# Codes

February 6, 2021

[6]:
```
!pip install pytse_client
```

```
Collecting pytse_client
  Downloading https://files.pythonhosted.org/packages/aa/df/bfb78e36240e364f2645
e2b3bfeafe116f5d1cf91c359e8d2878c588a664/pytse_client-0.6.3-py3-none-any.whl
Collecting jdatetime<4.0.0,>=3.6.2
  Downloading https://files.pythonhosted.org/packages/fa/a9/2c9f8ff1c126835e497e
23f2a5a69fcd59ea2ca11030db310bdbd8c6fe76/jdatetime-3.6.2.tar.gz
Requirement already satisfied: requests<3.0.0,>=2.23.0 in
/usr/local/lib/python3.6/dist-packages (from pytse_client) (2.23.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages
(from pytse_client) (1.1.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.6/dist-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-
packages (from requests<3.0.0,>=2.23.0->pytse_client) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.6/dist-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (2020.12.5)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.6/dist-packages (from pandas->pytse_client) (2.8.1)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-
packages (from pandas->pytse_client) (1.19.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-
packages (from pandas->pytse_client) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-
packages (from python-dateutil>=2.7.3->pandas->pytse_client) (1.15.0)
Building wheels for collected packages: jdatetime
  Building wheel for jdatetime (setup.py) ... done
  Created wheel for jdatetime: filename=jdatetime-3.6.2-cp36-none-any.whl
size=11783
sha256=5d077b77600bf3d6e138d7e8835e658b989094d5684a4f4e16c9b2bca7e80f78
  Stored in directory: /root/.cache/pip/wheels/ea/7d/d2/92961c39b79a0556bc4ce7c2
0185fdab84969ecb8fe2e5e8cc
```

```
Successfully built jdatetime
Installing collected packages: jdatetime, pytse-client
Successfully installed jdatetime-3.6.2 pytse-client-0.6.3
```

[1]:
```
import pytse_client as tse
```

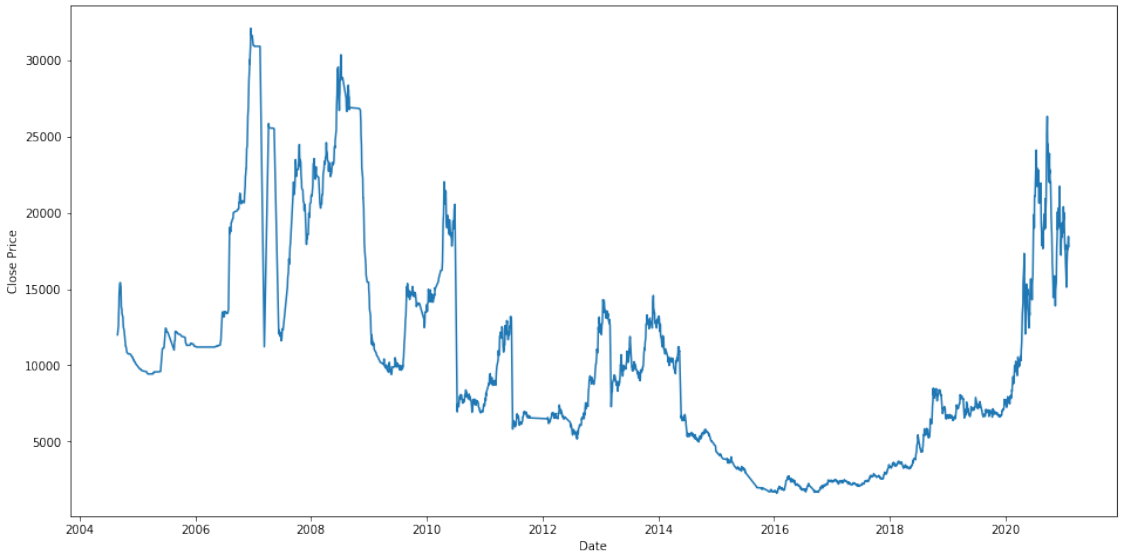
[6]:
```
tickers = tse.download(symbols="")
df=tickers[""]
df=df.reset_index()
```

### 0.0.1 Import Packages

[4]:
```
#import packages
import pandas as pd
import numpy as np

#to plot within notebook
import matplotlib.pyplot as plt
%matplotlib inline

#setting figure size
from matplotlib.pylab import rcParams
rcParams['figure.figsize'] = 20,10

#for normalizing data
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

[5]:
```
#setting index as date
df['date'] = pd.to_datetime(df.date,format='%Y-%m-%d')
df.index = df['date']

#plot
plt.figure(figsize=(16,8))
plt.plot(df['close'], label='Close Price history')
plt.xlabel("Date")
plt.ylabel("Close Price")
```

[5]:
```
Text(0, 0.5, 'Close Price')
```




[6]:
```
# setting the index as date
df['date'] = pd.to_datetime(df.date,format='%Y-%m-%d')
df.index = df['date']

#creating dataframe with date and the target variable
data = df.sort_index(ascending=True, axis=0)
new_data = pd.DataFrame(index=range(0,len(df)),columns=['date', 'close'])

for i in range(0,len(data)):
     new_data['date'][i] = data['date'][i]
     new_data['close'][i] = data['close'][i]
```

[7]:
```
#splitting into train and validation
train = new_data[:int(0.8*len(df))]
valid = new_data[int(0.8*len(df)):]

# shapes of training set
print('\n Shape of training set:')
print(train.shape)

# shapes of validation set
print('\n Shape of validation set:')
print(valid.shape)
```

```
 Shape of training set:
(2679, 2)

 Shape of validation set:
```

```
(670, 2)
```

### 0.1 Moving Average

[8]:
```
preds = []
for i in range(0,valid.shape[0]):
    a = train['close'][len(train)-670+i:].sum() + sum(preds)
    b = a/670
    preds.append(b)
```

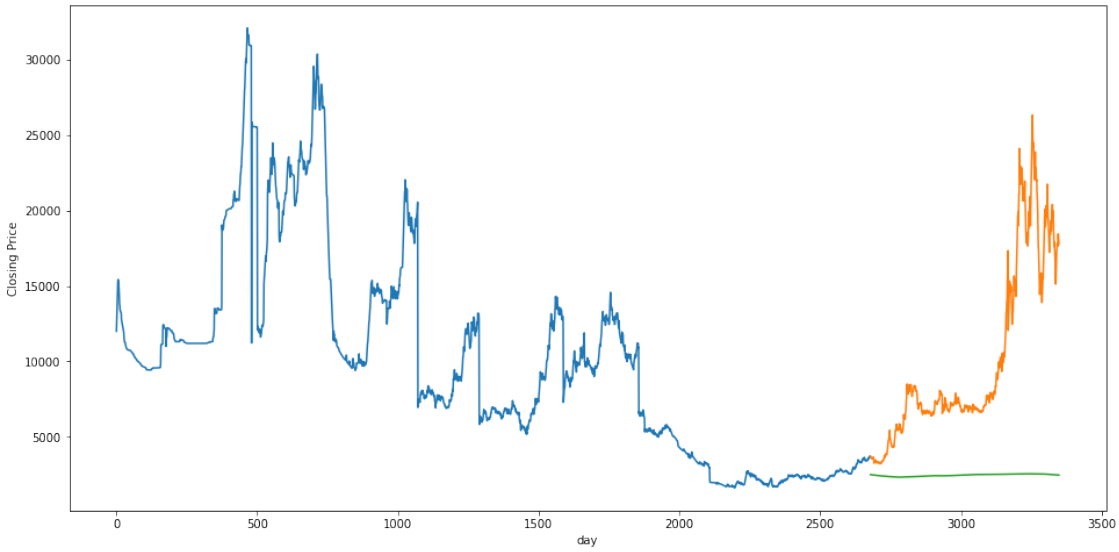
[9]:
```
# checking the results (RMSE value)
rms=np.sqrt(np.mean(np.power((np.array(valid['close'])-preds),2)))
print('\n RMSE value on validation set:')
print(rms)
```

```
 RMSE value on validation set:
9421.57997563058
```

[10]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.figure(figsize=(16,8))
plt.plot(train['close'])
plt.plot(valid[['close', 'Predictions']])
plt.xlabel("day")
plt.ylabel("Closing Price")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[10]:
```
Text(0, 0.5, 'Closing Price')
```




## 0.2 Linear Regression

[25]:
```
#setting index as date values
df['date'] = pd.to_datetime(df.date,format='%Y-%m-%d')
df.index = df['date']

#sorting
data = df.sort_index(ascending=True, axis=0)

#creating a separate dataset
new_data = pd.DataFrame(index=range(0,len(df)),columns=['date', 'close'])

for i in range(0,len(data)):
    new_data['date'][i] = data['date'][i]
    new_data['close'][i] = data['close'][i]
```

[63]:
```
new_data["year"]=pd.DatetimeIndex(new_data['date']).year
new_data["pyear"]=pd.DatetimeIndex(new_data['date']).year
new_data["month"]=pd.DatetimeIndex(new_data['date']).month
new_data["pmonth"]=pd.DatetimeIndex(new_data['date']).month
new_data["day"]=pd.DatetimeIndex(new_data['date']).day
new_data["pday"]=pd.DatetimeIndex(new_data['date']).day
new_data["pweekday"]=pd.DatetimeIndex(new_data['date']).day
```

[64]:
```
#!pip install persiantools
from persiantools.jdatetime import JalaliDate
import datetime
```

```
for i in range(0,len(new_data)):
  new_data['pyear'][i]=JalaliDate(datetime.date(new_data.year[i], new_data.
 ↪month[i], new_data.day[i])).year
  new_data['pmonth'][i]=JalaliDate(datetime.date(new_data.year[i], new_data.
 ↪month[i], new_data.day[i])).month
  new_data['pday'][i]=JalaliDate(datetime.date(new_data.year[i], new_data.
 ↪month[i], new_data.day[i])).day
  new_data['pweekday'][i]=JalaliDate(datetime.date(new_data.year[i], new_data.
 ↪month[i], new_data.day[i])).weekday()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6:
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/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7:
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/indexing.html#returning-a-view-versus-a-copy
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8:
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/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9:
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/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

[67]:
```
new_data.drop('date', axis=1, inplace=True)
```

[68]:
```
#split into train and validation
train = new_data[:2800]
valid = new_data[2800:]

x_train = train.drop('close', axis=1)
y_train = train['close']
```

```
x_valid = valid.drop('close', axis=1)
y_valid = valid['close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

[68]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

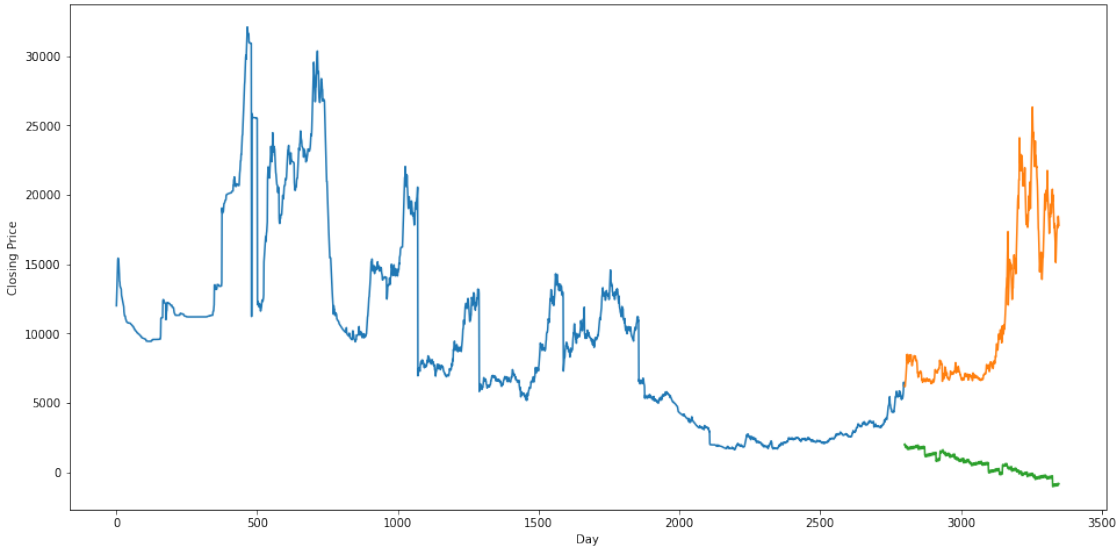
[71]:
```
#make predictions and find the rmse
preds = model.predict(x_valid)
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

[71]:
```
12311.415187756978
```

[73]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[2800:].index
train.index = new_data[:2800].index
plt.figure(figsize=(16,8))
plt.plot(train['close'])
plt.plot(valid[['close', 'Predictions']])
plt.xlabel("Day")
plt.ylabel("Closing Price")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

[73]: Text(0, 0.5, 'Closing Price')




### 0.3 KNN

[74]:
```
#importing libraries
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
```

[75]:
```
#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)

#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

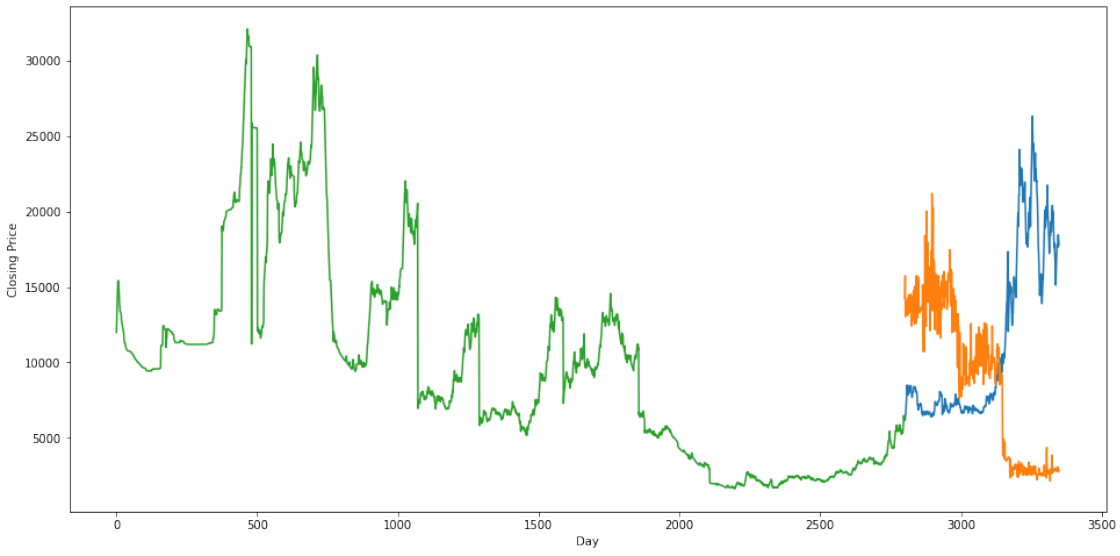
[76]:
```
#rmse
rms=np.sqrt(np.mean(np.power((np.array(y_valid)-np.array(preds)),2)))
rms
```

[76]:
```
10449.254642820486
```

[78]:
```
#plot
plt.figure(figsize=(16,8))
plt.xlabel("Day")
plt.ylabel("Closing Price")
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(valid[['close', 'Predictions']])
plt.plot(train['close'])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[78]:
```
[<matplotlib.lines.Line2D at 0x7f288fedccf8>]
```



### 0.4 ARMIA

[33]:
```
!pip install pmdarima
from pmdarima import auto_arima
data = df.sort_index(ascending=True, axis=0)

train = data[:2800]
valid = data[2800:]

training = train['close']
validation = valid['close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3,␣
 ↪m=12,start_P=0, seasonal=True,d=1, D=1,␣
 ↪trace=True,error_action='ignore',suppress_warnings=True)
model.fit(training)

forecast = model.predict(n_periods=549)
```

```
Collecting pmdarima
  Downloading https://files.pythonhosted.org/packages/c9/d7/61af1897449638
822f97c8b43ef0c2fce2ec68a6cda9a43ebbbdd12b967c/pmdarima-1.8.0-cp36-cp36m-manylin
ux1_x86_64.whl (1.5MB)
     || 1.5MB 5.7MB/s
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.6/dist-packages (from pmdarima) (1.0.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.6/dist-packages
(from pmdarima) (1.24.3)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.6/dist-
packages (from pmdarima) (1.1.5)
Collecting statsmodels!=0.12.0,>=0.11
  Downloading https://files.pythonhosted.org/packages/0d/7b/c17815648dc313
96af865b9c6627cc3f95705954e30f61106795361c39ee/statsmodels-0.12.2-cp36-cp36m-man
ylinux1_x86_64.whl (9.5MB)
     || 9.5MB 21.7MB/s
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in
/usr/local/lib/python3.6/dist-packages (from pmdarima) (53.0.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.6/dist-
packages (from pmdarima) (1.4.1)
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.6/dist-packages (from pmdarima) (0.22.2.post1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.6/dist-
packages (from pmdarima) (1.19.5)
Collecting Cython<0.29.18,>=0.29
  Downloading https://files.pythonhosted.org/packages/e7/d7/510ddef0248f3e
1e91f9cc7e31c0f35f8954d0af92c5c3fd4c853e859ebe/Cython-0.29.17-cp36-cp36m-manylin
ux1_x86_64.whl (2.1MB)
     || 2.1MB 52.1MB/s
```

```
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.6/dist-packages (from pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-
packages (from pandas>=0.19->pmdarima) (2018.9)
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.6/dist-
packages (from statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-
packages (from python-dateutil>=2.7.3->pandas>=0.19->pmdarima) (1.15.0)
Installing collected packages: statsmodels, Cython, pmdarima
  Found existing installation: statsmodels 0.10.2
    Uninstalling statsmodels-0.10.2:
      Successfully uninstalled statsmodels-0.10.2
  Found existing installation: Cython 0.29.21
    Uninstalling Cython-0.29.21:
      Successfully uninstalled Cython-0.29.21
Successfully installed Cython-0.29.17 pmdarima-1.8.0 statsmodels-0.12.2
Performing stepwise search to minimize aic
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=inf, Time=18.01 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=46042.126, Time=0.14 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=45173.226, Time=5.96 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=inf, Time=7.83 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=45905.290, Time=0.27 sec
 ARIMA(1,1,0)(2,1,0)[12]             : AIC=44887.833, Time=13.25 sec
 ARIMA(1,1,0)(2,1,1)[12]             : AIC=inf, Time=29.02 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=inf, Time=9.67 sec
 ARIMA(0,1,0)(2,1,0)[12]             : AIC=44995.390, Time=3.63 sec
 ARIMA(2,1,0)(2,1,0)[12]             : AIC=44886.883, Time=19.52 sec
 ARIMA(2,1,0)(1,1,0)[12]             : AIC=45172.159, Time=8.35 sec
 ARIMA(2,1,0)(2,1,1)[12]             : AIC=inf, Time=33.81 sec
 ARIMA(2,1,0)(1,1,1)[12]             : AIC=inf, Time=14.50 sec
 ARIMA(3,1,0)(2,1,0)[12]             : AIC=44888.178, Time=20.37 sec
 ARIMA(2,1,1)(2,1,0)[12]             : AIC=inf, Time=30.12 sec
 ARIMA(1,1,1)(2,1,0)[12]             : AIC=44886.690, Time=21.88 sec
 ARIMA(1,1,1)(1,1,0)[12]             : AIC=45169.889, Time=7.80 sec
 ARIMA(1,1,1)(2,1,1)[12]             : AIC=inf, Time=50.42 sec
 ARIMA(1,1,1)(1,1,1)[12]             : AIC=inf, Time=17.03 sec
 ARIMA(0,1,1)(2,1,0)[12]             : AIC=44884.863, Time=12.98 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=45168.072, Time=5.83 sec
 ARIMA(0,1,1)(2,1,1)[12]             : AIC=inf, Time=31.20 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=inf, Time=10.28 sec
 ARIMA(0,1,2)(2,1,0)[12]             : AIC=44886.682, Time=16.99 sec
 ARIMA(1,1,2)(2,1,0)[12]             : AIC=inf, Time=40.97 sec
 ARIMA(0,1,1)(2,1,0)[12] intercept   : AIC=44886.860, Time=18.97 sec

Best model:  ARIMA(0,1,1)(2,1,0)[12]
Total fit time: 448.794 seconds
```

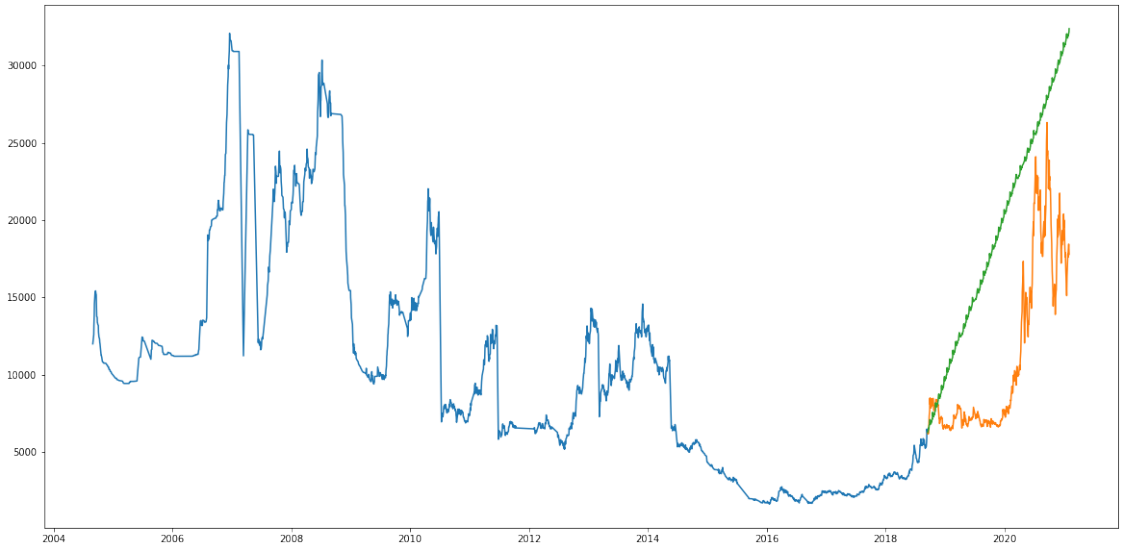
[34]:
```
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

[36]:
```
rms=np.sqrt(np.mean(np.power((np.array(valid['close'])-np.
 ↪array(forecast['Prediction'])),2)))
rms
```

[36]: 9171.867692760092

[38]:
```
#plot
plt.plot(train['close'])
plt.plot(valid['close'])
plt.plot(forecast['Prediction'])
```

[38]: [<matplotlib.lines.Line2D at 0x7f7dde8aceb8>]




## 0.5 Prophet

[41]:
```
#importing prophet
from fbprophet import Prophet

#creating dataframe
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['date'][i]
    new_data['Close'][i] = data['close'][i]

new_data['Date'] = pd.to_datetime(new_data.Date,format='%Y-%m-%d')
```

```
new_data.index = new_data['Date']

#preparing data
new_data.rename(columns={'Close': 'y', 'Date': 'ds'}, inplace=True)

#train and validation
train = new_data[:987]
valid = new_data[987:]

#fit the model
model = Prophet()
model.fit(train)

#predictions
close_prices = model.make_future_dataframe(periods=len(valid))
forecast = model.predict(close_prices)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with
daily_seasonality=True to override this.
```

[42]:
```
#rmse
forecast_valid = forecast['yhat'][987:]
rms=np.sqrt(np.mean(np.power((np.array(valid['y'])-np.array(forecast_valid)),2)))
rms
```

[42]:
```
6145.408288096017
```

[43]:
```
#plot
valid['Predictions'] = 0
valid['Predictions'] = forecast_valid.values

plt.plot(train['y'])
plt.plot(valid[['y', 'Predictions']])
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[43]:
```
[<matplotlib.lines.Line2D at 0x7f7ddb977e48>,
 <matplotlib.lines.Line2D at 0x7f7ddb9d9400>]
```


## 0.6 LSTM

[214]:
```
import numpy as np # linear algebra
import random
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from pandas import datetime
import math, time
import itertools
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
import datetime
from operator import itemgetter
from sklearn.metrics import mean_squared_error
from math import sqrt
import torch
import torch.nn as nn
from torch.autograd import Variable
```

[216]:
```
tickers = tse.download(symbols="")
df=tickers[""]
df=df.reset_index()
```

[217]:
```
df.index = df['date']
df=df[['close']]
```

[218]:
```
scaler = MinMaxScaler(feature_range=(-1, 1))
df['close'] = scaler.fit_transform(df['close'].values.reshape(-1,1))
```

[219]:
```
def load_data(stock, look_back):
    data_raw = stock.values
    data = []
    for index in range(len(data_raw) - look_back):
        data.append(data_raw[index: index + look_back])

    data = np.array(data);
    test_set_size = int(np.round(0.2*data.shape[0]));
    train_set_size = data.shape[0] - (test_set_size);

    x_train = data[:train_set_size,:-1,:]
    y_train = data[:train_set_size,-1,:]

    x_test = data[train_set_size:,:-1]
    y_test = data[train_set_size:,-1,:]

    return [x_train, y_train, x_test, y_test]

look_back = 60
x_train, y_train, x_test, y_test = load_data(df, look_back)
print('x_train.shape = ',x_train.shape)
print('y_train.shape = ',y_train.shape)
print('x_test.shape = ',x_test.shape)
print('y_test.shape = ',y_test.shape)
```

```
x_train.shape =  (2631, 59, 1)
y_train.shape =  (2631, 1)
x_test.shape =  (658, 59, 1)
y_test.shape =  (658, 1)
```

[220]:
```
x_train = torch.from_numpy(x_train).type(torch.Tensor)
x_test = torch.from_numpy(x_test).type(torch.Tensor)
y_train = torch.from_numpy(y_train).type(torch.Tensor)
y_test = torch.from_numpy(y_test).type(torch.Tensor)
```

[221]:
```
y_train.size(),x_train.size()
```

[221]:
```
(torch.Size([2631, 1]), torch.Size([2631, 59, 1]))
```

[222]:
```
n_steps = look_back-1
batch_size = 32
num_epochs = 100

train = torch.utils.data.TensorDataset(x_train,y_train)
test = torch.utils.data.TensorDataset(x_test,y_test)

train_loader = torch.utils.data.DataLoader(dataset=train,
                                           batch_size=batch_size,
                                           shuffle=False)

test_loader = torch.utils.data.DataLoader(dataset=test,
                                          batch_size=batch_size,
                                          shuffle=False)
```

[223]:
```
input_dim = 1
hidden_dim = 32
num_layers = 2
output_dim = 1
class LSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers, output_dim):
        super(LSTM, self).__init__()
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers
        self.lstm = nn.LSTM(input_dim, hidden_dim, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
 ↪requires_grad_()
        c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_dim).
 ↪requires_grad_()
        out, (hn, cn) = self.lstm(x, (h0.detach(), c0.detach()))
        out = self.fc(out[:, -1, :])
        return out

model = LSTM(input_dim=input_dim, hidden_dim=hidden_dim, output_dim=output_dim,␣
 ↪num_layers=num_layers)

loss_fn = torch.nn.MSELoss()

optimiser = torch.optim.Adam(model.parameters(), lr=0.01)
print(model)
print(len(list(model.parameters())))
for i in range(len(list(model.parameters()))):
```

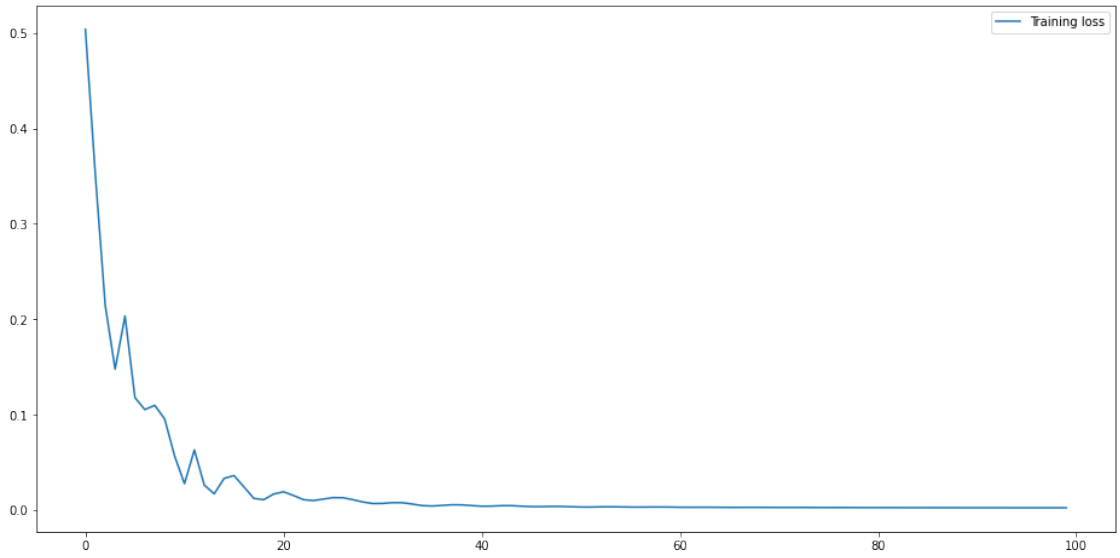
```
    print(list(model.parameters())[i].size())
```

```
LSTM(
  (lstm): LSTM(1, 32, num_layers=2, batch_first=True)
  (fc): Linear(in_features=32, out_features=1, bias=True)
)
10
torch.Size([128, 1])
torch.Size([128, 32])
torch.Size([128])
torch.Size([128])
torch.Size([128, 32])
torch.Size([128, 32])
torch.Size([128])
torch.Size([128])
torch.Size([1, 32])
torch.Size([1])
```

[224]:
```
hist = np.zeros(num_epochs)
seq_dim =look_back-1
for t in range(num_epochs):
    y_train_pred = model(x_train)

    loss = loss_fn(y_train_pred, y_train)
    if t % 10 == 0 and t !=0:
        print("Epoch ", t, "MSE: ", loss.item())
    hist[t] = loss.item()
    optimiser.zero_grad()
    loss.backward()
    optimiser.step()
```

```
Epoch  10 MSE:  0.027949582785367966
Epoch  20 MSE:  0.019462060183286667
Epoch  30 MSE:  0.007295519579201937
Epoch  40 MSE:  0.004391076508909464
Epoch  50 MSE:  0.0035073161125183105
Epoch  60 MSE:  0.0033111043740063906
Epoch  70 MSE:  0.003068132558837533
Epoch  80 MSE:  0.0028996511828154325
Epoch  90 MSE:  0.00279419869184494
```

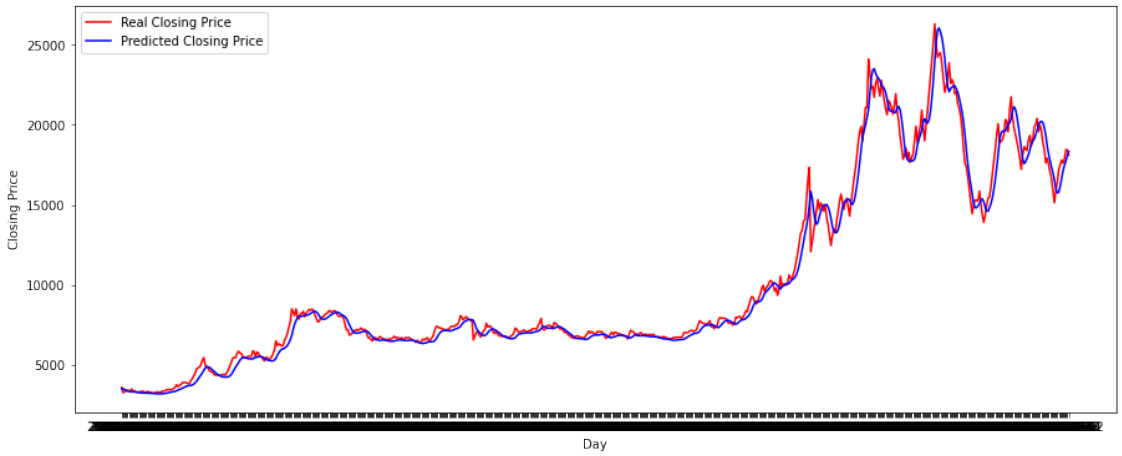
[225]:
```
plt.figure(figsize=(16,8))
plt.plot(hist, label="Training loss")
plt.legend()
plt.show()
```




[226]:
```
y_test_pred = model(x_test)
y_train_pred = scaler.inverse_transform(y_train_pred.detach().numpy())
y_train = scaler.inverse_transform(y_train.detach().numpy())
y_test_pred = scaler.inverse_transform(y_test_pred.detach().numpy())
y_test = scaler.inverse_transform(y_test.detach().numpy())

trainScore = math.sqrt(mean_squared_error(y_train[:,0], y_train_pred[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(y_test[:,0], y_test_pred[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

```
Train Score: 795.63 RMSE
Test Score: 677.86 RMSE
```

[230]:
```
figure, axes = plt.subplots(figsize=(15, 6))
axes.xaxis_()
axes.plot(df[len(df)-len(y_test):].index, y_test, color = 'red', label = 'Real␣
 ↪Closing Price')
axes.plot(df[len(df)-len(y_test):].index, y_test_pred, color = 'blue', label =␣
 ↪'Predicted Closing Price')
plt.xlabel('Day')
plt.ylabel('Closing Price')
plt.legend()
plt.show()
```




## 0.7 CNN

[2]:
```
import numpy as np
import pandas as pd
from numpy import array
import torch
import gc
import torch.nn as nn
from tqdm import tqdm_notebook as tqdm
from torch.utils.data import Dataset,DataLoader
```

[3]:
```
df = pd.read_csv('../input/tehran-stock-price/output.csv').
 ↪rename(columns={'Unnamed: 0':'
l+s+s1'}).set_index('
l+s+s1')
df=df[["close","open"]]
```

[4]:
```
test_set_size = int(np.round(0.2*df.shape[0]));
train_set_size = df.shape[0] - (test_set_size);
train_set =df[:train_set_size]
valid_set = df[train_set_size:]
print('Proportion of train_set : {:.2f}%'.format(len(train_set)/len(df)))
print('Proportion of valid_set : {:.2f}%'.format(len(valid_set)/len(df)))
```

```
Proportion of train_set : 0.80%
Proportion of valid_set : 0.20%
```

[5]:
```
def split_sequence(sequence, n_steps):
    x, y = list(), list()
    for i in range(len(sequence)):
```

```
        end_ix = i + n_steps

        if end_ix > len(sequence)-1:
            break
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        x.append(seq_x)
        y.append(seq_y)
    return array(x), array(y)
n_steps = 2
train_x,train_y = split_sequence(train_set["close"],n_steps)
valid_x,valid_y = split_sequence(valid_set["close"],n_steps)
```

### 0.7.1 Build CNN Forecast Model

[59]:
```
class StockDataset(Dataset):
    def __init__(self,feature,target):
        self.feature = feature
        self.target = target

    def __len__(self):
        return len(self.feature)

    def __getitem__(self,idx):
        item = self.feature[idx]
        label = self.target[idx]

        return item,label
```

[62]:
```
class CNN_ForecastNet(nn.Module):
    def __init__(self):
        super(CNN_ForecastNet,self).__init__()
        self.conv1d = nn.Conv1d(2,128,kernel_size=1)
        self.relu = nn.ReLU(inplace=True)
        self.fc1 = nn.Linear(128,50)
        self.fc2 = nn.Linear(50,1)

    def forward(self,x):
        x = self.conv1d(x)
        x = x.view(-1)
        x = self.relu(x)
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)

        return x
```

[67]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model = CNN_ForecastNet().to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)
criterion = nn.MSELoss()
```

[64]:
```
print(model)
```

```
CNN_ForecastNet(
  (conv1d): Conv1d(2, 128, kernel_size=(1,), stride=(1,))
  (relu): ReLU(inplace=True)
  (fc1): Linear(in_features=128, out_features=50, bias=True)
  (fc2): Linear(in_features=50, out_features=1, bias=True)
)
```

**Do I have to think autocorrelation when setting batch_size?** Maybe Batch_size is important, and It is related with autocorrelation I guess.

[72]:
```
train = StockDataset(train_x.reshape(train_x.shape[0],train_x.
 ↪shape[1],1),train_y)
valid =  StockDataset(valid_x.reshape(valid_x.shape[0],valid_x.
 ↪shape[1],1),valid_y)
train_loader = torch.utils.data.DataLoader(train,batch_size=1,shuffle=False)
valid_loader = torch.utils.data.DataLoader(train,batch_size=1,shuffle=False)
```

[75]:
```
train_losses = []
valid_losses = []
def Train():

    running_loss = .0

    model.train()

    for idx, (inputs,labels) in enumerate(train_loader):
        inputs = inputs.to(device)
        labels = labels.to(device)
        optimizer.zero_grad()
        preds = model(inputs.float())
        loss = criterion(preds,labels)
        loss.backward()
        optimizer.step()
        running_loss += loss**(0.5)

    train_loss = running_loss/len(train_loader)
    train_losses.append(train_loss.detach().numpy())

    print(f'train_loss {train_loss}')
```

```
def Valid():
    running_loss = .0

    model.eval()

    with torch.no_grad():
        for idx, (inputs, labels) in enumerate(valid_loader):
            inputs = inputs.to(device)
            labels = labels.to(device)
            #optimizer.zero_grad()
            preds = model(inputs.float())
            loss = criterion(preds,labels)
            running_loss += loss**(0.5)

        valid_loss = running_loss/len(valid_loader)
        valid_losses.append(valid_loss.detach().numpy())
        print(f'valid_loss {valid_loss}')
```

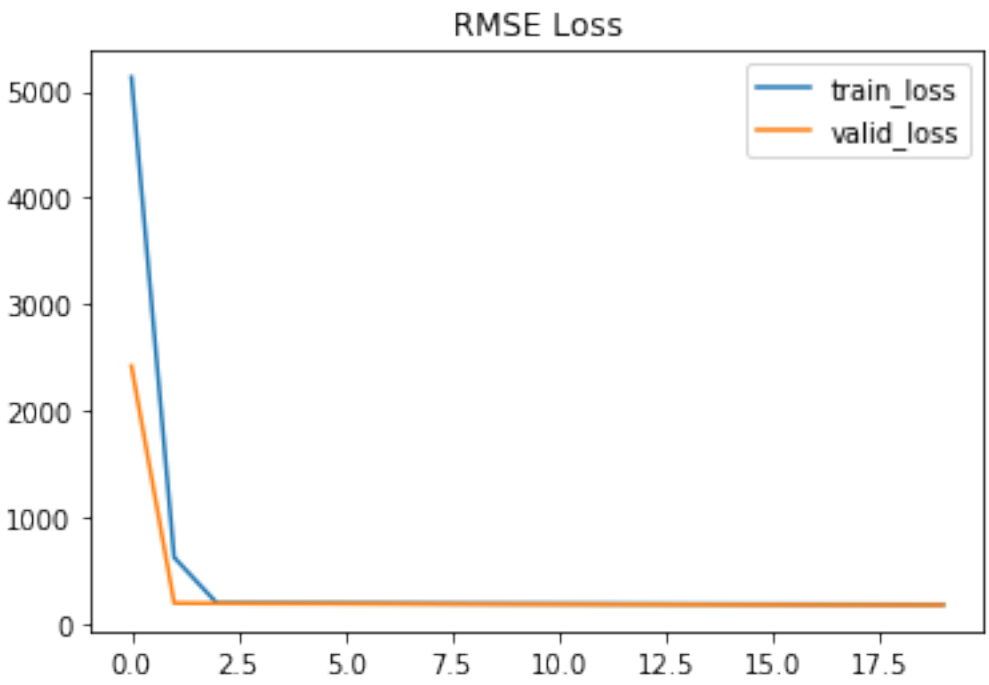
[77]:
```
epochs = 20
for epoch in range(epochs):
    print('epochs {}/{}'.format(epoch+1,epochs))
    Train()
    Valid()
    gc.collect()
```

```
epochs 1/20
train_loss 188.8728485107422
valid_loss 184.86624145507812
epochs 2/20
train_loss 188.8064727783203
valid_loss 184.2938232421875
epochs 3/20
train_loss 188.2359619140625
valid_loss 183.78916931152344
epochs 4/20
train_loss 187.732177734375
valid_loss 183.2985076904297
epochs 5/20
train_loss 187.268798828125
valid_loss 182.82408142089844
epochs 6/20
train_loss 186.78408813476562
valid_loss 182.35791015625
epochs 7/20
train_loss 186.34613037109375
valid_loss 181.8966827392578
epochs 8/20
```

```
train_loss 185.90756225585938
valid_loss 181.44407653808594
epochs 9/20
train_loss 185.46580505371094
valid_loss 181.00173950195312
epochs 10/20
train_loss 185.05825805664062
valid_loss 180.57640075683594
epochs 11/20
train_loss 184.63479614257812
valid_loss 180.16436767578125
epochs 12/20
train_loss 184.2467498779297
valid_loss 179.75796508789062
epochs 13/20
train_loss 183.83807373046875
valid_loss 179.36192321777344
epochs 14/20
train_loss 183.4651336669922
valid_loss 178.9736785888672
epochs 15/20
train_loss 183.05935668945312
valid_loss 178.59542846679688
epochs 16/20
train_loss 182.6997833251953
valid_loss 178.2197265625
epochs 17/20
train_loss 182.32150268554688
valid_loss 177.85203552246094
epochs 18/20
train_loss 181.9767608642578
valid_loss 177.49322509765625
epochs 19/20
train_loss 181.6243133544922
valid_loss 177.14208984375
epochs 20/20
train_loss 181.28668212890625
valid_loss 176.79466247558594
```

[39]:
```
import matplotlib.pyplot as plt
plt.plot(train_losses,label='train_loss')
plt.plot(valid_losses,label='valid_loss')
plt.title('RMSE Loss')
plt.legend()
```

[39]:
```
<matplotlib.legend.Legend at 0x7f2bf9862dd8>
```




[40]:
```
target_x , target_y = split_sequence(valid_set["close"],n_steps)
inputs = target_x.reshape(target_x.shape[0],target_x.shape[1],1)
```

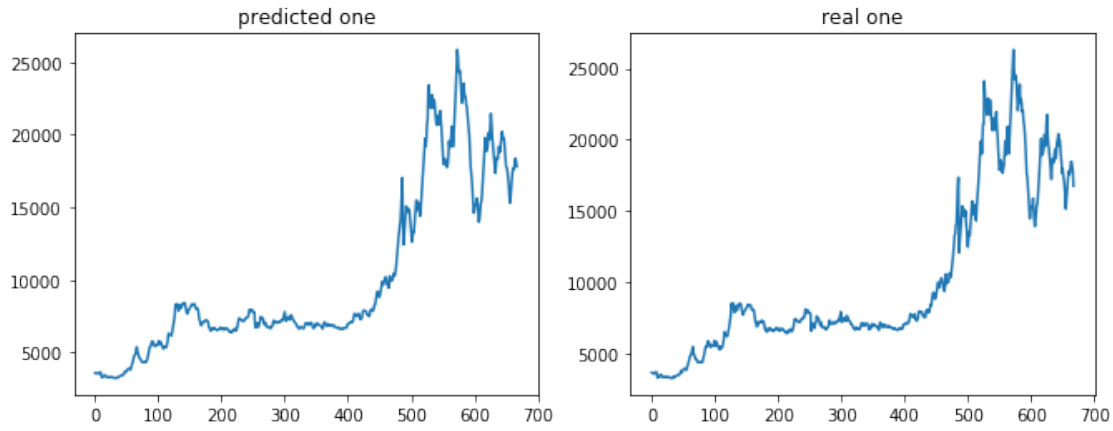
[50]:
```
target_x.shape
```

[50]: (668, 2)

[41]:
```
model.eval()
prediction = []
batch_size = 1
iterations =  inputs.shape[0]

for i in range(iterations):
    preds = model(torch.tensor(inputs[batch_size*i:batch_size*(i+1)]).float())
    prediction.append(preds.detach().numpy())
```

**Prediction Result**

[42]:
```
fig, ax = plt.subplots(1, 2,figsize=(11,4))
ax[0].set_title('predicted one')
ax[0].plot(prediction)
ax[1].set_title('real one')
ax[1].plot(target_y)
plt.show()
```




[43]:
```
plt.figure(figsize=(16,8))
plt.plot(prediction,label='Prediction Closing Price')
plt.plot(target_y,label='Real Closing Price')
plt.legend()
```

[43]: <matplotlib.legend.Legend at 0x7f2bf9461fd0>