```
import pandas as pd
import numpy as np
from sklearn.neural_network import MLPRegressor,MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error,mean_squared_log_error,med



df=pd.read_csv("RELIANCEFINAL.csv")


#personalised normalization by junaid
x= df['Close'].max()
y= df['Close'].min()
z=df['Open'].max()
df['NOpen']=1.0
df['NClose']=1.0
df['NPreClose']=1.0
df['NPPreClose']=1.0
#print(x/y)
#(df['Close'])[1]=2
print((df['Close'])[1])
df.dropna()
for i in range(len(df['Close'])):
  (df['NClose'])[i]=float((df['Close'])[i])/x
  (df['NPreClose'])[i]=float((df['PreClose'])[i])/x
  (df['NOpen'])[i]=float((df['Open'])[i])/z
  (df['NPPreClose'])[i]=float((df['PPreClose'])[i])/x


#df.dtypes
df.min()
```

```
530.323059
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
  app.launch_new_instance()
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/ind
Unnamed: 0                                                    0
Unnamed: 0.1                                                  0
Open                                                    334.331
Date                                                 01-01-2013
```

```
Close                                                    334.876
Subjectivity                                                   0
Polarity                                                    -0.8
Compound                                                 -0.9988
Negative                                                       0
Neutral                                                    0.633
Positive                                                       0
label                                                          0
headlines       \tA proposal by Axis bank to increase the numb...
sentences       ['"All branches of State Bank of Indore will f...
scores                          [-0.501, -0.847, -0.829, -0.981]
scores_sum                                                 -5.88
PreClose                                                 334.876
PPreClose                                                334.876
NOpen                                                   0.208978
NClose                                                  0.209975
NPreClose                                               0.209975
NPPreClose                                              0.209975
dtype: object
```

```
#FOR PERSOANLISED SCALLER
#X=df[['Compound','Negative','Neutral','Positive','label','NPPreClose','Subjectivity','Polarity','scor
X=df[['Negative','Neutral','Positive','NPreClose']]
Y=df[['NPreClose']]
Z=df[['NOpen']]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,shuffle=False)



#change hidden layer,learning rate,activation function,validation fraction...etc
#i have used relu insted of logistics ,basic hidden layers =2 ,
MLPC=MLPRegressor(activation='relu',hidden_layer_sizes=(2),solver='lbfgs',learning_rate='adaptive',max_
#MLPC=MLPRegressor(hidden_layer_sizes=(3,2), activation='logistic',solver='lbfgs', alpha=0.00001, lear
# MLP=MLPClassifier(hidden_layer_sizes=(3,2),)
#MLP=MLPRegressor(hidden_layer_sizes=(5,2),learning_rate='adaptive')
#MLPC=MLPClassifier(hidden_layer_sizes=(100, 200), activation='relu',solver='lbfgs', alpha=0.005, lear


MLPC.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:1342: Dat
  y = column_or_1d(y, warn=True)
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=2, learning_rate='adaptive',
             learning_rate_init=0.001, max_fun=15000, max_iter=300000,
             momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
             power_t=0.5, random_state=None, shuffle=True, solver='lbfgs',
             tol=0.0001, validation_fraction=0.1, verbose=False,
             warm_start=False)
```

```
import numpy as np
from collections import defaultdict
#this one is written for the normal normalization written by me
```

```
#npreclose can be replaced by nopen for intraday prediction
pred=[]
t=[]

o=0
#print(x)
for i, row in X_test.iterrows():
  p=[]
  t=row
  #p.append(row['Compound'])
  p.append(row['Negative'])
  p.append(row['Neutral'])
  p.append(row['Positive'])
  #p.append(row['label'])
  p.append(row['NPreClose'])
  #p.append(row['Subjectivity'])
  #p.append(row['Polarity'])
  #p.append(row['scores_sum'])




 # print(p)

  q=np.reshape(p,(1,4))
  pred.append(MLPC.predict(q))
  if i<len(X_test)-1:
    X_test[i+1,5]=pred[i]
   # print (pred[i],Y_test[i],X_test[i+1,5])



#loop though x train and predict y train and then update next xtrain and preddict next y train and get

#pred=MLP.predict(x)
#print(pred, Y_test[0])




#here i need to predict recod by record

#the below prediction is based on the real previious close values
#we have stored predications in pred variable and use it for getting the real accuracy
#Y_predicted=MLPC.predict(X_test)
biaspred=[]
for i in range(0,len(pred)):
  biaspred.append(pred[i]+0.0)
Y_predicted=biaspred
#here we have added necessary bias to increase the model's accuracy

#print(Y_predicted[0],Y_test[0])
from math import sqrt
print("R2 SCORE ")
```

```
print(r2_score(Y_test,Y_predicted))
print("mean absolute error ")

print(mean_absolute_error(Y_test,Y_predicted))
print("mean squared error")
print(mean_squared_error(Y_test,Y_predicted))
print("root mean squared error")
print(sqrt(mean_squared_error(Y_test,Y_predicted)))
#print("mean squared log error ")
#print(mean_squared_log_error(Y_test,Y_predicted))
print("median absolute error ")
print(median_absolute_error(Y_test,Y_predicted))


#on increasing the hidden to 5,2 we reduced mean square error by 2% when we employed subjectivity and
```

```
R2 SCORE
0.9994158503219165
mean absolute error
0.0020930617446924494
mean squared error
6.9769891703446016e-06
root mean squared error
0.0026413990933489397
median absolute error
0.0018410392152206079
```

```
#test accuracy
import numpy as np
bp=[]
for i in range(0,len(pred)):
  bp.append(pred[i]+0.0)
a = np.array(Y_test)   # actual labels
b = np.array(bp)   # predicted labels
trend=[]
ptrend=[]
trendn=[]
ptrendn=[]
#accuracy =0
#TREND ACcuracy
#checking trend of the original data
trend.append(0)
for i in range (1,len(a)):

  if (a[i]-a[i-1])>0:
    trend.append(1)
  if (a[i]-a[i-1])<0:
    trend.append(-1)
  if(a[i]==a[i-1]):
    trend.append(0)



#print(trend)
#check the trend of the predicted data
ptrend.append(0)
```

```
ptrend.append(0)
for i in range (1,len(b)):

  if (b[i]-b[i-1])>0:
    ptrend.append(1)
  if (b[i]-b[i-1])<0:
    ptrend.append(-1)
  if(b[i]==b[i-1]):
    ptrend.append(0)
#print(ptrend)









v=[]
w=[]
#calculation of trend accuracy :
tp=0
tn=0
tnu=0
fnu=0
fp=0
fn=0
#for i in range(0,int(len(trend))):
for i in range(0,30):
  if (trend[i]==ptrend[i]):
    if trend[i]==1:
      tp=tp+1
    if trend[i]==0:
      tp=tp+1
    if trend[i]==-1:
      tn=tn+1

  if (trend[i]==1 and ptrend[i]==-1):
    fn=fn+1
  if (trend[i]==-1 and ptrend[i]==1):
    fp=fp+1
#down from here i will handle some extra 4 cases which were left last time
  if (trend[i]==0 and ptrend[i]==-1):
    fn=fn+1
  if (trend[i]==0 and ptrend[i]==1):
    tp=tp+1
  if (trend[i]==1 and ptrend[i]==0):
    tp=tp+1
  if (trend[i]==-1 and ptrend[i]==0):
    fp=fp+1



#extra two cases added in case the prdicted model show no change while the model shows positive or nega

print("TP:TN:FP:FN:TNU:FNU")
print (tp,tn,fp,fn,tnu,fnu)
print("precion ")
```

```
precision =(tp+tnu)/(fp+tp+tnu)
print(precision)
recall = (tp+tnu)/(fn+tp+tnu)
print("recall")
print(recall)

print("accuracy ")
print((tp+tn+tnu)/(tp+tn+fp+fn+tnu+fnu))
#F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score/)
fscore= (2*precision*recall)/(precision + recall)
print("F1 SCORE")
print(fscore)
for i in range(0, int(len(a))):



 #b[i]=b[i]
  v.append(a[i]*x)
  w.append(b[i]*x)




from sklearn.metrics import mean_absolute_error as mae
print("MAE")
print(mae(v,w))
print(mae(a,b))


def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100
print("MAPE")
print(mape(v,w))
print(mape(a,b))


def mse(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.square(np.subtract(actual,pred)).mean()
print('MSE')
print(mse(v,w))
print(mse(a,b))
print("RMSE")
print(sqrt(mse(a,b)*x))
```

```
TP:TN:FP:FN:TNU:FNU
15 15 0 0 0 0
precion
1.0
recall
1.0
accuracy
```

```
     1.0
     F1 SCORE
     1.0
     MAE
     3.3380831921569842
     0.0020930617446924494
     MAPE
     0.2687708772818461
     0.2687708772818464
     MSE
     17.745910250200406
     6.9769891703446016e-06
     RMSE
     0.10548521257382984


# here we have to check trend with span of n days, FUTURE TREND
#this code block is n day test accuracy

n_days=5
for i in range(0,n_days):
  trendn.append(0)

for i in range (n_days,(len(a))):

  if (a[i]-a[i-n_days])>0:
    trendn.append(1)
  if (a[i]-a[i-n_days])<0:
    trendn.append(-1)
  if (a[i]==a[i-n_days]):
    trendn.append(0)
#above is real n day trend and now we will add the predicted n day trend
for i in range(0,n_days):
  ptrendn.append(0)

for i in range (n_days,(len(b))):

  if (b[i]-b[i-n_days])>0:
    ptrendn.append(1)
  if (b[i]-b[i-n_days])<0:
    ptrendn.append(-1)
  if (b[i]==b[i-n_days]):
    ptrendn.append(0)



v=[]
w=[]
#calculation of trend accuracy :
tp=0
tn=0
tnu=0
fnu=0
fp=0
fn=0
#for i in range(0,int(len(trend))):
for i in range(0,30):
```

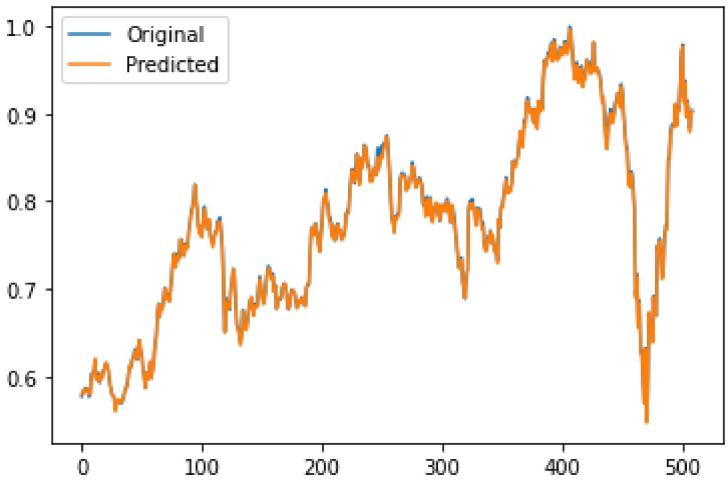
```
  if (trendn[i]==ptrendn[i]):
    if trendn[i]==1:
      tp=tp+1
    if trendn[i]==0:
      tp=tp+1
    if trendn[i]==-1:
      tn=tn+1

  if (trendn[i]==1 and ptrendn[i]==-1):
    fn=fn+1
  if (trendn[i]==-1 and ptrendn[i]==1):
    fp=fp+1
#down from here i will handle some extra 4 cases which were left last time
  if (trendn[i]==0 and ptrendn[i]==-1):
    fn=fn+1
  if (trendn[i]==0 and ptrendn[i]==1):
    tp=tp+1
  if (trendn[i]==1 and ptrendn[i]==0):
    tp=tp+1
  if (trendn[i]==-1 and ptrendn[i]==0):
    fp=fp+1

#extra two cases added in case the prdicted model show no change while the model shows positive or neg

print("TP:TN:FP:FN:TNU:FNU")
print (tp,tn,fp,fn,tnu,fnu)
print("precion ")
precision =(tp)/(fp+tp)
print(precision)
recall = (tp)/(fn+tp)
print("recall")
print(recall)

print("accuracy ")
print((tp+tn)/(tp+tn+fp+fn))
#F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score/)
fscore= (2*precision*recall)/(precision + recall)
print("F1 SCORE")
print(fscore)
for i in range(0, int(len(a))):



 #b[i]=b[i]
  v.append(a[i]*x)
  w.append(b[i]*x)




from sklearn.metrics import mean_absolute_error as mae
print("MAE")
print(mae(v,w))
print(mae(a,b))
```

```
def mape(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.mean(np.abs((actual - pred) / actual)) * 100
print("MAPE")
print(mape(v,w))
print(mape(a,b))


def mse(actual, pred):
    actual, pred = np.array(actual), np.array(pred)
    return np.square(np.subtract(actual,pred)).mean()
print('MSE')
print(mse(v,w))
print(mse(a,b))
print("RMSE")
print(sqrt(mse(a,b)*x))
```

```
     TP:TN:FP:FN:TNU:FNU
     18 12 0 0 0 0
     precion
     1.0
     recall
     1.0
     accuracy
     1.0
     F1 SCORE
     1.0
     MAE
     3.3380831921569842
     0.0020930617446924494
     MAPE
     0.2687708772818461
     0.2687708772818464
     MSE
     17.745910250200406
     6.9769891703446016e-06
     RMSE
     0.10548521257382984
```

```
#for i in range (0,15):
 # print(trend[i],ptrend[i])
  #print("next day")
#store more than one models and plot together
sdl=b



#pdl=b


#sdnl=b


#pdnl=b
```

```
#here we will start  plotting
import matplotlib.pyplot as plt
day=[]
for i in range(len(a)):
  day.append(i)


# plot lines
plt.plot(day, a, label = "Original")
plt.plot(day, sdl, label = "Predicted")
#plt.plot(a, sdl, label = "PREDICTED[SDNL]")
#plt.plot(day, pdl, label = "PREDICTED[PDL]")
#plt.plot(day, pdnl, label = "PREDICTED[PDNL]")

plt.legend()
#plt.figure(figsize=(200,200))

#image = plt.figure(figsize=(50,50), dpi= 500)

plt.savefig("Relaince.png",dpi=500)
plt.show()
from google.colab import files

#files.download('Relaince.png')
```



✓ 0s completed at 2:30 PM