Laporan Akhir Project Robotika dan Kecerdasan Buatan

Disusun Oleh :

Nama	: Xena Mutiara S	(4211701015)
	: Patuan Sitorus	(4211701017)
	: M. Rayfigo P	(4211701026)
	: M. Dwi Heryanto	(4211701027)

POLITEKNIK NEGERI BATAM

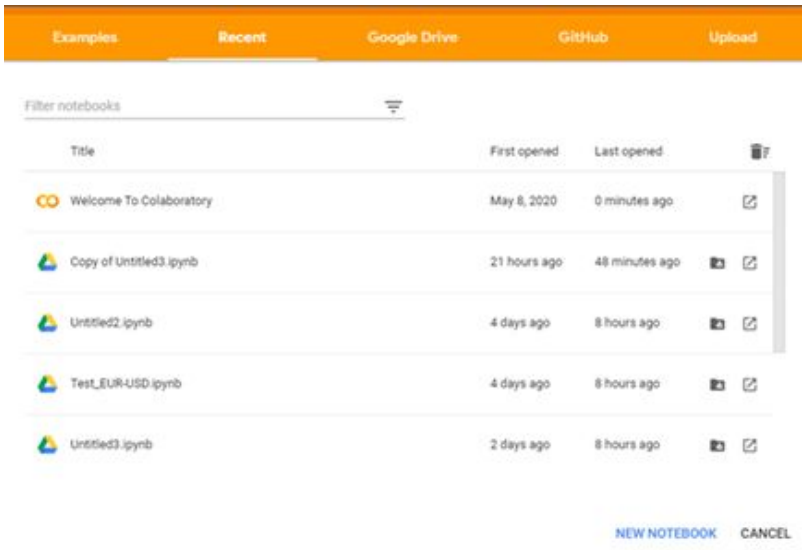
BATAM

2020

Seorang trader harus mampu memprediksi baik itu harga, saham dan lainnya. Untuk itu dibutuhkan sebuah *learning machine* yang dapat memprediksi harga atau saham atau yang lainnya dari data sebelumnya yang telah didapat. Pada projek kali ini dibutuhkan sebuah *learning machine* untuk dapat memprediksi nilai tukar EUR to USD. Untuk membuat *learning machine* tersebut digunakan sebuah situs yaitu Google Colab. Google Colab merupakan sebuah tools yang berbasis cloud dan free untuk tujuan penelitian. Google colab dibuat dengan environment jupyter dan mendukung library yang dibutuhkan dalam pengembangan Artificial Intelligence. Untuk melakukan proses learning dibutuhkan yang beberapa data, pada program ini digunakan sebuah data day-train.csv dimana data tersebut didapat dari history EURUSD pada metatrader 4, yang isinya terdapat beberapa data diantaranya kolom Date, Time, Open, Low, Close, dan Volume. Data inilah yang akan dilakukan proses learning untuk dapat memprediksi data berikutnya.

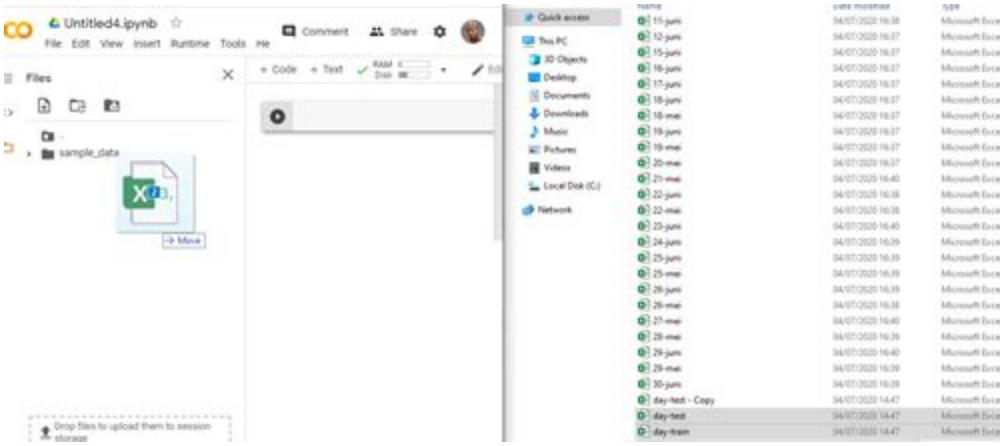
Untuk membuat program pada Google Colab dapat dilakukan sebagai berikut :

1. Membuka situs web colab.research.google.com di google chrome atau sejenisnya.
2. Kemudian akan tampil seperti gambar dibawah ini.



Jika terlihat seperti gambar diatas artinya sudah pernah membuat program menggunakan google colab, selanjutnya tinggal klik kiri pada nama programnya. Jika ingin membuat program baru dapat langsung mengklik new notebook pada bagian bawah.

3. Seperti yang dijelaskan sebelumnya *learning machine* membutuhkan data sebelumnya untuk dapat memprediksi data selanjutnya, untuk itu diperlukan untuk mengunggah data yang akan dilakukan *learning* pada google colab dengan cara mengklik pada bagian file kemudian seret data yang akan dilakukan learning ke dalam zona file seperti gambar berikut.



4. Setelah data diunggah, selanjutnya dapat dilakukan pembuatan program.

Penjelasan program:

a. Proses memasukkan library yang akan digunakan untuk prediksi.

```
#Import libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

b. Pada baris awal berfungsi untuk memberikan informasi keberadaan file day-train.csv, kemudian baris selanjutnya adalah pembacaan data menggunakan library pandas kemudian menampilkan bagian akhir dari data. Program terakhir digunakan untuk menampilkan data terakhir dari file csv, digunakan untuk mengecek apakah sudah sesuai dengan aslinya atau tidak.

```
[ ]
dataset_train_path = os.getcwd() + "/day-train.csv"
dataset_train = pd.read_csv(dataset_train_path)
dataset_train.tail()
```

	Date	Time	Open	High	Low	Close	Volume
2591	2019.12.24	0:00	1.10895	1.10939	1.10690	1.10851	22030
2592	2019.12.26	0:00	1.10900	1.11088	1.10820	1.10974	10530
2593	2019.12.27	0:00	1.10973	1.11882	1.10940	1.11760	36676
2594	2019.12.30	0:00	1.11754	1.12206	1.11713	1.11985	36372
2595	2019.12.31	0:00	1.11985	1.12392	1.11976	1.12154	32207

c. Berikut merupakan pemisahan kolom dimana kolom yang diambil atau digunakan adalah kolom ke-5 (close), kemudian dilakukan print untuk data pada kolom close dan menampilkan

banyak data dari kolom tersebut. Data pada kolom close dimasukkan ke dalam variabel training_set.

```
[ ] training_set = dataset_train.iloc[:,5:6].values

    print(training_set)
    print("********************")
    print(training_set.shape)
```

```
[[1.44111]
 [1.43624]
 [1.43998]
 ...
 [1.1176 ]
 [1.11985]
 [1.12154]]
********************
(2596, 1)
```

d. Pada bagian ini menampilkan tipe data dari kolom close.

```
#-----------------------------------------------#
# Additional Information (Things to Remember!) #
#-----------------------------------------------#

print(type(dataset_train))
print(type(dataset_train.iloc[:,5:6]))
print(type(dataset_train.iloc[:,5:6].values))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'numpy.ndarray'>
```

e. Pada proses ini digunakan untuk preprocessing data training_set, dimana library yang digunakan adalah MinMaxScaler dengan range 0 untuk minimum dan 1 untuk maksimum. Range ini mengartikan bahwa data yang terdapat pada train_set akan diubah dengan skala 0 hingga 1 dan ditampung pada variabel scaled_training_set.

```
[5] from sklearn.preprocessing import MinMaxScaler

    scaler = MinMaxScaler(feature_range = (0,1))
    scaled_training_set = scaler.fit_transform(training_set)

    # scaled_training_set
```

f. Pada baris pertama dan kedua adalah pendeklarasian variabel x_train dan y_train dimana nilai pada variabel tersebut masih kosong. Kemudian baris berikutnya melakukan perulangan dimulai dari 60 hingga banyak data pada training_set, lalu x_train diisi dengan 60 baris pertama, Perintah [i-60:i] artinya baris ke [0:60], karena i dimulai dari 60. Kemudian diikuti

dengan kolom ke 0 pada training_set_scaled (kolom close). Dan pada baris akhir x_train dan y_train diubah kedalam bentuk array dengan perintah np.array, hal ini dikarenakan RNN hanya dapat menerima inputan berupa array.

```
[6]  X_train = []
     y_train = []
     for i in range(60,len(training_set)):
         X_train.append(scaled_training_set[i-60:i, 0])
         y_train.append(scaled_training_set[i, 0])
     X_train = np.array(X_train)
     y_train = np.array(y_train)
```

g. Pada bagian ini menampilkan banyak data pada x_train dan y_train dimana x_train, dari gambar dibawah dapat dilihat bahwa x_train adalah variabel yang memiliki 2 elemen.

```
[7]  print(X_train.shape)
     print(y_train.shape)
```

```
(2536, 60)
(2536,)
```

h. Untuk melakukan RNN dilakukan proses yang disebut LSTM dimana input dari LSTM adalah berupa array 3 dimensi untuk itu data x_train yang awalnya array 2 dimensi harus diubah ke dalam bentuk 3 dimensi dengan perintah x_train.

```
[8]  X_train = np.reshape(X_train,(X_train.shape[0], X_train.shape[1], 1))

     X_train.shape
```

```
(2536, 60, 1)
```

i. Berikut ini adalah proses memasukkan library untuk proses LSTM

```
[11] from keras.models import Sequential
     from keras.layers import LSTM
     from keras.layers import Dense
     from keras.layers import Dropout
```

```
Using TensorFlow backend.
```

j. Saat memulai pembuatan layer digunakan perintah Sequential untuk menginisialisasi. Untuk menambahkan sebuah layer pada RNN diberikan perintah add, kemudian terdapat beberapa parameter diantaranya adalah units (jumlah neuron), return_sequence adalah perintah yang digunakan untuk menentukan ada atau tidaknya layer selanjutnya (defaultnya adalah false, dan true diberikan jika terdapat layer tambahan), kemudian yang ketiga adalah input_shape dimana x_train yang sebelumnya 2 dimensi telah diubah menjadi 3 dimensi sehingga dapat digunakan pada input_shape. Selanjutnya menambahkan layer dropout dengan nilai 0.2. Penentuan angka 0.2 adalah yang biasa dipakai oleh pengguna LSTM. Dropout bertujuan untuk menghindari overfitting.

   Kemudian masuk ke layer selanjutnya dengan perintah yang sama yaitu add, pada layer selanjutnya hanya menggunakan 2 buah parameter yaitu units dan return_sequences, untuk input_shape tidak digunakan karena telah diberikan pada layer 1.

   Perintah tersebut berulang hingga layer kedua dari akhir, pada layer ini hanya diberi satu parameter yaitu units, dan parameter return_sequences tidak diberikan karena default dari nilai return_sequence adalah false yang artinya tidak ada layer tambahan lagi.

   Untuk layer terakhir digunakan untuk mendefinisikan output layer, dan hanya memerlukan 1 unit untuk forecasting

```
[ ] #----------------------------------------#
    #  Initialization and Adding layers to RNN   #
    #----------------------------------------#
    regressor = Sequential()

    regressor.add(LSTM(units = 100, return_sequences= True, input_shape = (X_train.shape[1], 1)))   # the first LSTM layer
    regressor.add(Dropout(0.2))

    regressor.add(LSTM(units = 100, return_sequences= True))  # the second LSTM layer
    regressor.add(Dropout(0.2))

    regressor.add(LSTM(units = 100, return_sequences= True))  # the third LSTM layer
    regressor.add(Dropout(0.2))

    regressor.add(LSTM(units = 100, return_sequences= True))  # the fourth LSTM layer
    regressor.add(Dropout(0.2))

    regressor.add(LSTM(units = 100))  # the fifth LSTM layer
    regressor.add(Dropout(0.2))

    regressor.add(Dense(units=1))   # the dense layer
```

k. Pada baris pertama adalah persiapan sebelum melakukan learning dengan layer yang telah dibuat sebelumnya dengan parameter optimizer menggunakan adam dan kriteria loss adalah mean squared error

   Pada baris kedua mulai proses learning dengan menentukan jumlah epoch, jumlah epoch ini bebas. Epoch dan iterasi berbeda. Perbedaan terdapat dalam prosesnya. Untuk iterasi

contohnya terdapat 2 iterasi maka prosesnya adalah A-B-C-D-A-B-C-D, sementara untuk 2 epoch melakukan proses seperti berikut A-B-C-D-C-B-A-B-C-D-C-B-A, atau dapat dikatakan epoch adalah iterasi dengan perambatan balik (dapat maju dan mundur).

```
[ ]  #-------------------------------------------------------#
     #  Compiling and Fitting the RNN to the Training set  #
     #-------------------------------------------------------#

     regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
     regressor.fit(X_train, y_train, epochs=100, batch_size=32)
```

l. Memasukkan data yang digunakan untuk prediksi

Dalam hal ini program dimaksudkan untuk membaca file csv bernama 'EURUSD1440(18Mei).csv' dengan menggunakan library pandas. Lalu data pembacaan file tersebut dimasukkan ke dalam variabel predict_next_day.

```
[ ]  #Read Data from CSV file (For Prediction 20 Mei 2020)
     predict_next_day = pd.read_csv('02-juli.csv', parse_dates=True, index_col=0)
```

- Baris ke-3 digunakan untuk memfilter bagian dari kolom CLOSE
- Baris ke 5 digunakan untuk mengambil data dari 60 data terakhir
- Baris ke 7 digunakan untuk memberi skala antara 0 hingga 1 untuk data pada baris ke 5
- Baris ke 9 dibuat sebuah array baru
- Baris ke 11 array baru pada baris ke 9 akan diisi dengan data yang telah di skala pada baris ke 7
- Baris ke 13 data array pada baris 11 dijadikan numpy
- Baris ke 15 data numpy array yang awalnya berbentuk 2 dimensi diubah menjadi 3 dimensi, dikarenakan proses LSTM membutuhkan input array 3 dimensi
- Baris ke 17 berfungsi untuk menggunakan model *training* untuk memprediksi nilai selanjutnya dari data
- Baris ke 19 berfungsi untuk membalikkan nilai dikarenakan saat dimasukkan ke model *training* data tersebut berada dalam skala 0-1.

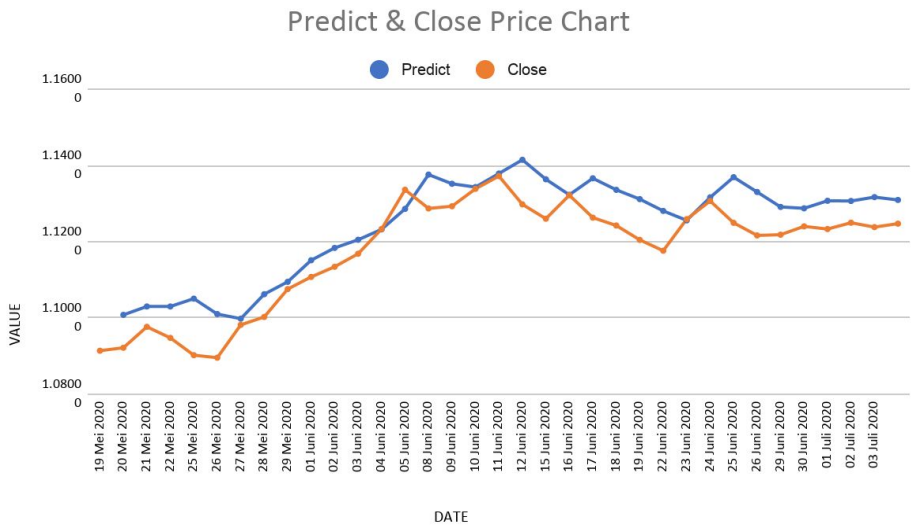
Dan pada baris akhir digunakan untuk menampilkan data hasil prediksi atau data selanjutnya dari data yang telah didapat.

```
#create a new dataframe
#iloc[:,4:5] is to select all range in column 4-5 that is close column
new_df = predict_next_day.iloc[:,4:5]
#get the last 60 day closing price values and convert the dataframe to an array
last_60_days = new_df[-60:].values
#scale the data to be values beetween 0 and 1
last_60_days_scaled = scaler.transform(last_60_days)
#create an empty list
Xx_test = []
#append the past 60 days
Xx_test.append(last_60_days_scaled)
#convert the X_test data set to a numpy array
Xx_test = np.array(Xx_test)
#Reshape the data
Xx_test = np.reshape(Xx_test, (Xx_test.shape[0], Xx_test.shape[1], 1))
#Get the predicted scaled price
pred_price = regressor.predict(Xx_test)
#undo the scaling
pred_price = scaler.inverse_transform(pred_price)

print(pred_price)
# pred_price
```

m. Mengumpulkan data hasil prediksi dan perintah mengambil posisi

Setelah melakukan prediksi selama beberapa hari, didapat data prediksi setiap hari yang dikumpulkan ke dalam excel dan dibuat grafiknya. Data tersebut digunakan untuk memberi perintah kepada metatrader untuk mengambil posisi beli atau jual. Namun, untuk saat ini belum dapat diterapkan mengenai perintah mengambil posisi pada metatrader, dan algoritma yang digunakan hanya untuk mengambil posisi beli saja dengan menggunakan perhitungan jika harga prediksi lebih besar dibanding harga saat ini, maka perintah akan "BUY".



Kesimpulan

Dalam melakukan trading trader harus mampu dengan tepat menentukan waktu untuk buy dan waktu untuk sell, untuk itu trader harus mampu memprediksi data selanjutnya agar tidak mengalami kerugian. Untuk membantu trader dalam memprediksi digunakan sebuah learning machine yang memanfaatkan data sebelumnya untuk dapat memprediksi data selanjutnya. Dengan adanya learning machine tersebut trader dapat mengetahui dengan mudah kapan waktunya untuk buy atau sell.

Dari data yang telah didapat, dapat disimpulkan bahwa dengan adanya prediksi menggunakan machine learning dapat menjadi salah satu acuan dalam trading. Namun hal itu hanya sebatas acuan saja, tidak dapat diandalkan 100% karena hanya menggunakan data input dari close price saja.

Open the link to see the code: https://github.com/dwihp2/Forex-Prediction-ML