STOCK PRICE PREDICTION OF BANKS USING MACHINE LEARNING

A STUDY ON

STOCK PRICE PREDICTION OF BANKS USING MACHINE LEARNING

A PROJECT REPORT SUBMITTED TO,

GITAM UNIVERSITY

Visakhapatnam

Submitted in partial fulfillment of the requirements

For the award of the degree of

"MASTER OF BUSINESS ADMISTRATION"

Submitted by:

BALAJI

Regd No. : 121923901002

AMMAN NAIDU

Regd No. :121923901004

Under the esteemed guidance of

Sri. Leben Joshnon

(Associate Professor)

GITAM INSTITUTE OF MANAGEMENT

GITAM UNIVERSITY

(U/s 3 of UGC Act)

VISAKHAPATANM

(2019-2020)

DECLARATION

I, the undersigned, hereby declare that, the project titled **STOCK PRICE PREDICTION OF BANKS USING MACHINE LEARNING**" submitted to GITAM Institute of Management, GITAM University for the award of the Masters of Business Administration (FinTech) is the original work done by us, under the guidance of Mr. Leben Johnson, Associate Professor, GITAM Institute of Management The empirical findings in this report are based on the data collected by us. It was not copied from any other project. The project has not been submitted to any Substitute/University for the award of any Diploma/Degree.

Date: -

Place: - Visakhapatnam

BALAJI

(121823901002)

AMMAN NAIDU

(121823901004)

CERTIFICATE

This is to certify that the project entitled, "**STOCK PRICE PREDICTION OF BANKS USING MACHINE LEARNING",** is a bonafide work done by **AMMAN NAIDU (121823901004),BALAJI (121923901002**) and is submitted in partial fulfillm**ent for the Masters of Business Administration (Fintech**) of GITAM University. It has not been **submitted for the award of any** diploma/degree in any other Institution/University.

Date:	Mr. Leben Johnson
Place: Visakhapatnam	Associate Professor

ACKNOWLEDGEMENT

I am thankful to Prof. Y. Gowtham Rao, Director, GITAM Institute of Management, GITAMUniversity, Vishakhapatnam, for giving me an opportunity to learn and have a study on very important topic of management. I am thankful to Mr. Leben Johnson, Program Coordinator for his guidance.

K. AMMAN NAIDU (121823901004)

A.BALAJI (121923901002)

Table of content

Executive Summary

The project titled "STOCK PRICE PREDICTION OF BANKS USING MACHINE LEARNING" the basic problem of investing on stock market is most of the investors are not clear about the fundamental and techniques of investments, they mainly concentrate on risk and return involved in investing and make decisions without the guidance of experts which are thus based on judgments made on their individual assessments.

Therefore, the objective here is to understand the basic techniques involved in investing in equity market. Learning the above said objective will give a clear idea of the concept involved in investment decisions.

Technical analysis is the study of financial market action. The technical analysts look at the price movement that occur on day to day or week to week or over other constant time period displayed in graphic form, called charts. Technical analysis is a process of analysis is a process of analyzing a security's historical prices in an effort to determine probable future prices.

From the study it has been understand that the price movement of the shares can be determined by using technical analysis and accurate buy and sell decision can be made so that the investors can minimize their risk and maximize the profits also show that all technical analysis indicators can give accurate decisions. The Study show that the historical prices have an impact on future prices, however, even if you are unable to accurately forecast prices, technical analysis can be used to consistently reduce your risks and improve your profits.

Machine Learning Algorithm

Multiple Linear Regression

Multiple Linear Regression is a supervised learning algorithm for finding the existence of an association relationship between a dependent variable and several independent variables

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.

Explaining Multiple Linear Regression

A simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables.
- The independent variables are not too highly correlated with each other.
- yi observations are selected independently and randomly from the population.
- Residuals should be normally distributed with a mean of 0 and variance σ.

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R2 always increases as more predictors are added to the MLR model even though the predictors may not be related to the outcome variable.

R2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

Macro Economic Factors

GDP

Unemployment Rate

Inflation Rate

Consumer Confidence

Consumer Price Index

Gold price

Yield Value Of 10 Years bond

COLLECTING DATA:

We used eikon and quandl Sources for collecting the macro details and four banks stock prices of 20 years

We used API key and RIC’s to gets the data from eikon by the below codes and saved the data in the csv files.

```
import eikon as ek

import pandas as pd

ek.set_app_key('53f3b6e6a2af4ee987df42330c0fcbf40f75a204')
```

As we are unable to get the 20 years of data we merged the data by calling it in separate parts

```
bank = ek.get_timeseries(["JPM"], start_date = "1998-12-01", end_date ="2007-12-21",
interval="daily",corax="adjusted") # JP Morgan

bank_1 = ek.get_timeseries(["JPM"], start_date = "2007-12-22", end_date ="2019-11-12",
interval="daily",corax="adjusted")

bank_jpm_1=pd.merge(bank,bank_1,on=['Date','HIGH', 'CLOSE', 'LOW', 'OPEN', 'COUNT', 'VOLUME'],how="outer")

bank_jpm_1.to_csv("JPM.csv")
```

Unemployment Rate :

```
US_UNEMPLOY = ek.get_timeseries(["USUNR=ECI"], start_date = "1998-12-01", end_date = "2019-11-
12",interval="monthly")

US_UNEMPLOY.to_csv("unemployment.csv")
```

Inflation Rate:

```
US_INFLATION = ek.get_timeseries(["aUSWOCPIPR"], start_date = "1998-12-01", end_date = "2019-11-
12",interval="yearly")

US_INFLATION.to_csv("inflation.csv")
```

Consumer Confidence Rate:

```
US_CC = ek.get_timeseries(["USCONC=ECI"], start_date = "1998-12-01", end_date = "2019-11-
12",interval="monthly")

US_CC.to_csv("consumer_confidence.csv")
```

USA GDP:

```
import quandl

USA_GDP = quandl.get("FRED/GDP")

USA_GDP.to_csv("gdp.csv")
```

CPI:

```
US_CPI = ek.get_timeseries(["USCPSA=ECI"], start_date = "1998-12-01", end_date = "2019-11-12",interval="monthly")

US_CPI.to_csv("cpi.csv")
```

10Y Bond yield%:

```
Yield = ek.get_timeseries(["aUSEBM10Y"], start_date = "1998-12-01", end_date = "2019-11-12",interval="monthly")

Yield.to_csv("yield.csv")
```

Gold price world wise:

```
GOLD1 = ek.get_timeseries(["XAU="], start_date = "1998-12-01", end_date = "2007-12-21",interval="daily")

GOLD2 = ek.get_timeseries(["XAU="], start_date = "2007-12-22", end_date = "2019-11-12",interval="daily")

GOLD=pd.merge(GOLD1,GOLD2,on=['Date','HIGH', 'CLOSE', 'LOW', 'OPEN'],how="outer")

GOLD.to_csv("gold.csv")
```

CLEANING DATA:

Our data has different granularity so we while merging the data we found lot of NaN values by using interpolate we are able to remove the NaN values and arranged them in equal granularity.

```
import pandas

def bank(filename):

   bank=pandas.read_csv(filename,parse_dates=["Date"],index_col="Date")

   bank.drop(columns=['HIGH','LOW', 'OPEN', 'COUNT','VOLUME'],inplace=True)

   dt=pandas.date_range(start="1998-12-01",end="2019-11-11",freq="D")

   idx=pandas.DatetimeIndex(dt)

   bank=bank.reindex(idx)

   bank.index.name='DATE'

   return(bank)

BOA_BANK=bank("BOA1.csv")

BOA_BANK.rename(columns={"CLOSE":"BOA_CLOSE"},inplace=True)
```

USA GDP:

```
usa_gdp=pandas.read_csv("gdp12.csv",parse_dates=["Date"],index_col="Date")

usa_gdp.rename(columns={"Value":"GDP_VALUE"},inplace=True)

dt=pandas.date_range(start="1998-12-01",end="2019-11-11",freq="D")

idx=pandas.DatetimeIndex(dt)

usa_gdp=usa_gdp.reindex(idx)

usa_gdp.interpolate(method="time",inplace=True)
```

Usa Unemployement:

```
usa_unemp=pandas.read_csv("unemployment.csv",parse_dates=["Date"],index_col="Date")

usa_unemp.rename(columns={"VALUE":"UNEMPLO_RATE"},inplace=True)
```

Usa Inflation:

```
usa_inf=pandas.read_csv("inflation.csv",parse_dates=["Date"],index_col="Date")

usa_inf.rename(columns={"VALUE":"INFLA_RATE"},inplace=True)
```

Usa Consumer Confidence:

```
usa_CC=pandas.read_csv("consumer_confidence.csv",parse_dates=["Date"],index_col="Date")

usa_CC.rename(columns={"VALUE":"CC_VALUE"},inplace=True)
```

USA CPI:

```
usa_CPI=pandas.read_csv("CPI.csv",parse_dates=["Date"],index_col="Date")

usa_CPI.rename(columns={"VALUE":"CPI_VALUE"},inplace=True)
```

GOLD:

```
GOLD=pandas.read_csv("gold.csv",parse_dates=["Date"],index_col="Date")

GOLD.drop(columns=['HIGH','LOW', 'OPEN'],inplace=True)

GOLD.rename(columns={"CLOSE":"GOLD_CLOSE"},inplace=True)
```

YIELD:

```
YIELD=pandas.read_csv("yield.csv",parse_dates=["Date"],index_col="Date")

YIELD.rename(columns={"VALUE":"YIELD_VALUE"},inplace=True)
```

Merging Data:

```
data = BOA_BANK.merge(CITY_BANK, how='outer', left_index=True, right_index=True)

data=data.merge(WELLS_BANK, how='outer', left_index=True, right_index=True)

data=data.merge(JPM_BANK, how='outer', left_index=True, right_index=True)
```

```
data=data.merge(usa_gdp, how='outer', left_index=True, right_index=True)

data=data.merge(usa_unemp, how='outer', left_index=True, right_index=True)

data=data.merge(usa_inf, how='outer', left_index=True, right_index=True)

data=data.merge(usa_CC, how='outer', left_index=True, right_index=True)

data=data.merge(usa_CPI, how='outer', left_index=True, right_index=True)

data=data.merge(GOLD, how='outer', left_index=True, right_index=True)

data=data.merge(YIELD, how='outer', left_index=True, right_index=True)
```

For Filling The Nan Values:

```
data.interpolate(method="time",inplace=True)
```

Slicing Data:

```
data=data['19990101':'20191111']

data.index.name='DATE'
```

Checking For NaN values:

```
data.info()
```

OUTPUT:

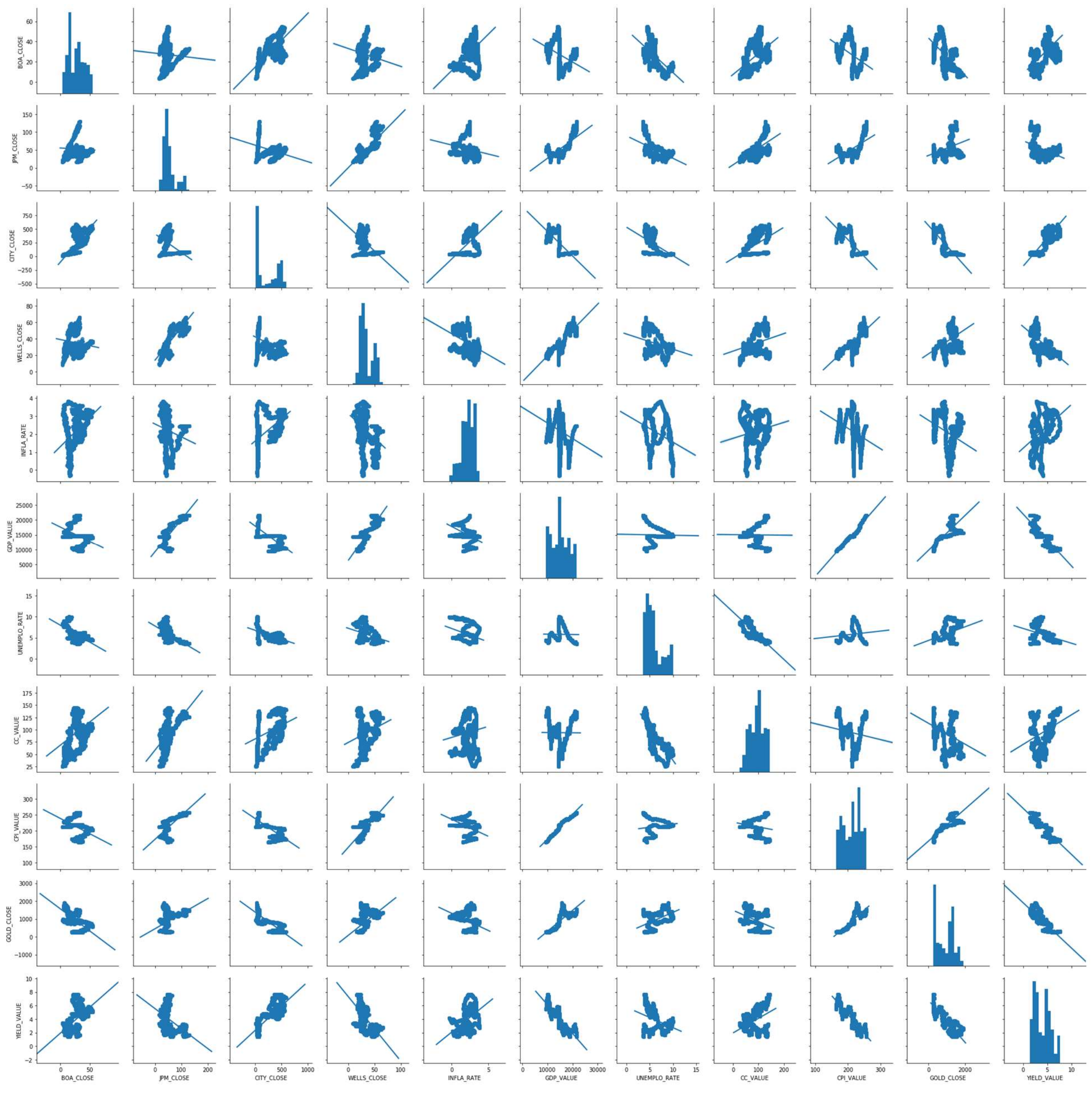
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 7620 entries, 1999-01-01 to 2019-11-11
Data columns (total 11 columns):
BOA_CLOSE      7620 non-null float64
JPM_CLOSE      7620 non-null float64
CITY_CLOSE     7620 non-null float64
WELLS_CLOSE    7620 non-null float64
INFLA_RATE     7620 non-null float64
GDP_VALUE      7620 non-null float64
UNEMPLO_RATE   7620 non-null float64
CC_VALUE       7620 non-null float64
CPI_VALUE      7620 non-null float64
GOLD_CLOSE     7620 non-null float64
YIELD_VALUE    7620 non-null float64
dtypes: float64(11)
memory usage: 1.0 MB
```

VISUALIZING THE DATA:

We used pair plot, histograms and scatter plots to visualize the data to know the insights of data.

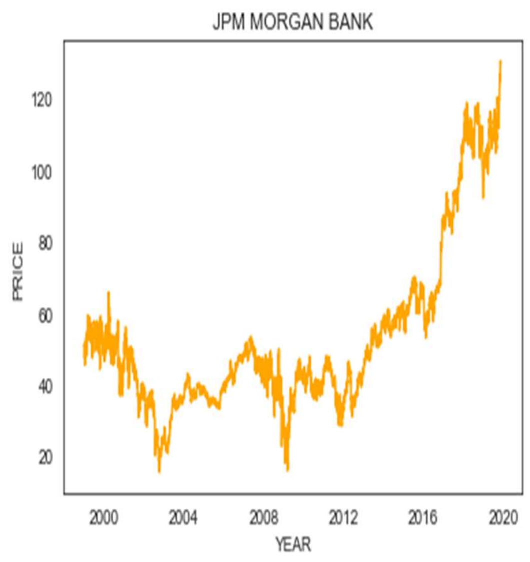
Below are the codes and outputs:

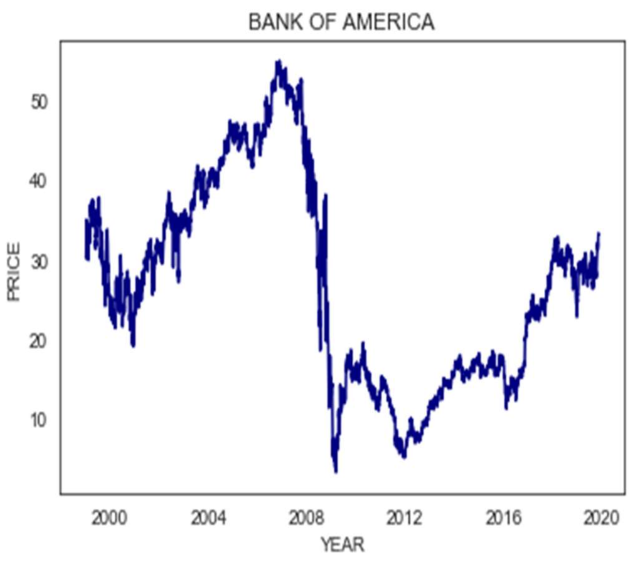
```
sns.pairplot(data,kind='reg')

plt.show()
```

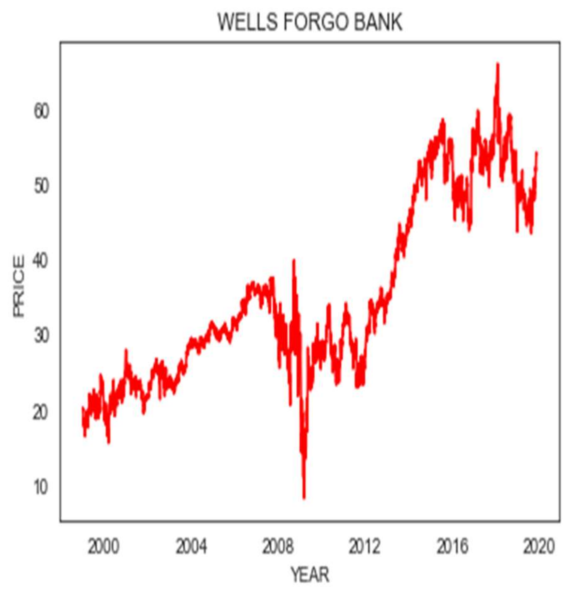


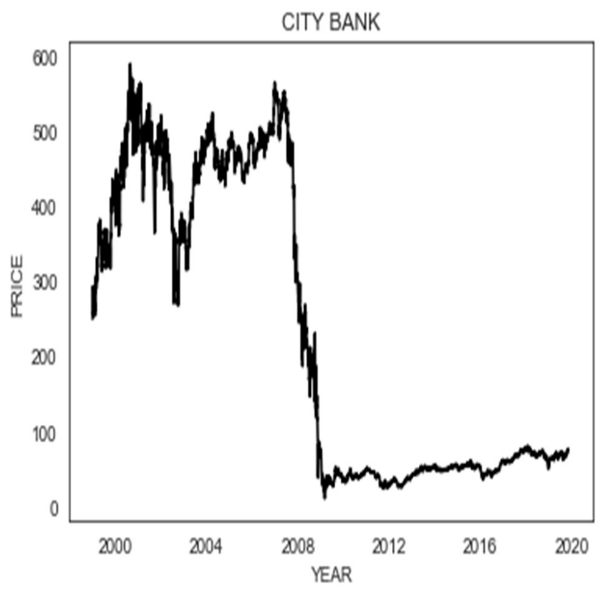
SCATTER PLOTS: Below is the code for

```
plt.plot(data["JPM_CLOSE"],color='orange')

plt.xlabel("YEAR")

plt.ylabel("PRICE")

plt.title("JPM MORGAN BANK")
```



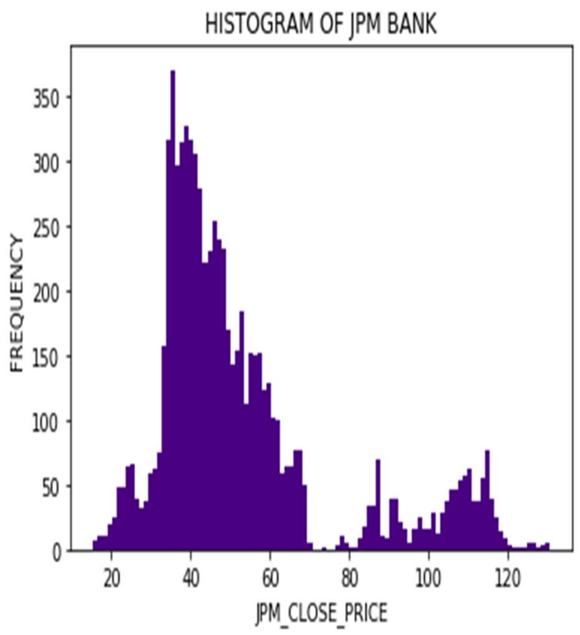


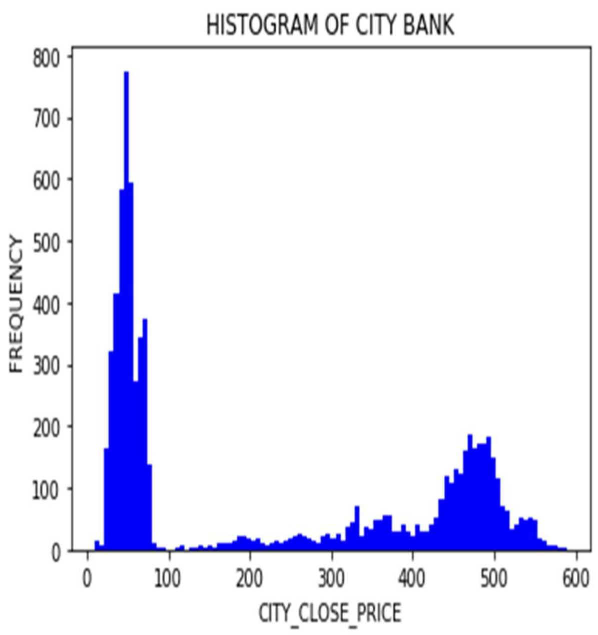


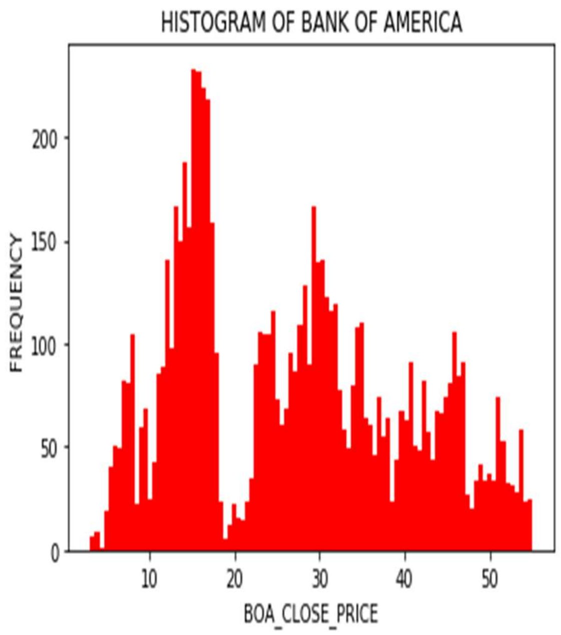


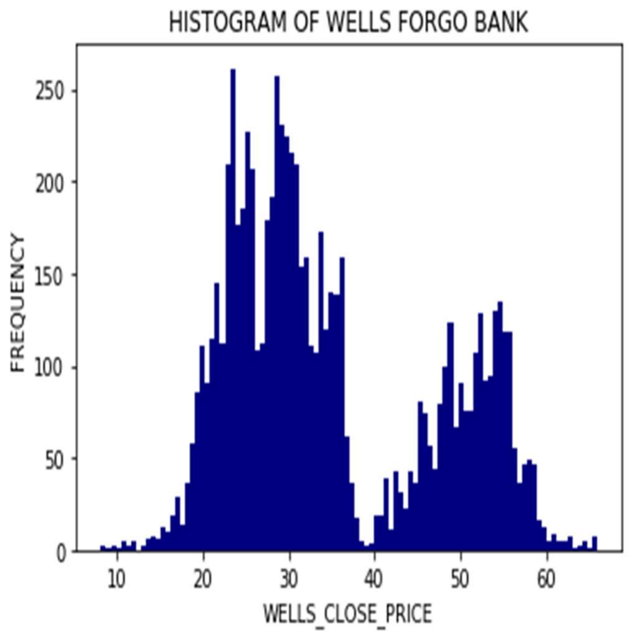
HISTOGRAM PLOTS: Below is the code for

```
plt.hist(data["JPM_CLOSE"],bins=100,color='indigo')

plt.xlabel("JPM_CLOSE_PRICE")

plt.ylabel("FREQUENCY")

plt.title("HISTOGRAM OF JPM BANK")
```



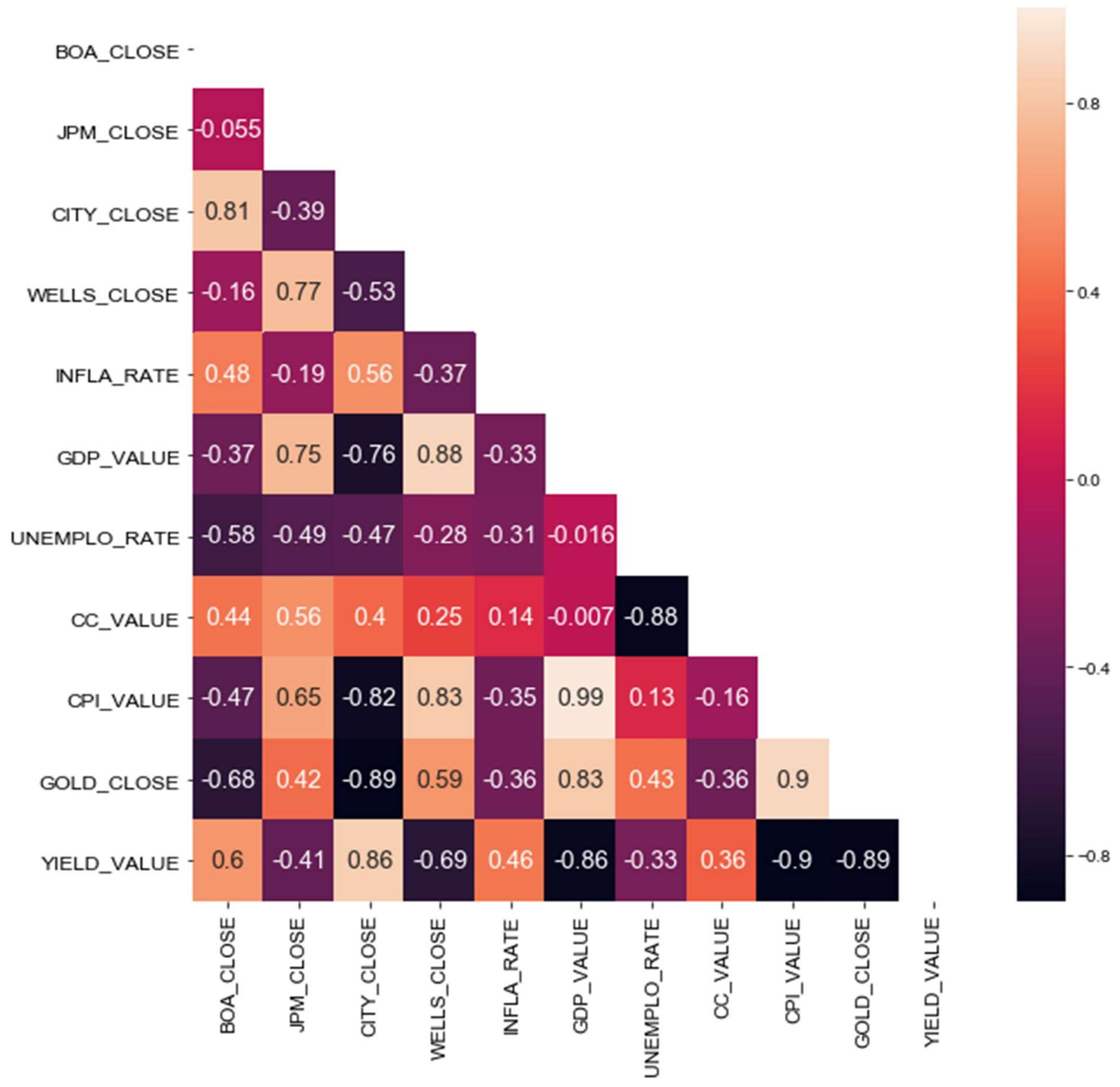






CORRELATION PLOT: Below is the code for correlation

```
mask=np.zeros_like(data.corr())

traingle=np.triu_indices_from(mask)

mask[traingle]=True

plt.figure(figsize=(10,10))

sns.heatmap(data.corr(),mask=mask,annot=True,annot_kws={"size":14})

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

sns.set_style("white")
```



SPLITTING DATA:

Below is the code for splitting data.in the below code x is the target value y is the features.used sklearn module for splitting the data into training and testing data in the ratio 80:20.

```
from sklearn.model_selection import train_test_split

x2=data[['JPM_CLOSE']]

y2=data.drop(['BOA_CLOSE', 'JPM_CLOSE', 'CITY_CLOSE','WELLS_CLOSE'],axis=1)

x_train2,x_test2,y_train2,y_test2=train_test_split(y2,x2,test_size=0.2,random_state=0)
```

BUILDING MODEL:

Used multi linear regression in order to predict the model we considered JP Morgan bank Close price as target variable and features as all the macro factors

MULTI LINEAR REGRESSION MODEL CREATION OF JPM BANK:

```
regression2=LinearRegression()

regression2.fit(x_train2,y_train2)

print(pandas.DataFrame({"index":x_train2.columns,"coeff":regression2.coef_.tolist()[0]}))

pandas.DataFrame(data=regression2.coef_,index=['coefficient'],columns=x_train2.columns)

print("intercept:",regression2.intercept_)

print('R^2 train dataset:',regression2.score(x_train2,y_train2))

print('R^2 test dataset:',regression2.score(x_test2,y_test2))
```

OUTPUT:

```
        index           coeff

0    INFLA_RATE       - 0.941970

1     GDP_VALUE       0.013303

2  UNEMPLO_RATE   1.701889

3      CC_VALUE        0.381072

4     CPI_VALUE       -0.871155

5    GOLD_CLOSE       0.004967

6   YIELD_VALUE       4.657317

intercept: [-28.65914051]

R^2 train dataset: 0.9002138646376936

R^2 test dataset: 0.9067647947946865
```

EVALUATING THE MODEL: In this we done some statistical test in order to find out the significance of each independent variable to the target variable so we performed p statistics by using OLS (ordinary Least Square) method of statsmodels.

Code:

```
import statsmodels.api as sm

x2_include_constant=sm.add_constant(x_train2)

model2=sm.OLS(y_train2,x2_include_constant)

results2=model2.fit()

results2.params

results2.pvalues

print(pandas.DataFrame({"Coeffiencients":results2.params,"P-Values":round(results2.pvalues,2)}))
```

output:

```
              Coeffiencients   P-Values

const              -28.659141     0.0

INFLA_RATE          -0.941970     0.0

GDP_VALUE            0.013303     0.0

UNEMPLO_RATE         1.701889     0.0

CC_VALUE             0.381072     0.0

CPI_VALUE           -0.871155     0.0

GOLD_CLOSE           0.004967     0.0

YIELD_VALUE          4.657317     0.0
```

By above results we found all the variables are significant to the target value.

CHECKING FOR MULTI COLLINEARITY:

Variance inflation Factor (VIF) is used to check the multi co-linearity this can be done by importing variance_inflation_factor from statsmodels in python.

CODE:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
VIF=[]
for i in range(0, len(x_include_constant.columns)):
    VIF.append(variance_inflation_factor(exog=x_include_constant.values,exog_idx=i))
print(VIF)
print(pandas.DataFrame({"coeffients":x_include_constant.columns,"VIF":np.around(VIF,3)}))
```

output:

```
       coeffients       VIF
0           const    6185.503
1      INFLA_RATE       1.968
2       GDP_VALUE     295.629
3    UNEMPLO_RATE      10.599
4        CC_VALUE      12.833
5       CPI_VALUE     350.044
6      GOLD_CLOSE      22.206
7     YIELD_VALUE      11.811
```

For the above output table we observed the VIF values are high for some variable so we will try to drop those variables in the table. So we will drop CPI_VALUE and then run the model below are the results.

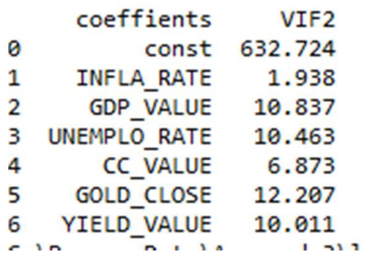
```
      index          coeff
0   INFLA_RATE     -0.698328
1    GDP_VALUE      0.007073
2  UNEMPLO_RATE   1.957227
3     CC_VALUE      0.490040
4   GOLD_CLOSE     -0.003325
5   YIELD_VALUE     3.654057
```

intercept: [-121.04221419]

R^2 train dataset: 0.8973329178860255

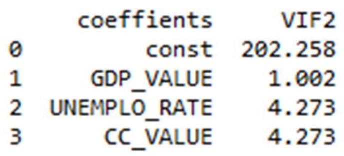
R^2 test dataset: 0.9034611975914223

VIF RESULTS AFTER DROPING CPI_VALUE:



```
     coeffients     VIF2
0         const  632.724
1    INFLA_RATE    1.938
2     GDP_VALUE   10.837
3   UNEMPLO_RATE  10.463
4      CC_VALUE    6.873
5    GOLD_CLOSE   12.207
6   YIELD_VALUE   10.011
```

Still we have multi co-linearity as the GOLD _CLOSE is high so we will drop that, and similarly we checked for the rest once also and find and below are the results.



```
     coeffients     VIF2
0         const  202.258
1     GDP_VALUE    1.002
2   UNEMPLO_RATE   4.273
3      CC_VALUE    4.273
```

	index	coeff
0	**GDP_VALUE**	0.005197
1	**UNEMPLO_RATE**	1.319710
2	**CC_VALUE**	0.548225

intercept:	[-84.52143208]
R^2 train dataset:	0.8871161790732169
R^2 test dataset:	0.8955482922548127

So we will continue the model with the above parameters as we considered the VIF limit is less than 5

RESIDUALS: These are the difference between predicted value and target value.

Below code will be used to find the residuals

CODE:

```
results2.resid
```

Co-relation Between y_Train (actual target value) And Predicted y_Train(predicted value)

CODE:

```
act_pred2=pandas.DataFrame({"actual":y_train2["JPM_CLOSE"],"predicted":results2.fittedvalues})

corr2=round(act_pred2["actual"].corr(act_pred2["predicted"]),2)

print(corr2)
```

output:

```
0.94
```

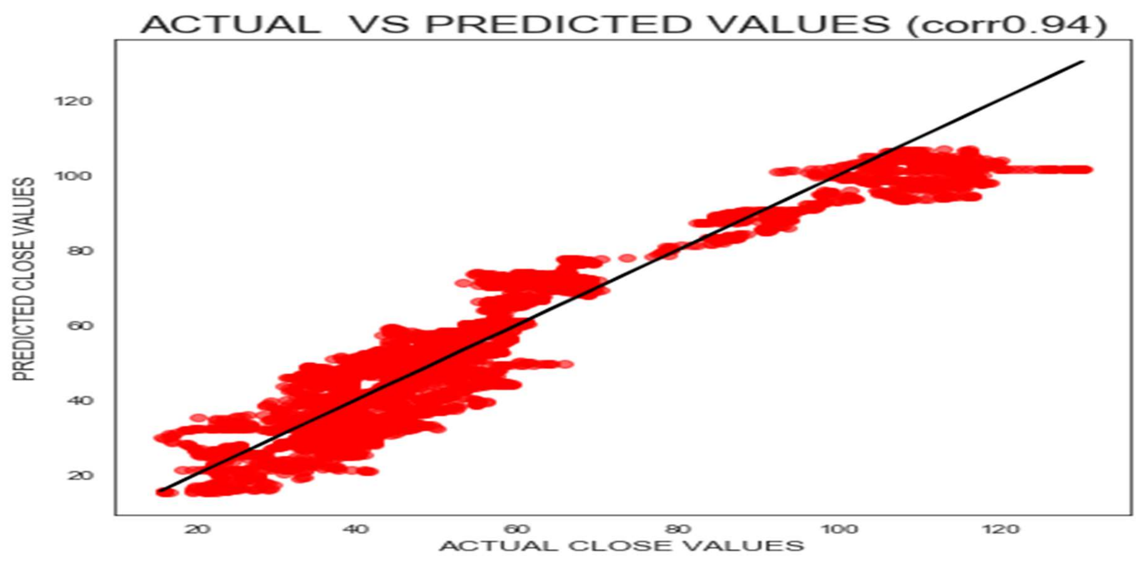
So we can say by above result the relation holds well.

GRAPH OF ACTUAL VS PREDICTION:

SCATTER PLOT

```
plt.figure(figsize=(7,7))

plt.scatter(x=act_pred2["actual"],y=act_pred2["predicted"],color="red",alpha=0.6)

plt.plot(act_pred2["actual"],act_pred2["actual"],color="black")

plt.xlabel("ACTUAL CLOSE VALUES",fontsize=12)

plt.ylabel("PREDICTED CLOSE VALUES",fontsize=12)

plt.title(f'ACTUAL  VS PREDICTED VALUES (corr{corr})',fontsize=18 )

plt.show()
```

output:

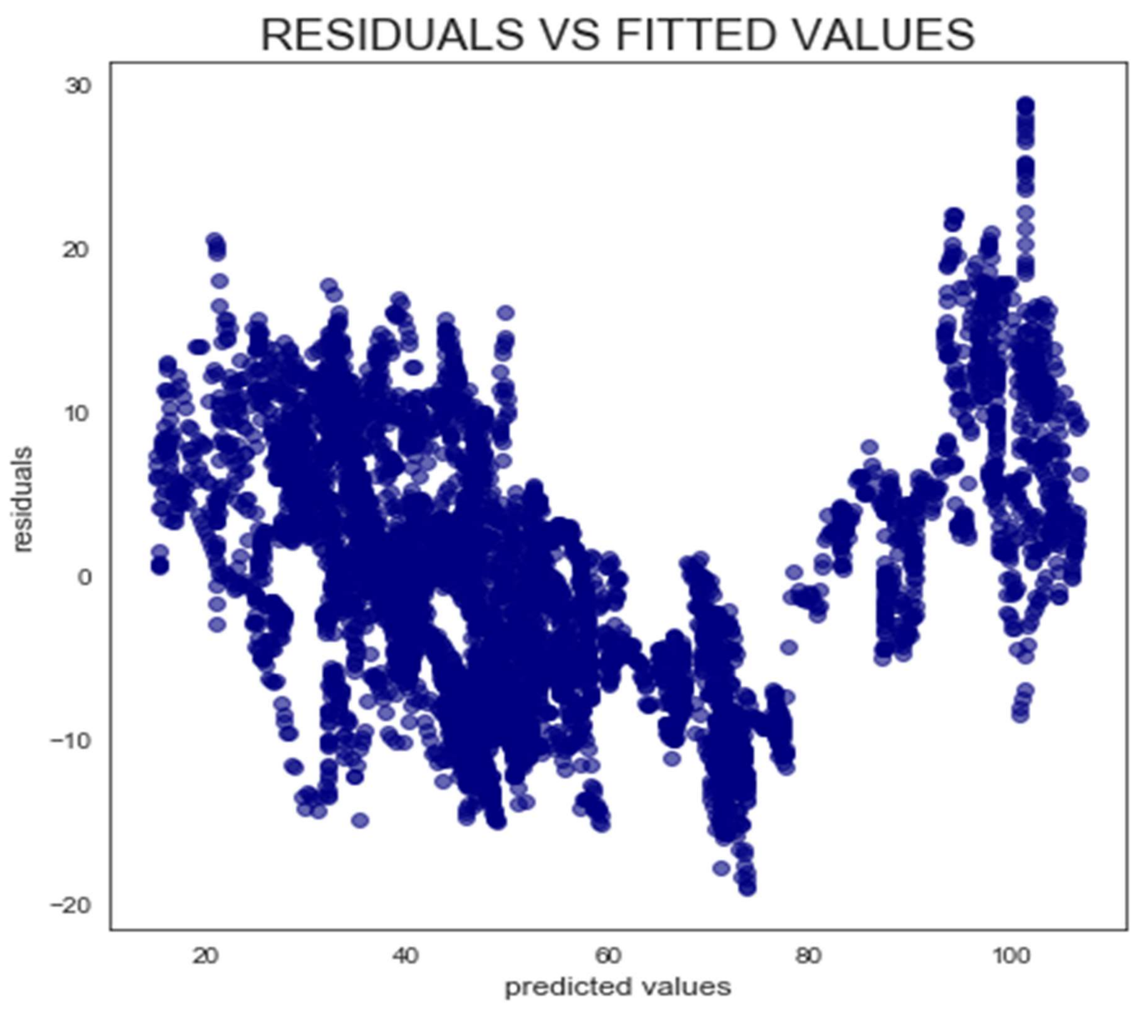


RESIDUAL VS PREDICTED VALUES: In general we should have irregularity between these two parameters

CODE:

```
plt.figure(figsize=(7,7))

plt.scatter(x=results2.fittedvalues,y=results2.resid,color="navy",alpha=0.6)

plt.xlabel("predicted values",fontsize=12)

plt.ylabel("residuals",fontsize=12)

plt.title("RESIDUALS VS FITTED VALUES",fontsize=18 )

plt.show()
```

output:

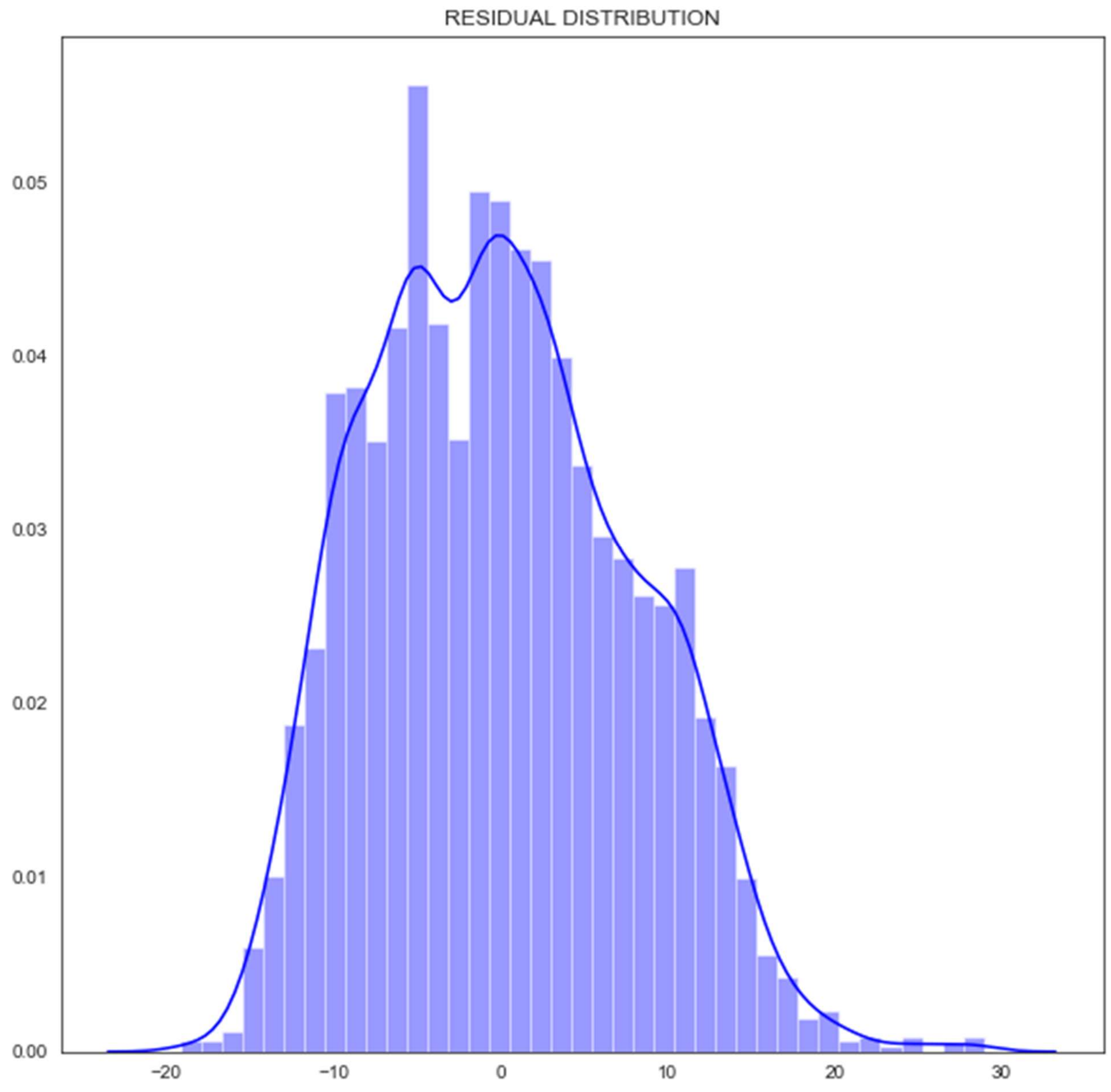


DISTRIBUTION OF RESIDUALS:

CODE:

```
residual_mean2=round(results2.resid.mean(),3)

residual_skew2=round(results2.resid.skew(),3)

plt.figure(figsize=(10,10))

sns.distplot(results2.resid,color="blue")

plt.title("RESIDUAL DISTRIBUTION")
```

OUTPUT:



MEAN SQUARED ERROR:

CODE:

```
print(results2.mse_resid)
```

OUTPUT:

```
60.263730672821495
```

ROOT MEAN SQUARED ERROR:

CODE:

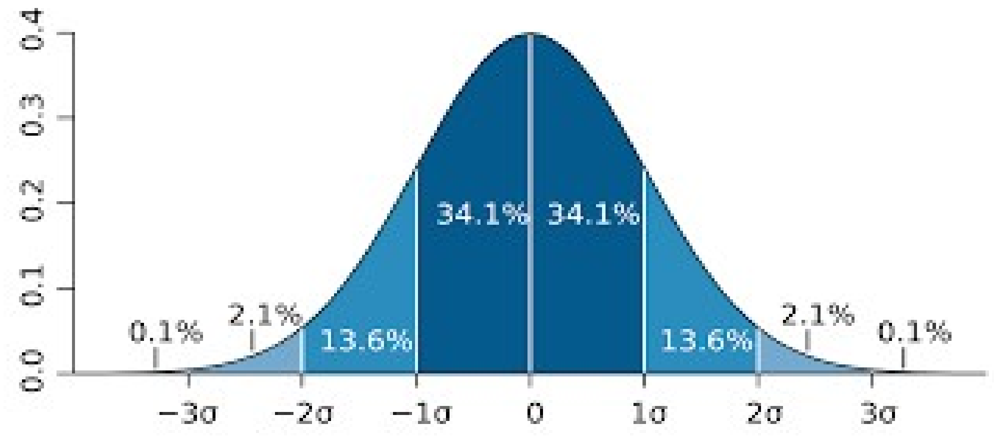
```
RMSE2=np.sqrt(results2.mse_resid)

pandas.DataFrame({"R-Squared":[results2.rsquared],"Mean Square Error":[results2.mse_resid],"Root Mean Square":np.sqrt(results2.mse_resid)},index=["JPM_CLOSE"])
```

OUTPUT:

```
              R-Squared  Mean    Square Error      Root Mean Square

JPM_CLOSE        0.887116          60.263731          7.762972
```

NORMAL DISTRIBUTION GRAPH:

Below is the figure representing normal distribution graph with standard deviations (1 sigma =68%; 2sigma=95%,3 sigma=100%)



CODE:

```
print("one standard deviation:",np.sqrt(results2.mse_resid)*1)

print("Two standard deviation:",np.sqrt(results2.mse_resid)*2)

print("Three standard deviation:",np.sqrt(results2.mse_resid)*3)
```

OUTPUT:

```
one standard deviation: 7.762971768132452

Two standard deviation: 15.525943536264904

Three standard deviation: 23.288915304397356
```

PREDICTING THE VALUES WITH ONE STANDARD DEVIATION:

In order to do this we defined a function that will plot the graph with predicted, upper, lower limit and actual graph.

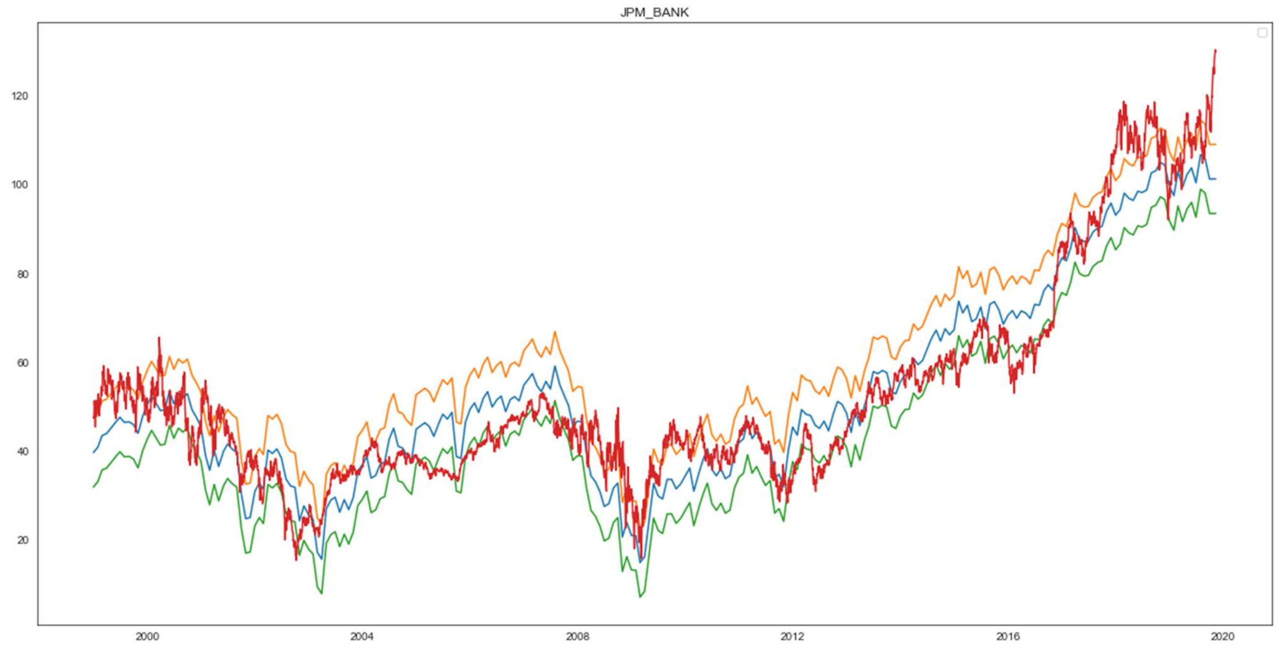
CODE:

```
def predict_Bank_price(dataset):
   a=regression2.predict(dataset)
   upper=a+1*RMSE2
   lower=a-1*RMSE2
   predict=pandas.DataFrame.from_records(a,columns=["PREDICTED PRICE"])
   upper=pandas.DataFrame.from_records(upper,columns=["UPPER PRICE"])
   lower=pandas.DataFrame.from_records(lower,columns=["LOWER PRICE"])
   ind=pandas.DataFrame(dataset.index)
   frames=[ind,predict,upper,lower]
   df=pandas.concat(frames,axis=1)
   df.set_index(["DATE"],inplace=True)
   print(df)
   plt.plot(df)
   plt.legend(loc='best')
   plt.show(block=False)
   return df
```

Calling function and plotting:

```
b2=predict_Bank_price(y2)
plt.figure(figsize=(20,10))
plt.plot(b2)
plt.plot(x2)
plt.title("JPM_BANK")
```

OUTPUT:



CONCLUSION:

The stock price of 4 listed companies in US stock market has been predicted using multi linear regression. Based on the stock of company either of the models can be used to predict the stock values the accuracy of stock models can be further improved by optimizing the model for relevant indicators and current market trends.