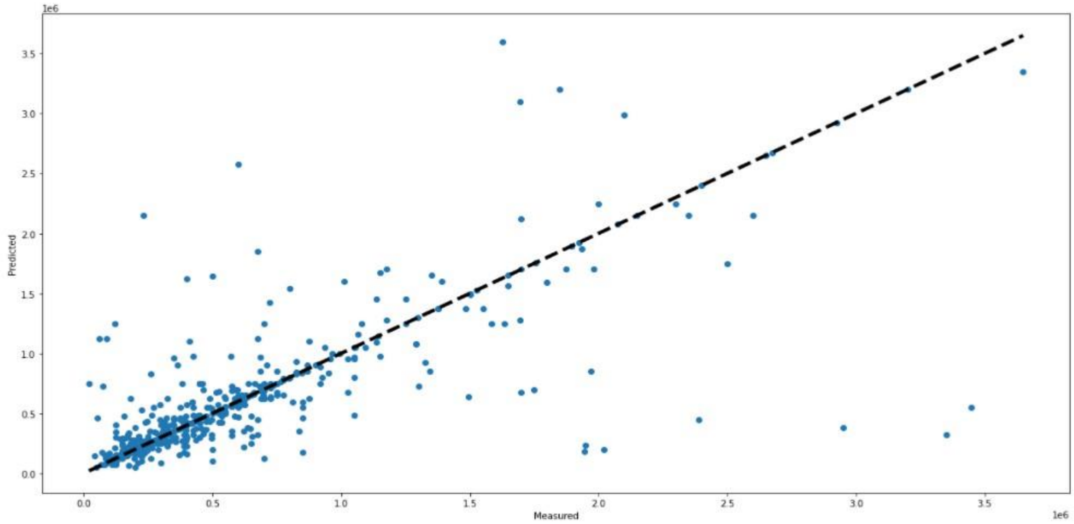
Tunning our model to get the best accuracy of the model

PLOTTING A GRAPH STATING THE PREDICTED & MEASURED RESULTS

In [113]:

```
#Ploting a diagram of the predicted & measured results

fig_dims = (20, 10)
fig, ax = plt.subplots(figsize=fig_dims)
ax.scatter(y_test, pred)
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.show()
```



The line that tries to cover the Blues Points that is our predicted Line Which tries to get mix with the Measured results

SAVING THE MODEL: -

```
In [114]: #Importing pickle for saving the model
          #Saving it in the pickle file

          import pickle
          filename= 'CAR-PREDICTION.pkl'
          pickle.dump(Final_mod, open(filename, 'wb'))
```

Saving the model with the help of pickle library

CONCLUSION: -

In [115]:
```
#load the model from the disk

loaded_model = pickle.load(open('CAR-PREDICTION.pkl', 'rb'))
result = loaded_model.score(x_test,y_test)
print(result)
```

```
0.2360248447204969
```

In [116]:
```
#Printing the dataframe of the predicted & measured

conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred[:]],index=["Predicted","Orginal"])
conclusion
```

Out[116]:

	0	1	2	3	4	5	6	7	8	9	...	473	474	475	476	
Predicted	725000.0	625000.0	265000.0	575000.0	200000.0	3600000.0	510000.0	750000.0	3100000.0	835000.0	...	150000.0	399999.0	485000.0	640000.0	67
Orginal	725000.0	625000.0	265000.0	575000.0	200000.0	3600000.0	510000.0	750000.0	3100000.0	835000.0	...	150000.0	399999.0	485000.0	640000.0	67

2 rows × 483 columns

In [117]:
```
#END
```

Also, making a data frame of predicted answers & measured answers & printing it