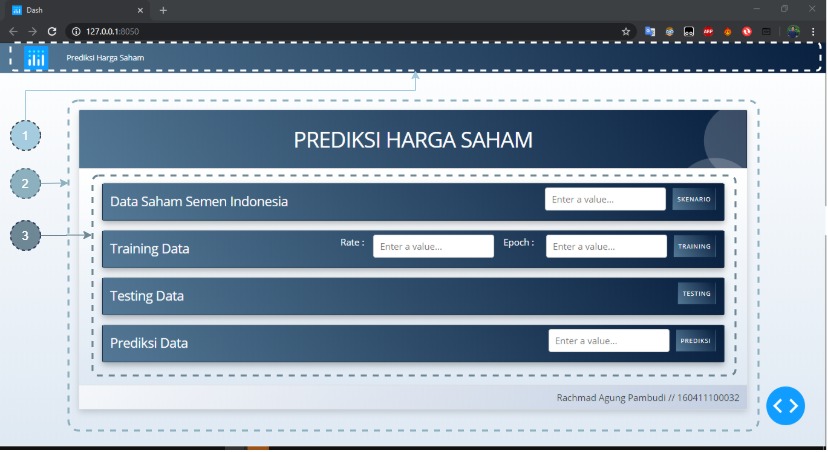
# 1. Tampilan Aplikasi

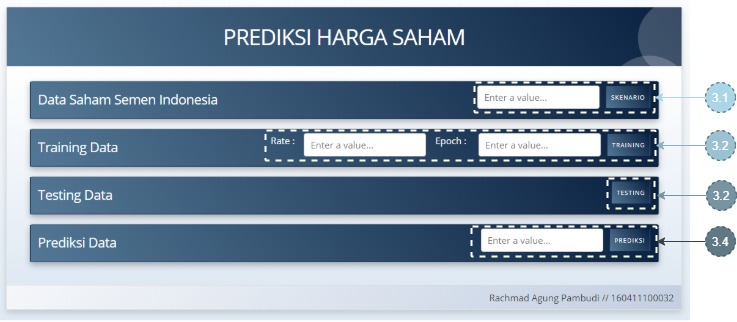
Aplikasi dibuat dengan menggunakan Bahasa pemrograman python 3.7 dibantu dengan library (pandas, numpy, plotly, dan dash).

Gambar 1. Tampilan Awal Aplikasi

Dari *Gambar 1*. tampilan awal aplikasi ini diberikan 3 bagian yaitu:

1. Navbar atau judul dari aplikasi.
2. Pengelompokan fitur-fitur yang diberikan oleh aplikasi.
3. *Field* dantomboluntuk menjalankan fitur – fitur didalam aplikasi.

# 2. Tampilan Menu



Gambar 2 Tampilan Menu fitur – fitur

Dari *Gambar 2*. Tampilan Menu fitur – fitur ini dibagi menjadi 4 sesuai dengan fungsinya yaitu:

1. *Poin 3.1* diberikan 1 *Field* dan 1tombol yang berfungsi untuk menyimpan variabel skenario berapa yang akan dieksekusi, dan akan menampilkan data tabel dan grafik harga saham.
2. *Poin 3.2* diberikan 2 *Field* dan 1tombol yang berfungsi untuk menyimpan variabel learning rate dan epoch berapa yang akan digunakan, dan akan menampilkan data tabel dan grafik hasil training.
3. *Poin 3.3* diberikan 1tombol yang berfungsi untuk mengeksekusi data testing dan akan menampilkan data testing tabel dan grafik hasil testing.
4. *Poin 3.1* diberikan 1 *Field* dan 1tombol yang berfungsi untuk menyimpan variabel berapa hari yang akan diprediksi, dan akan menampilkan data tabel dan grafik hasil prediksi.

# 3. Tampilan fitur

## 3.1. Tampilan tabel dan grafik full data close saham

Dalam tampilan ini terdapat juga 1 *Field* dan 1tombol yang berfungsi untuk menyimpan variabel skenario, ada 3 skenario yang dapat dijalankan yaitu skenario pertama 250 data training dan 250 data testing, skenario kedua 500 data training dan 500 data testing, dan skenario ketiga 1000 data training dan 500 data testing.



Gambar 3. Tampilan tabel dan grafik full data close saham

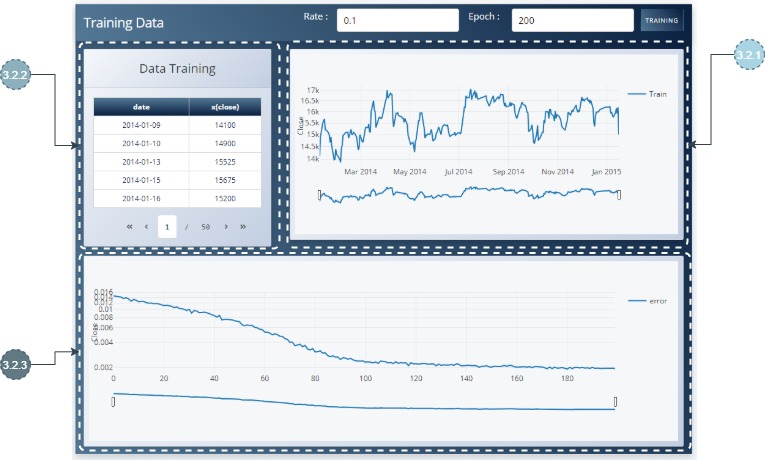
Dari *Gambar 3*. Tampilan tabel dan grafik full data close saham dibagi menjadi 2 yaitu:

1. *Poin 3.1.1* diberikan tampilan tabel data yang berfungsi untuk memperlihatkan data saham seluruhnya.
2. *Poin 3.1.2* diberikan tampilan grafik data yang berfungsi untuk memperlihatkan grafik data saham seluruhnya.

## 3.2 Tampilan Training data

Dalam tampilan ini juga terdapat 2 *Field* dan 1tombol yang berfungsi untuk menyimpan variabel epoch dan learning rate, dan jika ditekan tombonya akan mengarah ke proses fungsi training.

Pada tampilan ini menjalankan fungsi *training* dalam proses *backend* nya menjalan dari proses *forward, backward, update* bobot, dan sampai optimasi.proses training ini akan menghasilkan bobot atau model baru yang sudah di *training* dan disimpan dalam file berformat csv, yang nantinya akan digunakan untuk proses testing maupun proses prediksi.



Gambar 4. Tampilan Training data

Dari *Gambar 4*. Tampilan Training data dibagi menjadi 3 konten sesuai dengan fungsinya yaitu:

1. *Poin 3.2.1* diberikan tampilan tabel data yang berfungsi untuk memperlihatkan data training saja.
2. *Poin 3.2.2* diberikan tampilan grafik data yang berfungsi untuk memperlihatkan grafik data training.
3. *Poin 3.2.3* diberikan tampilan grafik hasil proses training yang menunjukan tingkat errornya.

## 3.2 Tampilan Testing data



Gambar 5 . Tampilan Testing Data

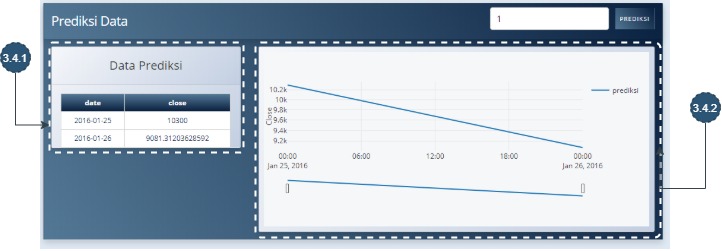
Dari *Gambar 5*. Tampilan Testing data dibagi menjadi 4 konten sesuai dengan fungsinya yaitu:

1. *Poin 3.3.1* diberikan tampilan tabel data yang berfungsi untuk memperlihatkan data *testing* saja.
2. *Poin 3.3.2* diberikan tampilan grafik data yang berfungsi untuk memperlihatkan grafik data *testing*.
3. *Poin 3.3.3* diberikan tampilan grafik hasil proses testing yang menunjukan perbandingan hasil prediksi dengan nilai aktualnya, dan diperlihatkan juga hasil dari MAPE, Akurasi, MSE.
4. *Poin 3.3.4* diberikan tampilan tabel data yang berfungsi untuk memperlihatkan data hasil *testing*.

Pada saat tombol ditekan akan menunjukkan tampilan seperti gambar 5 yang akan menjalankan fungsi *testing* dalam proses *backend* nya memproses *forward saja*. proses testing ini akan menghitung juga hasil akurasi, MAPE, MSE, yang didapat dan tabel hasil prediksi dan nilai aktualnya.

## 3.2 Tampilan Prediksi data

Pada saat tombol ditekan akan menunjukkan tampilan seperti gambar 5 yang akan menjalankan fungsi prediskidalam proses *backend* nya memproses *forward saja*.



Gambar 6. Tampilan tabel dan grafik prediksi data close saham

Dari *Gambar 6*. Tampilan tabel dan grafik full data close saham dibagi menjadi 2 yaitu:

1. *Poin 3.4.1* diberikan tampilan tabel data yang berfungsi untuk memperlihatkan data prediksi.
2. *Poin 3.4.2* diberikan tampilan grafik data yang berfungsi untuk memperlihatkan grafik prediksi naik atau turun harganya.

4. Source code Objek LSTM

|  |
| --- |
| class LSTMCell:      def \_\_init\_\_(self, inputSize, numCells):          self.inputSize = inputSize          self.numCells = numCell          self.W = [[-0.245714286 ,0.850360602    ,0.029262045    ,0.184398087]                  ,[0.868020398   ,0.860429754    ,-0.379580925   ,0.079506914]                  ,[-0.206444161  ,-0.24856166    ,-0.085253247   ,0.25112624 ]                  ,[0.842874383   ,-0.324206065   ,0.907722829    ,-0.593738792]]          W = pd.DataFrame(self.W)          W.to\_csv("P\_W.csv",header=False,index=False)          self.h = []          self.C = []          self.C\_bar = []          self.i = []          self.f = []          self.o = []          self.I = []          self.z = []      def forwardStep(self, x, jenis):          if jenis =="prediksi":              models = pd.read\_csv("model.csv")              model = models.values          else:              model = self.W          I = np.concatenate((x, self.h[-1]))          self.I.append(I)          z = np.dot(model, I)          self.z.append(z)          C\_bar = np.tanh(z[0:self.numCells])          self.C\_bar.append(C\_bar)          i = sigmoid(z[self.numCells:self.numCells \* 2])          self.i.append(i)          f = sigmoid(z[self.numCells \* 2:self.numCells \* 3])          self.f.append(f)          o = sigmoid(z[self.numCells \* 3:])          self.o.append(o)          C = np.multiply(f, self.C[-1]) + np.multiply(i, C\_bar)          self.C.append(C)          h = np.multiply(o, np.tanh(C))          self.h.append(h)          return (h,C,o,f,i,C\_bar,z,I,model)      def forwardPass(self, x,jenis):          numCells = self.numCells          self.h.append(np.zeros(numCells))          self.C.append(np.zeros(numCells))          self.C\_bar.append(np.zeros(numCells))          self.i.append(np.zeros(numCells))          self.f.append(np.zeros(numCells))          self.o.append(np.zeros(numCells))          self.I.append(np.zeros(numCells))          self.z.append(np.zeros(numCells))          O\_W= [self.forwardStep(x\_t,jenis)[8] for x\_t in x]          O\_I= [self.forwardStep(x\_t,jenis)[7] for x\_t in x]          O\_z= [self.forwardStep(x\_t,jenis)[6] for x\_t in x]          O\_c= [self.forwardStep(x\_t,jenis)[1] for x\_t in x]          O\_o= [self.forwardStep(x\_t,jenis)[2] for x\_t in x]          O\_f= [self.forwardStep(x\_t,jenis)[3] for x\_t in x]          O\_in= [self.forwardStep(x\_t,jenis)[4] for x\_t in x]          O\_c\_bar= [self.forwardStep(x\_t,jenis)[5] for x\_t in x]          O\_h = [self.forwardStep(x\_t,jenis)[0] for x\_t in x]          return (O\_I,O\_z,O\_c,O\_o,O\_f,O\_in,O\_c\_bar,O\_h,O\_W)      def backwardStep(self, t, dE\_dh\_t, dE\_dc\_tplus1):          dE\_do\_t = np.multiply(dE\_dh\_t, np.tanh(self.C[t]))          dE\_dc\_t = dE\_dc\_tplus1 + np.multiply(np.multiply(dE\_dh\_t, self.o[t]), (np.ones(self.numCells) - np.square(np.tanh(self.C[t]))))          dE\_di\_t = np.multiply(dE\_dc\_t, self.C\_bar[t])          dE\_dcbar\_t = np.multiply(dE\_dc\_t, self.i[t])          dE\_df\_t = np.multiply(dE\_dc\_t, self.C[t - 1])          dE\_dc\_tminus1 = np.multiply(dE\_dc\_t, self.f[t])          dE\_dzcbar\_t = np.multiply(dE\_dcbar\_t, (np.ones(self.numCells) - np.square(np.tanh(self.z[t][0:self.numCells]))))          dE\_dzi\_t = np.multiply(np.multiply(dE\_di\_t, self.i[t]), (np.ones(self.numCells) - self.i[t]))          dE\_dzf\_t = np.multiply(np.multiply(dE\_df\_t, self.f[t]), (np.ones(self.numCells) - self.f[t]))          dE\_dzo\_t = np.multiply(np.multiply(dE\_do\_t, self.o[t]), (np.ones(self.numCells) - self.o[t]))          dE\_dz\_t = np.concatenate((dE\_dzcbar\_t, dE\_dzi\_t, dE\_dzf\_t, dE\_dzo\_t))          dE\_dI\_t = np.dot(np.transpose(self.W), dE\_dz\_t)          dE\_dh\_tminus1 = dE\_dI\_t[self.inputSize:]          dE\_dz\_t.shape = (len(dE\_dz\_t), 1)          self.I[t].shape = (len(self.I[t]), 1)          dE\_dW\_t = np.dot(dE\_dz\_t, np.transpose(self.I[t]))          return (dE\_dW\_t, dE\_dh\_tminus1, dE\_dc\_tminus1, dE\_do\_t, dE\_dc\_t, dE\_di\_t, dE\_dcbar\_t,dE\_df\_t,dE\_dzcbar\_t,dE\_dzi\_t,dE\_dzf\_t,dE\_dzo\_t,dE\_dz\_t,dE\_dI\_t)      def BPTT(self, y):          numTimePeriods = len(y)          dE\_dW = 0          dE\_dh\_t = 0          dE\_dc\_t = 0          E = 0.0          for i in range(numTimePeriods):              index = numTimePeriods - i              E = E + 0.5 \* np.sum(np.absolute(self.h[index] - y[index - 1]))              lessThan = np.less(self.h[index], y[index - 1])              greaterThan = np.greater(self.h[index], y[index - 1])              dE\_dh\_t -= 0.5 \* lessThan              dE\_dh\_t += 0.5 \* greaterThan              result = self.backwardStep(index, dE\_dh\_t, dE\_dc\_t)              dE\_dW = dE\_dW + result[0]              dE\_dh\_t = result[1]              dE\_dc\_t = result[2]          return (E / (numTimePeriods), dE\_dW)      def train(self, trainingData, numEpochs, learningRate, sequenceLength,max\_ex,min\_ex):          adaptiveLearningRate = learningRate          for epoch in range(numEpochs):              trainingSequences = sequenceProducer(trainingData, sequenceLength)              epochError = 0.0              counter = 0              for sequence in trainingSequences:                  counter += 1                  forecast\_h = self.forwardPass(sequence[:],"no\_prediksi")                  result = self.BPTT(sequence[:,2:])                  update\_bobot = [result[18]]                  E = result[0]                  dE\_dW = result[1]                  w = dE\_dW.shape                  adaptiveLearningRate = learningRate / (1 + (epoch/10))                  self.W = self.W - adaptiveLearningRate \* dE\_dW                  optimasi = [[self.W]]                  epochError += E              print('Epoch ' + str(epoch) + ' error: ' + str(epochError / counter))          return (epochError)      def forecast(self, forecastingData):          forward = self.forwardPass(forecastingData,"prediksi")          f\_l = np.transpose(np.transpose(forward[0]))          f\_z = np.transpose(np.transpose(forward[1]))          f\_c = np.transpose(np.transpose(forward[2]))          f\_o = np.transpose(np.transpose(forward[3]))          f\_f = np.transpose(np.transpose(forward[4]))          f\_i = np.transpose(np.transpose(forward[5]))          f\_c\_bar = np.transpose(np.transpose(forward[6]))          f\_h = np.transpose(np.transpose(forward[7]))          f\_W = np.transpose(np.transpose(forward[8]))          return (f\_h[-1],f\_l,f\_z,f\_c,f\_o,f\_f,f\_i,f\_c\_bar,f\_h,f\_W) |



Gambar 7. Tampilan full Aplikasi