Stock Prediction Using Convolutional Neural Network

Fateme Rahimi

December 29, 2020

To get started, we need Tehran Stock Exchange data that could be extracted using the ”pytse_client” module. In this module, all the index of Tehran Stock Exchange are available and in this report, we we will focus on the ”Golkohar” symbol.

1 Data Preparation

[1]:

```
!pip install pytse_client
```

```
Collecting pytse_client
  Downloading pytse_client-0.6.2-py3-none-any.whl (34 kB)
Requirement already satisfied: jdatetime<4.0.0,>=3.6.2 in
c:\users\user\anaconda3\lib\site-packages (from pytse_client) (3.6.2)
Collecting requests<3.0.0,>=2.23.0
  Downloading requests-2.25.1-py2.py3-none-any.whl (61 kB)
Requirement already satisfied: pandas in c:\users\user\anaconda3\lib\site-
packages (from pytse_client) (0.25.1)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\user\anaconda3\lib\site-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (2020.6.20)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\user\anaconda3\lib\site-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (1.24.2)
Requirement already satisfied: chardet<5,>=3.0.2 in
c:\users\user\anaconda3\lib\site-packages (from
requests<3.0.0,>=2.23.0->pytse_client) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\user\anaconda3\lib\site-
packages (from requests<3.0.0,>=2.23.0->pytse_client) (2.8)
Requirement already satisfied: python-dateutil>=2.6.1 in
c:\users\user\anaconda3\lib\site-packages (from pandas->pytse_client) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\user\anaconda3\lib\site-
packages (from pandas->pytse_client) (2019.3)
Requirement already satisfied: numpy>=1.13.3 in
c:\users\user\anaconda3\lib\site-packages (from pandas->pytse_client) (1.16.5)
Requirement already satisfied: six>=1.5 in c:\users\user\anaconda3\lib\site-
packages (from python-dateutil>=2.6.1->pandas->pytse_client) (1.12.0)
Installing collected packages: requests, pytse-client
  Attempting uninstall: requests
```

```
    Found existing installation: requests 2.22.0
    Uninstalling requests-2.22.0:
      Successfully uninstalled requests-2.22.0
Successfully installed pytse-client-0.6.2 requests-2.25.1
```

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

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

```
[64]:           date     open     high      low  adjClose         value    volume  \
      0     2004-08-29  12000.0  12021.0  12000.0   12000.0   18841605000   1570000
      1     2004-09-04  12600.0  12600.0  12600.0   12600.0   12600000000   1000000
      2     2004-09-05  13230.0  13230.0   7115.0   13230.0   34449329770   2708823
      3     2004-09-07  13891.0  13891.0  13891.0   13891.0   41395180000   2980000
      4     2004-09-08  14585.0  14585.0  14585.0   14585.0   20305659965   1392229
      ...          ...      ...      ...      ...       ...           ...       ...
      3320 2020-12-23  19300.0  19580.0  18650.0   19340.0  587458355110  30376324
      3321 2020-12-26  19600.0  19620.0  18600.0   18940.0  288611128190  15235035
      3322 2020-12-27  18940.0  19370.0  18180.0   18910.0  268241942070  14188123
      3323 2020-12-28  19090.0  19850.0  18800.0   19650.0  635550356530  32338277
      3324 2020-12-29  20300.0  20300.0  19200.0   19770.0  374225605950  18929701

            count    close
      0      2708  12000.0
      1       849  12600.0
      2      3887  13230.0
      3       996  13891.0
      4       409  14585.0
      ...     ...      ...
      3320   5988  19340.0
      3321   4684  18670.0
      3322   3396  19090.0
      3323   5415  19850.0
      3324  15491  20010.0

      [3325 rows x 9 columns]
```

Import Packages

```
[28]: #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))
```

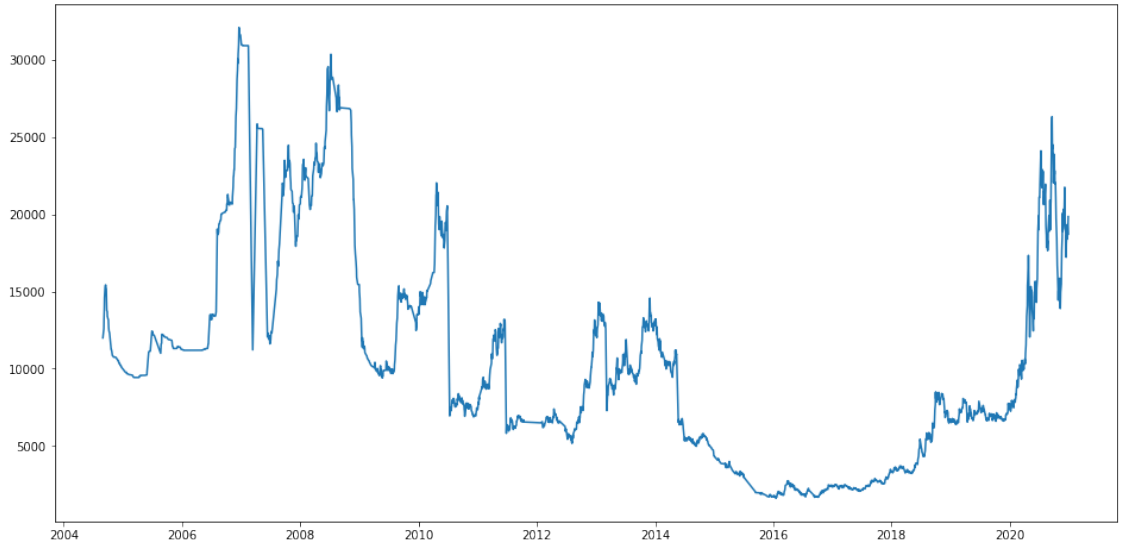
Our data covered from August 29, 2004 to December 29, 2020. Each one- minute data contains the opening price, the closing price, the highest price and the lowest price. For example, the plot of the closing price is shown in below:

[29]:

```
#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')
```

[29]:

```
[<matplotlib.lines.Line2D at 0xc81e2c8>]
```



[32]:

```
# 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]
```

[38]:

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

# 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:
(2800, 2)

 Shape of validation set:
(523, 2)
```

2 Models

We will implement a mix of machine learning algorithms to predict the future stock close price of this index, starting with simple algorithms like averaging and linear regression, and then move on to advanced techniques

2.1 Moving Average

[39]:

```
# In the next step, we will create predictions for the validation set and check␣
 ↪the RMSE using the actual values.
# making predictions
preds = []
for i in range(0,valid.shape[0]):
    a = train['close'][len(train)-523+i:].sum() + sum(preds)
    b = a/523
    preds.append(b)
```

[40]:

```
# 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:
9328.882395543873
```

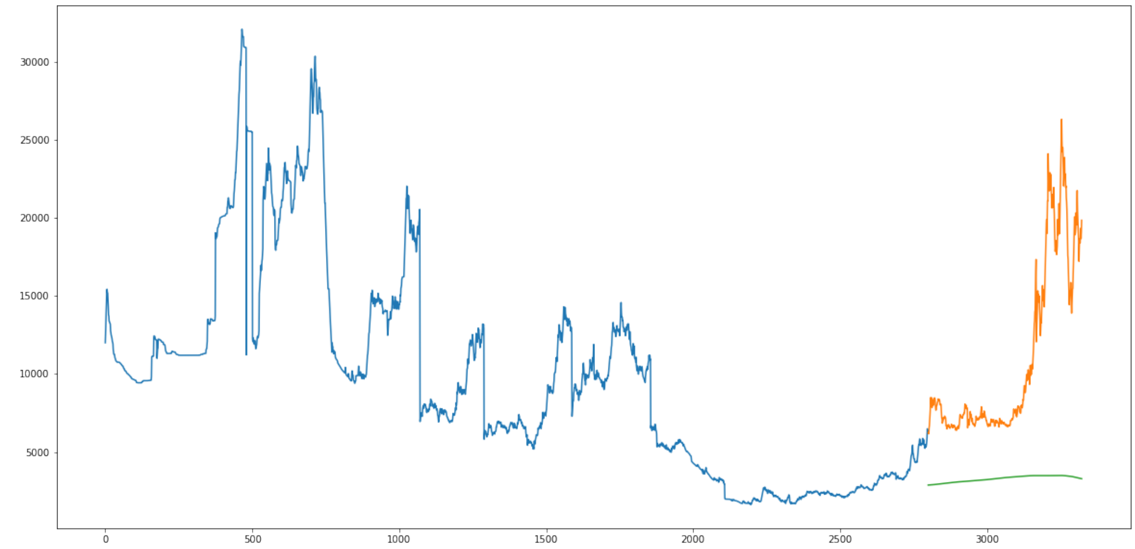
[42]:

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(train['close'])
plt.plot(valid[['close', 'Predictions']])
```

```
C:\Users\user\Anaconda3\lib\site-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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\user\Anaconda3\lib\site-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: http://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
```

[42]:

```
[<matplotlib.lines.Line2D at 0x6ae7d88>,
 <matplotlib.lines.Line2D at 0x6af2348>]
```



2.1.1 Inference

The results are not very promising (as you can gather from the plot). The predicted values are of the same range as the observed values in the train set (there is an increasing trend initially and then a slow decrease).

2.2 Linear Regression

[44]:

```
#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]
```

[47]:

```
!pip install regex
```

```
Collecting regex
  Downloading regex-2020.11.13-cp37-cp37m-win_amd64.whl (269 kB)
Installing collected packages: regex
Successfully installed regex-2020.11.13
```

```
[48]: import regex as re
      def add_datepart(df, fldname, drop=True):
          fld = df[fldname]
          if not np.issubdtype(fld.dtype, np.datetime64):
              df[fldname] = fld = pd.to_datetime(fld, infer_datetime_format=True)
          targ_pre = re.sub('[Dd]ate$', '', fldname)
          for n in ('Year', 'Month', 'Week', 'Day', 'Dayofweek', 'Dayofyear',
                  'Is_month_end', 'Is_month_start', 'Is_quarter_end',␣
        ↪'Is_quarter_start', 'Is_year_end', 'Is_year_start'):
              df[targ_pre+n] = getattr(fld.dt,n.lower())
          df[targ_pre+'Elapsed'] = fld.astype(np.int64) // 10**9
          if drop: df.drop(fldname, axis=1, inplace=True)
```

```
[50]: add_datepart(new_data,"date")
      new_data.drop('Elapsed', axis=1, inplace=True)
```

```
[51]: new_data
```

```
[51]:       close  Year  Month  Week  Day  Dayofweek  Dayofyear  Is_month_end  \
      0     12000  2004      8    35   29          6        242         False
      1     12600  2004      9    36    4          5        248         False
      2     13230  2004      9    36    5          6        249         False
      3     13891  2004      9    37    7          1        251         False
      4     14585  2004      9    37    8          2        252         False
      ...     ...   ...    ...   ...  ...        ...        ...           ...
      3318  19040  2020     12    52   22          1        357         False
      3319  19340  2020     12    52   23          2        358         False
      3320  18670  2020     12    52   26          5        361         False
      3321  19090  2020     12    52   27          6        362         False
      3322  19850  2020     12    53   28          0        363         False

            Is_month_start  Is_quarter_end  Is_quarter_start  Is_year_end  \
      0              False           False             False        False
      1              False           False             False        False
      2              False           False             False        False
      3              False           False             False        False
      4              False           False             False        False
      ...              ...             ...               ...          ...
      3318           False           False             False        False
      3319           False           False             False        False
      3320           False           False             False        False
      3321           False           False             False        False
      3322           False           False             False        False

            Is_year_start
      0             False
      1             False
```

```
2             False
3             False
4             False
...             ...
3318          False
3319          False
3320          False
3321          False
3322          False

[3323 rows x 13 columns]
```

[53]:

```python
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

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)
```

[53]:

```
LinearRegression()
```

[54]:

```python
#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
```

[54]:

```
18246.35274528345
```

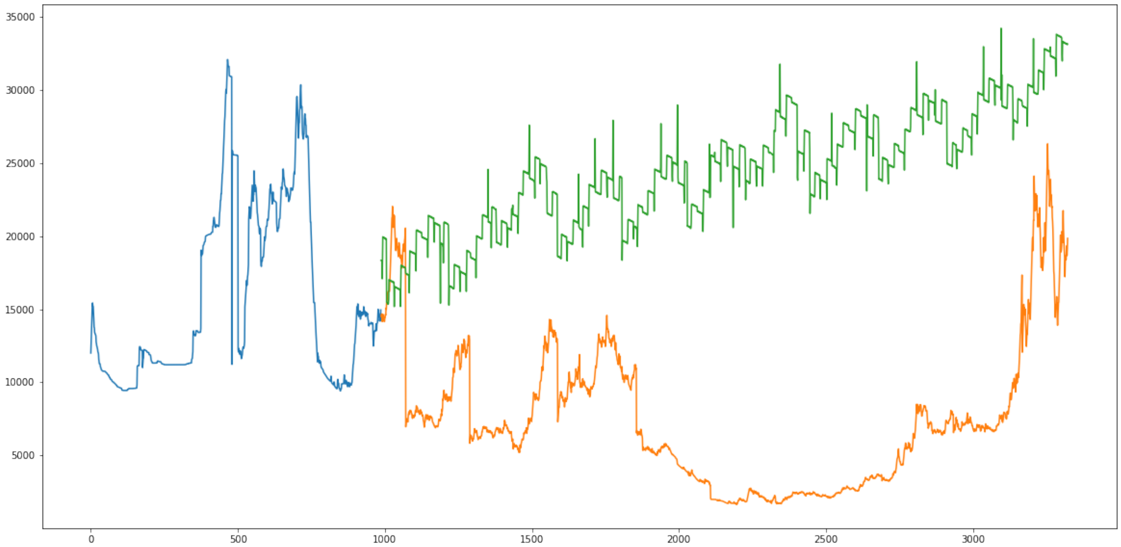
[55]:

```python
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds

valid.index = new_data[987:].index
train.index = new_data[:987].index

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

```
C:\Users\user\Anaconda3\lib\site-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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\user\Anaconda3\lib\site-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: http://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
```

```
[55]: [<matplotlib.lines.Line2D at 0xcace248>,
       <matplotlib.lines.Line2D at 0xcace788>]
```



2.2.1 Inference

The RMSE value is higher than the previous technique, which clearly shows that linear regression has performed poorly,

2.3 KNN

[56]:

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

[57]:

```
#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)
```

[58]:

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

[58]:

```
11196.773347490891
```

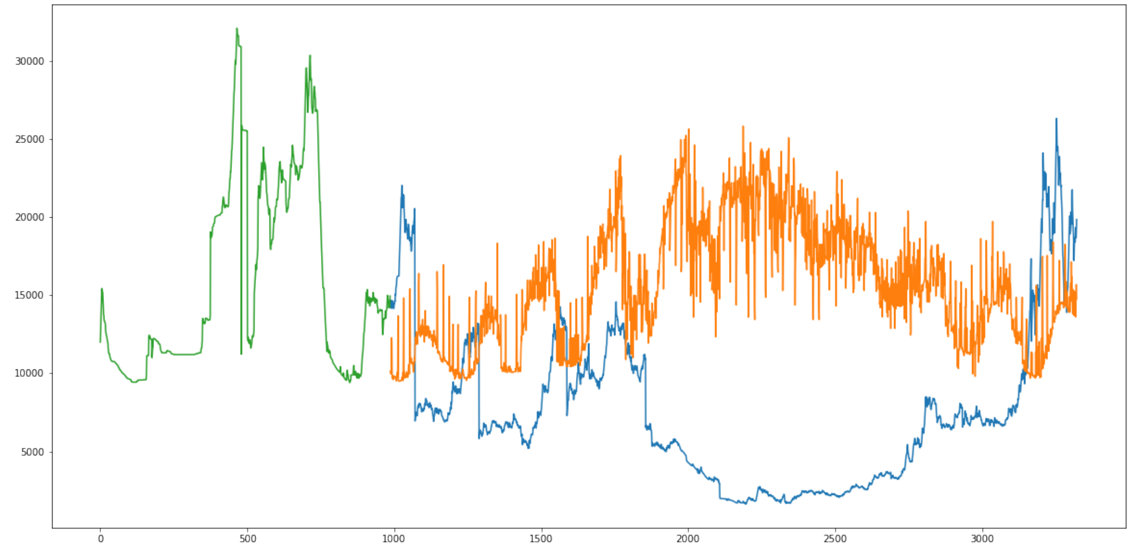
[59]:

```
#plot
valid['Predictions'] = 0
valid['Predictions'] = preds
plt.plot(valid[['close', 'Predictions']])
plt.plot(train['close'])
```

```
C:\Users\user\Anaconda3\lib\site-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: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

C:\Users\user\Anaconda3\lib\site-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: http://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
```

```
[59]: [<matplotlib.lines.Line2D at 0xcde1fc8>]
```



2.3.1 Inference

The RMSE value is almost similar to the linear regression model and the plot shows the same pattern.

3 Future work

In the next, we want to make this prediction using deep learning methods and compare the result with the results of these methods.