```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,LSTM
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import math
from sklearn.metrics import mean_squared_error
from kerastuner.engine.hyperparameters import HyperParameters
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:10: DeprecationWarning: `import kerastuner` is deprecated, please use
  # Remove the CWD from sys.path while we load stuff.
```

```
df_test = pd.read_csv('../input/gooogle-stock-price/Google_Stock_Price_Test.csv')
df_train = pd.read_csv('../input/gooogle-stock-price/Google_Stock_Price_Train.csv')
df = pd.concat([df_test,df_train])
```

```
df.head()
```

	Date	Open	High	Low	Close	Volume
0	1/3/2017	778.81	789.63	775.80	786.14	1,657,300
1	1/4/2017	788.36	791.34	783.16	786.9	1,073,000
2	1/5/2017	786.08	794.48	785.02	794.02	1,335,200
3	1/6/2017	795.26	807.90	792.20	806.15	1,640,200
4	1/9/2017	806.40	809.97	802.83	806.65	1,272,400

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1278 entries, 0 to 1257
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    1278 non-null   object
 1   Open    1278 non-null   float64
 2   High    1278 non-null   float64
 3   Low     1278 non-null   float64
 4   Close   1278 non-null   object
 5   Volume  1278 non-null   object
dtypes: float64(3), object(3)
memory usage: 69.9+ KB
```

```
df.describe()
```

	Open	High	Low
count	1278.000000	1278.000000	1278.000000
mean	537.994906	542.168897	533.278803
std	154.508365	155.582501	153.174011
min	279.120000	281.210000	277.220000
25%	406.037500	408.230000	403.335000
50%	538.395000	542.330000	534.355000
75%	668.862500	677.705000	662.190000
max	837.810000	841.950000	827.010000

```
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume'], dtype='object')
```

```
df.shape
```

```
(1278, 6)
```

```
df.isnull().sum()
```

```
     Date      0
     Open      0
     High      0
     Low       0
     Close     0
     Volume    0
     dtype: int64
```

```
df.dtypes
```

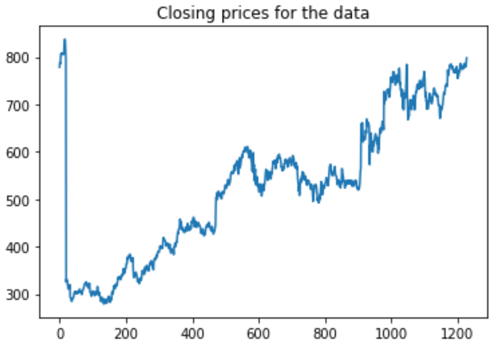
```
     Date       object
     Open      float64
     High      float64
     Low       float64
     Close      object
     Volume     object
     dtype: object
```

```
df    = df.loc[:,["Open"]].values
train = df[:len(df)-50]
test = df[len(train):]
# reshape
train = train.reshape(train.shape[0],1)
```

```
train.shape
```

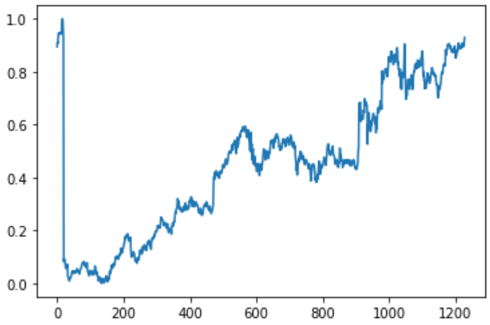
```
     (1228, 1)
```

```
plt.plot(train);
plt.title("Closing prices for the data");
```



```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range= (0,1)) # defining of Scaler
train_scaled = scaler.fit_transform(train) # applying to Scaler to train

plt.plot(train_scaled)
plt.show()
```



```
# We add first 50 location to "X_train" and we 51. location to "y_train" .
X_train = []
y_train = []
timesteps = 50

for i in range(timesteps, train_scaled.shape[0]):
    X_train.append(train_scaled[i-timesteps:i,0])
    y_train.append(train_scaled[i,0])
```

```
X_train, y_train = np.array(X_train), np.array(y_train)


# Reshaping
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)  # Dimension of array is 3.


# --- RNN ---

# Importing the Keras libraries and packages

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout # it block to overfitting

# Initialising the RNN
regressor = Sequential()

# Adding the first RNN layer and some Dropout regularisation
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True, input_shape = (X_train.shape[1], 1)))
regressor.add(Dropout(0.2))

# Adding a second RNN layer and some Dropout regularisation.
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third RNN layer and some Dropout regularisation.
regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth RNN layer and some Dropout regularisation.
regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))


# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

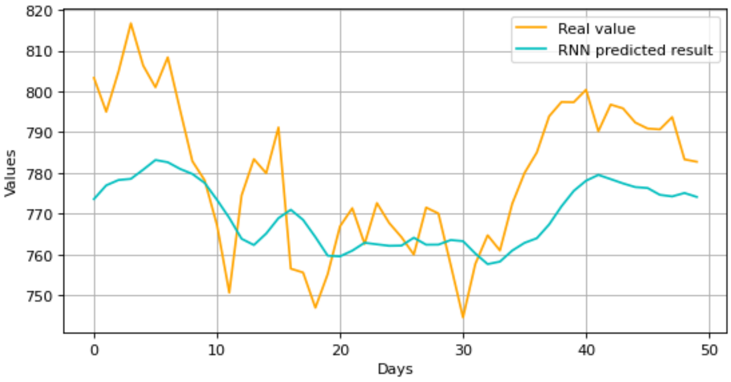
```
Epoch 94/100
37/37 [==============================] - 4s 111ms/step - loss: 0.0021
Epoch 95/100
37/37 [==============================] - 4s 112ms/step - loss: 0.0022
Epoch 96/100
37/37 [==============================] - 4s 118ms/step - loss: 0.0020
Epoch 97/100
37/37 [==============================] - 5s 133ms/step - loss: 0.0019
Epoch 98/100
37/37 [==============================] - 4s 117ms/step - loss: 0.0021
Epoch 99/100
37/37 [==============================] - 4s 116ms/step - loss: 0.0020
Epoch 100/100
37/37 [==============================] - 4s 115ms/step - loss: 0.0021
<keras.callbacks.History at 0x7fd7ae0f1a10>
```

```
inputs = df[len(df) - len(test) - timesteps:]
inputs = scaler.transform(inputs)  # min max scaler
```

```
X_test = []
for i in range(timesteps, inputs.shape[0]):
    X_test.append(inputs[i-timesteps:i, 0]) # 0 dan 50 ye, 1 den 51 e gibi kaydirarark 50 eleman aliyoruz
X_test = np.array(X_test)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

```
predicted_data = regressor.predict(X_test)
predicted_data = scaler.inverse_transform(predicted_data)
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(test,color="orange",label="Real value")
plt.plot(predicted_data,color="c",label="RNN predicted result")
plt.legend()
plt.xlabel("Days")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



LSTM Modules

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
```

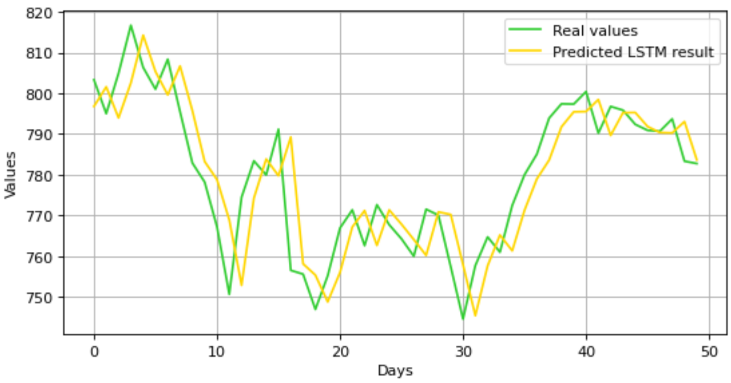
```
model = Sequential()
model.add(LSTM(10, input_shape=(None,1)))
model.add(Dense(1))
model.compile(loss="mean_squared_error",optimizer='Adam')
model.fit(X_train,y_train,epochs=50, batch_size=1)
```

```
Epoch 1/50
1178/1178 [==============================] - 11s 8ms/step - loss: 0.0084
Epoch 2/50
1178/1178 [==============================] - 9s 8ms/step - loss: 0.0012
Epoch 3/50
1178/1178 [==============================] - 9s 8ms/step - loss: 9.9040e-04
Epoch 4/50
1178/1178 [==============================] - 9s 8ms/step - loss: 8.5377e-04
Epoch 5/50
1178/1178 [==============================] - 10s 8ms/step - loss: 7.2978e-04
Epoch 6/50
```

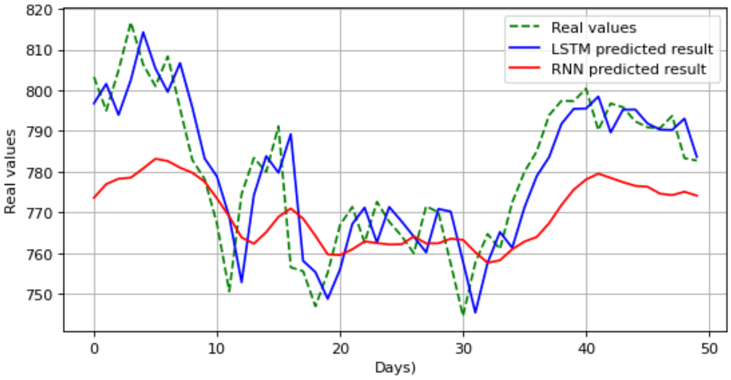
```
1178/1178 [==============================] - 9s 8ms/step - loss: 6.6802e-04
Epoch 7/50
1178/1178 [==============================] - 9s 8ms/step - loss: 6.2765e-04
Epoch 8/50
1178/1178 [==============================] - 9s 8ms/step - loss: 5.4094e-04
Epoch 9/50
1178/1178 [==============================] - 9s 8ms/step - loss: 4.4366e-04
Epoch 10/50
1178/1178 [==============================] - 9s 8ms/step - loss: 4.1634e-04
Epoch 11/50
1178/1178 [==============================] - 10s 8ms/step - loss: 3.7382e-04
Epoch 12/50
1178/1178 [==============================] - 9s 8ms/step - loss: 3.2931e-04
Epoch 13/50
1178/1178 [==============================] - 9s 8ms/step - loss: 3.3514e-04
Epoch 14/50
1178/1178 [==============================] - 9s 8ms/step - loss: 3.1120e-04
Epoch 15/50
1178/1178 [==============================] - 10s 8ms/step - loss: 2.9448e-04
Epoch 16/50
1178/1178 [==============================] - 9s 8ms/step - loss: 3.0684e-04
Epoch 17/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.9451e-04
Epoch 18/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.9575e-04
Epoch 19/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.9117e-04
Epoch 20/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.9946e-04
Epoch 21/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.8828e-04
Epoch 22/50
1178/1178 [==============================] - 10s 8ms/step - loss: 2.7637e-04
Epoch 23/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.7797e-04
Epoch 24/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.8064e-04
Epoch 25/50
1178/1178 [==============================] - 10s 8ms/step - loss: 2.8097e-04
Epoch 26/50
1178/1178 [==============================] - 10s 8ms/step - loss: 2.8477e-04
Epoch 27/50
1178/1178 [==============================] - 9s 8ms/step - loss: 3.0241e-04
Epoch 28/50
1178/1178 [==============================] - 9s 8ms/step - loss: 2.7349e-04
Epoch 29/50
1178/1178 [==============================] - 10s 8ms/step - loss: 2.8053e-04
```

```
predicted_data2=model.predict(X_test)
predicted_data2=scaler.inverse_transform(predicted_data2)
```

```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(test,color="LimeGreen",label="Real values")
plt.plot(predicted_data2,color="Gold",label="Predicted LSTM result")
plt.legend()
plt.xlabel("Days")
plt.ylabel("Values")
plt.grid(True)
plt.show()
```



```
plt.figure(figsize=(8,4), dpi=80, facecolor='w', edgecolor='k')
plt.plot(test,color="green", linestyle='dashed',label="Real values")
plt.plot(predicted_data2,color="blue", label="LSTM predicted result")
plt.plot(predicted_data,color="red",label="RNN predicted result") # ben ekledim
plt.legend()
plt.xlabel("Days)")
plt.ylabel("Real values")
plt.grid(True)
plt.show()
```


Real values
LSTM predicted result
RNN predicted result
Real values
Days)