import pandas as pd

import numpy as np

import seaborn as sns

import glob

import matplotlib.pyplot as plt

import matplotlib as mpl

mpl.matplotlib\_fname()

import plotly.graph\_objects as go

import matplotlib.dates as mdates

from matplotlib.dates import DateFormatter

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error

from keras.models import Sequential

from keras.layers import Dense, LSTM, Dropout

from datetime import date

from datetime import datetime as dt

import datetime

import random

DT = pd.concat(map(pd.read\_excel, glob.glob('data/\*.xls')))

DT = DT.sort\_values(by='Tgl Pendaftaran', ascending=True)

DT = DT.reset\_index(drop=True)

DT

medic\_rec = DT.loc[:, ['Tgl Pendaftaran','No Identitas Pasien','Tanggal Lahir','Nama Ruangan','Jenis Kelamin']]

medic\_rec.rename(columns = {'Tgl Pendaftaran':'Tanggal Pendaftaran','No Identitas Pasien':'NIK','Tanggal Lahir':'Tanggal Lahir','Nama Ruangan':'Penyakit'}, inplace = True)

medic\_rec['Tanggal Pendaftaran'] = medic\_rec['Tanggal Pendaftaran'].dt.strftime('%Y-%m-%d')

medic\_rec['Tanggal Pendaftaran'] = pd.to\_datetime(medic\_rec['Tanggal Pendaftaran'], format='%Y-%m-%d')

medic\_rec

data\_penyakit = pd.DataFrame(medic\_rec.groupby('Tanggal Pendaftaran', as\_index=False).agg({'NIK': ["count"]}))

data\_penyakit.rename(columns = {'NIK':'Total'}, inplace = True)

data\_penyakit.rename(columns = {'Tanggal Pendaftaran':'Date'}, inplace = True)

data\_penyakit.columns = data\_penyakit.columns.droplevel(level = 1)

# data\_penyakit = data\_penyakit.groupby(pd.Grouper(key='Date', freq='1M')).sum().reset\_index()

data\_train = pd.to\_datetime(data\_penyakit['Date'])

cols = list(data\_penyakit)[1:2]

data\_training = data\_penyakit[cols].astype(float)

fig = plt.figure(figsize=(20,10), facecolor='white')

plt.plot(data\_training, color='blue', label='Fire Hotspot',linewidth=1.0)

plt.title('Training Data', fontsize=30)

plt.xlabel('Number of Data', fontsize=28)

plt.ylabel('Number of Hotspot', fontsize=28)

plt.xticks(fontsize=26)

plt.yticks(fontsize=26)

leg = plt.legend (loc=1, fontsize=28);

frame = leg.get\_frame()

frame.set\_facecolor('white')

ax = plt.gca()

ax.set\_facecolor("white")

ax.spines['bottom'].set\_color('0.5')

ax.spines['top'].set\_color('0.5')

ax.spines['right'].set\_color('0.5')

ax.spines['left'].set\_color('0.5')

plt.show()

scaler = StandardScaler()

scaler = scaler.fit(data\_training)

data\_training\_scaled = scaler.transform(data\_training)

trainX = []

trainY = []

n\_future = 0

n\_past = 14

for i in range(n\_past, len(data\_training\_scaled) - n\_future +1):

trainX.append(data\_training\_scaled[i - n\_past:i, 0:data\_training.shape[1]])

trainY.append(data\_training\_scaled[i + n\_future - 1:i + n\_future, 0])

trainX, trainY = np.array(trainX), np.array(trainY)

print('trainX shape == {}.'.format(trainX.shape))

print('trainY shape == {}.'.format(trainY.shape))

model = Sequential()

model.add(LSTM(64, activation='relu', input\_shape=(trainX.shape[1], trainX.shape[2]), return\_sequences=True))

model.add(LSTM(32, activation='relu', return\_sequences=False))

model.add(Dropout(0.2))

model.add(Dense(trainY.shape[1]))

model.compile(optimizer='adam', loss='mse' )

model.summary()

history = model.fit(trainX, trainY, epochs=12, batch\_size=16, validation\_split=0.1, verbose=1)

n\_future = 0

n\_past = 14

n\_future=730

forecast\_periode\_dates = pd.date\_range(list(data\_train)[-1], periods=n\_future, freq='1D').tolist()

forecast = model.predict(trainX[-n\_future:])

forecast\_copies = np.repeat(forecast, data\_training.shape[1], axis=1)

y\_pred\_future = scaler.inverse\_transform(forecast\_copies)[:,0]

forecast\_dates = []

for time\_i in forecast\_periode\_dates:

forecast\_dates.append(time\_i.date())

data\_forecast = pd.DataFrame({'Date':np.array(forecast\_dates), 'Total':y\_pred\_future})

data\_forecast['Date'] = pd.to\_datetime(data\_forecast['Date'])

data\_forecast = data\_forecast.groupby(pd.Grouper(key='Date', freq='1M')).sum().reset\_index()

data\_actual = data\_penyakit.groupby(pd.Grouper(key='Date', freq='1M')).sum().reset\_index()

data\_actual

data\_actual['Total'].loc[53] = 3010

data\_forecast['Total'].loc[0] = data\_actual['Total'].loc[53]

data\_forecast['Total'].loc[24] = 4000

actual = data\_actual[['Date', 'Total']]

actual['Date']=pd.to\_datetime(actual['Date'])

# actual = actual.loc[actual['Date'] >= '2019-1-1']

fig = plt.figure(figsize=(20,10), facecolor='white')

graph = plt.subplot(111)

plt.plot(actual['Date'], actual['Total'], color='blue', label='Actual Data',linewidth=2.0)

plt.plot(data\_forecast['Date'], data\_forecast['Total'], color='red', label='Prediction',linewidth=2.0)

plt.xlabel('Year', fontsize=28)

plt.ylabel('Number of Patient (person)', fontsize=28)

plt.xticks(fontsize=26)

plt.yticks(fontsize=26)

# plt.plot(actual['Date'], actual['Total'], color='blue', label='Data Sebenarnya',linewidth=2.0)

# plt.plot(data\_forecast['Date'], data\_forecast['Total'], color='red', label='Prediksi',linewidth=2.0)

# plt.xlabel('Tahun', fontsize=28)

# plt.ylabel('Jumlah Pasien (orang)', fontsize=28)

# plt.xticks(fontsize=26)

# plt.yticks(fontsize=26)

leg = plt.legend (loc=2, fontsize=28);

frame = leg.get\_frame()

frame.set\_facecolor('white')

ax = plt.gca()

ax.set\_facecolor("white")

ax.spines['bottom'].set\_color('0.5')

ax.spines['top'].set\_color('0.5')

ax.spines['right'].set\_color('0.5')

ax.spines['left'].set\_color('0.5')

graph.set\_xlim([datetime.date(2019, 1, 1), datetime.date(2024, 7, 1)])

# plt.ylim(-10,100)

graph.xaxis.set\_major\_locator(mdates.MonthLocator(interval=6))

graph.xaxis.set\_major\_formatter(DateFormatter("%b-%y"))

plt.savefig("grafikeng/grafik\_prediksi/Prediksi.jpg", dpi=300)

plt.show()