IMPORT LIBRARY

import json

import pandas as pd

from pandas.io.json import json\_normalize

from datetime import datetime

from collections import namedtuple

from itertools import groupby, takewhile

from statistics import median, mean

from matplotlib import pyplot

import numpy as np

from geopy.distance import geodesic

import time

import datetime

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

import geopy

import pytz

from geopy.distance import vincenty

from matplotlib.collections import PatchCollection

from IPython.display import Image

from datetime import datetime

warnings.filterwarnings('ignore')

data = json\_normalize(json.load(open('C:/Users/User/Desktop/json/Location History(027).json', 'r'))['locations'])

data.head()

data.tail()

TRANSFORMASI DATA

data['longitude'] = data.longitudeE7 / 10000000

data['latitude'] = data.latitudeE7 / 10000000

data.drop(['longitudeE7', 'latitudeE7'], axis=1, inplace=True)

data.tail()

data['time'] = pd.to\_datetime(data.timestampMs.map(int) \* 1000000)

data['date'] = data['time']

data.tail()

def get\_city(latitude,longitude):

#latitude = int(latitude)

#longitude = int(longitude)

if 2.9253 <= latitude <= 2.9261 and 101.781 <=longitude <= 101.782:

return 'PTSL'

elif 2.9245 <= latitude <= 2.9253 and 101.781 <=longitude <= 101.782:

return 'PPSM'

elif 2.9236 <= latitude <= 2.9239 and 101.781 <=longitude <= 101.782:

return 'DAM'

elif 2.924 <= latitude <= 2.9245 and 101.7815 <=longitude <= 101.782:

return 'GEO'

elif 2.9217 <= latitude <= 2.9224 and 101.779 <=longitude <= 101.781:

return 'New FST'

elif 2.9230 <= latitude <= 2.9236 and 101.781352 <=longitude <= 101.781782:

return 'fizik'

elif 2.9228 <= latitude <= 2.9238 and 101.781 <=longitude <= 101.7826:

return 'Bio'

elif 2.922986 <= latitude <= 2.923344 and 101.780689 <=longitude <= 101.781114:

return 'Kimia'

elif 2.922677 <= latitude <= 2.923075 and 101.780948 <=longitude <= 101.781509:

return 'Kimia'

elif 2.9207 <= latitude <= 2.9223 and 101.779 <=longitude <= 101.7814:

return 'Sains Nuklear'

elif 2.9242 <= latitude <= 2.925 and 101.782 <=longitude <= 101.7829:

return 'FEP'

elif 2.9259 <= latitude <= 2.9267 and 101.78 <=longitude <= 101.7816:

return 'Pusanika'

elif 2.92739 <= latitude <= 2.92854 and 101.7797 <=longitude <= 101.7806:

return 'FSSK'

elif 2.930101 <= latitude <= 2.930759 and 101.778019 <=longitude <= 101.779034:

return 'Dectar'

elif 2.930760 <= latitude <= 2.931666 and 101.778690 <=longitude <= 101.779643:

return 'Punggung Seni'

elif 2.931689 <= latitude <= 2.932528 and 101.784238 <=longitude <= 101.784803:

return 'Stadium'

elif 2.931263 <= latitude <= 2.933037 and 101.784804 <=longitude <= 101.785884:

return 'Stadium'

elif 2.9306 <= latitude <= 2.931238 and 101.7874 <=longitude <= 101.786855:

return 'Dewan Gemilang'

elif 2.9288 <= latitude <= 2.92948 and 101.77674 <=longitude <= 101.77743:

return 'Masjid UKM'

elif 2.925610 <= latitude <= 2.926582 and 101.773497 <=longitude <= 101.775141:

return 'PPBL'

elif 2.925091 <= latitude <= 2.925727 and 101.772550 <=longitude <= 101.772979:

return 'DTAMS'

elif 2.922986 <= latitude <= 2.923658 and 101.773813 <=longitude <= 101.77521:

return 'FPEND'

elif 2.922547 <= latitude <= 2.923658 and 101.768582 <=longitude <= 101.769708:

return 'FUU'

elif 2.919859 <= latitude <= 2.923857 and 101.770872 <=longitude <= 101.773137:

return 'FKAB'

elif 2.919101 <= latitude <= 2.919557 and 101.768590 <=longitude <= 101.769044:

return 'BPA'

elif 2.925775 <= latitude <= 2.926878 and 101.788872 <=longitude <= 101.791243:

return 'KKM'

elif 2.926879 <= latitude <= 2.929429 and 101.787622 <=longitude <= 101.789489:

return 'KKM'

elif 2.928420 <= latitude <= 2.931013 and 101.781647 <=longitude <= 101.784962:

return 'KIZ'

elif 2.923852 <= latitude <= 2.933715 and 101.779575 <=longitude <= 101.784500:

return 'KRK'

elif 2.931839 <= latitude <= 2.923851 and 101.781786 <=longitude <= 101.783615:

return 'KRK'

elif 2.930214 <= latitude <= 2.931652 and 101.779575 <=longitude <= 101.781243:

return 'KDO'

elif 2.929980 <= latitude <= 2.930214 and 101.779717 <=longitude <= 101.781243:

return 'KDO'

elif 2.929671 <= latitude <= 2.930213 and 101.779123 <=longitude <= 101.779716:

return 'KTHO'

elif 2.929671 <= latitude <= 2.929980 and 101.779716 <=longitude <= 101.780110:

return 'KTHO'

elif 2.923558 <= latitude <= 2.926161 and 101.783211 <=longitude <= 101.784657:

return 'KAB'

elif 2.926162 <= latitude <= 2.927191 and 101.782237 <=longitude <= 101.783337:

return 'KAB'

elif 2.923483 <= latitude <= 2.924983 and 101.777892 <=longitude <= 101.778712:

return 'KIY'

elif 2.921442 <= latitude <= 2.923482 and 101.778026 <=longitude <= 101.779378:

return 'KIY'

elif 2.924547 <= latitude <= 2.925554 and 101.778799 <=longitude <= 101.781143:

return 'KUO'

elif 2.923781 <= latitude <= 2.924546 and 101.779394 <=longitude <= 101.781143:

return 'KUO'

elif 2.926845 <= latitude <= 2.928190 and 101.776621 <=longitude <= 101.778456:

return 'KBH'

elif 2.926004 <= latitude <= 2.926844 and 101.778043 <=longitude <= 101.778949:

return 'KBH'

elif 2.919095 <= latitude <= 2.920663 and 101.773252 <=longitude <= 101.776264:

return 'KPZ'

elif 2.917913 <= latitude <= 2.919094 and 101.773252 <=longitude <= 101.774034:

return 'KPZ'

elif 2.917913 <= latitude <= 2.919094 and 101.774743 <=longitude <= 101.776264:

return 'KPZ'

elif 2.928544 <= latitude <= 2.939036 and 101.774060 <=longitude <= 101.774640:

return 'FUTSAL KPZ'

elif 2.932982 <= latitude <= 2.933248 and 101.780574 <=longitude <= 101.780887:

return 'Gelanggang Bola Tampar Pantai UKM'

elif 2.932088 <= latitude <= 2.932885 and 101.780886 <=longitude <= 101.781391:

return 'Gelanggang Tenis'

elif 2.932466 <= latitude <= 2.932937 and 101.780345 <=longitude <= 101.780821:

return 'Gelanggang Bola Tampar'

elif 2.932466 <= latitude <= 2.932794 and 101.780139 <=longitude <= 101.780320:

return 'Gelanggang Bola Keranjang'

elif 2.932368 <= latitude <= 2.932667 and 101.779932 <=longitude <= 101.780131:

return 'Gelanggang Bola Keranjang'

elif 2.932037 <= latitude <= 2.932322 and 101.779677 <=longitude <= 101.780157:

return 'Gelanggang Bola Jaring'

elif 2.925070 <= latitude <= 2.925877 and 101.789580 <=longitude <= 101.790097:

return 'Gelanggang KKM'

elif 2.933716 <= latitude <= 2.933976 and 101.781326 <=longitude <= 101.785611:

return 'Kompleks Sukan'

elif 2.933976 <= latitude <= 2.934966 and 101.783857 <=longitude <= 101.785611:

return 'Kompleks Sukan'

elif 2.933976 <= latitude <= 2.935728 and 101.785612 <=longitude <= 101.783856:

return 'Kompleks Sukan'

elif 2.934497 <= latitude <= 2.935948 and 101.785660 <=longitude <= 101.788049:

return 'DANAU'

elif 2.933748 <= latitude <= 2.934409 and 101.786354 <=longitude <= 101.787194:

return 'Kolam Renang Danau'

elif 2.931886 <= latitude <= 2.932234 and 101.786945 <=longitude <= 101.787580:

return 'Kompleks Squasy'

elif 2.923083 <= latitude <= 2.923450 and 101.781812 <=longitude <= 101.782004:

return 'FST CAFE'

elif 2.928587 <= latitude <= 2.928908 and 101.779299 <=longitude <= 101.780324:

return 'Bangunan Canselori'

elif 2.926663 <= latitude <= 2.928306 and 101.785365 <=longitude <= 101.787140:

return 'CRIM'

elif 2.924692 <= latitude <= 2.925745 and 101.787789 <=longitude <= 101.789040:

return 'EIMAS'

elif 2.926457 <= latitude <= 2.926763 and 101.781112 <=longitude <= 101.781488:

return 'Pusanika'

elif 2.926148 <= latitude <= 2.926456 and 101.780847 <=longitude <= 101.781667:

return 'Pusanika'

elif 2.925912 <= latitude <= 2.926147 and 101.780788 <=longitude <= 101.781391:

return 'Pusanika'

elif 2.932672 <= latitude <= 2.933015 and 101.786938 <=longitude <= 101.787284:

return 'Bahagian Keselamatan UKM'

elif 2.931256 <= latitude <= 2.931864 and 101.786207 <=longitude <= 101.786880:

return 'Pusat Kembangan Pendidikan UKM'

elif 2.928882 <= latitude <= 2.929217 and 101.786989 <=longitude <= 101.787401:

return 'Pengajaran-UKM'

elif 2.915215 <= latitude <= 2.915885 and 101.789017 <=longitude <= 101.789601:

return 'Kolam Renang PERMATA pintar'

elif 2.914781 <= latitude <= 2.916415 and 101.789602 <=longitude <= 101.790701:

return 'Stadium PERMATA pintar'

elif 2.915886 <= latitude <= 2.918130 and 101.787600 <=longitude <= 101.789601:

return 'Pusat PERMATA pintar'

elif 2.928587 <= latitude <= 2.929591 and 101.785180 <=longitude <= 101.786537:

return 'Pusat Siswazah UKM'

elif 2.925313 <= latitude <= 2.925756 and 101.771357 <=longitude <= 101.772148:

return 'SERI'

elif 2.917363 <= latitude <= 2.937034 and 101.78202 <=longitude <= 101.793367:

return 'Tempat Lain dalam UKM'

elif 2.920408 <= latitude <= 2.933180 and 101.776284 <=longitude <= 101.78202:

return 'Tempat Lain dalam UKM'

elif 2.917340 <= latitude <= 2.927691 and 101.768593 <=longitude <= 101.776284:

return 'Tempat Lain dalam UKM'

else:

return 'Luar UKM'

data['city'] = np.vectorize(get\_city)(data['latitude'], data['longitude'])

degrees\_to\_radians = np.pi/180.0

data['phi'] = (90.0 - data.latitude) \* degrees\_to\_radians

data['theta'] = data.longitude \* degrees\_to\_radians

# Compute distance between two GPS points on a unit sphere

data['distance'] = np.arccos(

np.sin(data.phi)\*np.sin(data.phi.shift(-1)) \* np.cos(data.theta - data.theta.shift(-1)) +

np.cos(data.phi)\*np.cos(data.phi.shift(-1))) \* 6378.100

data['day\_of\_week'] = data['time'].dt.weekday\_name

data.tail()

data['hour'] = data['time'].dt.hour

data['minute'] = data['time'].dt.minute

data['second'] = data['time'].dt.second

data.head()

start\_date = '2018-09-10'

end\_date = '2018-12-21'

mask = (data['time'] >= start\_date) & (data['time'] <= end\_date)

data1 = data.loc[mask]

#data1 = data1.drop(data1[data1.longitude < 101.76].index)

#data1 = data1.drop(data1[data1.longitude > 101.8].index)

data1.head()

data1.describe()

data1['frequency'] = np.where(data1['city']==data1['city'].shift(),0,1)

PENGGABUNGAN DATA

import pandas as pd

import numpy as np

import getpass

user = getpass.getuser()

#data1 = pd.read\_pickle(path = r"C:/Users/" + user + r"/Desktop/020lalanew.pkl")

copy\_index = data1.index # retain the index before sorting

data1.sort\_values(by='time', ascending=True, inplace=True)

# make sure both the time and index are in ascending order

data1.index = copy\_index

cities = data1['city'].unique()

intervals = []

for city in cities:

# filter row by city name, then get the consecutive row only

city\_index = data1[data1['city'] == city].index # record the index by city

select\_continuous = np.append(0,np.diff(city\_index,1)) == 1

ori\_date = pd.to\_datetime(data1.loc[city\_index, 'time'])

shift\_date = ori\_date.shift(1)

interval = ori\_date - shift\_date

interval[np.append(0,np.diff(interval.index)) != 1] = pd.Timedelta(0)

interval.fillna(pd.Timedelta(0),inplace=True)

data1.loc[city\_index, 'interval'] = interval

interval = data1['interval']

data1['interval'] = [interval.at[ids].seconds for ids, time\_delta in interval.iteritems()]

data1[['sum']] = data1.groupby('city')['interval'].agg(['cumsum'])

data1[['sumf']]=data1.groupby('city')['frequency'].agg(['cumsum'])

data1 = data1.drop(data1[data1.accuracy > 1000].index)

pd.set\_option('display.max\_rows', 500)

pd.set\_option('display.max\_columns', 500)

pd.set\_option('display.width', 1000)

export\_csv = data1.to\_csv (r'C:\Users\User\Desktop\dataset\027.csv', index = None, header=True)

#data1.to\_pickle(r'C:\Users\User\Desktop\020lalanew.pkl')

data1.accuracy.describe()

count 2748.000000

mean 38.570961

std 40.768827

min 1.000000

25% 12.000000

50% 24.000000

75% 65.000000

max 510.000000

Name: accuracy, dtype: float64

PIVOT TABLE

import numpy as np

table = pd.pivot\_table(df, values='interval', index=['filename'],columns=['city'], aggfunc=np.sum)

table=table.fillna(0)

table['FST']=table['Bio']+table['DAM']+table['PPSM']+table['fizik']+table['GEO']+table['Kimia']+table['New FST']+table['Sains Nuklear']

table2=pd.pivot\_table(df, values='frequency', index=['filename'],columns=['city'], aggfunc=np.sum)

table2['FST']=table2['Bio']+table2['DAM']+table2['PPSM']+table2['fizik']+table2['GEO']+table2['Kimia']+table2['New FST']+table2['Sains Nuklear']

table2=table2.fillna(0)

DATA SEJARAH LOKASI DIGABUNGKAN DENGAN GOOGLE FORM DATA

df2 = pandas.read\_csv('C:/Users/User/Desktop/all/3newmergedinterval.csv')

df3 = pandas.read\_csv('C:/Users/User/Desktop/all/lxh\_fyp6a.csv')

df4 = pd.merge(df2, df3, on='filename', how='inner')

STATISTIK DESKRIPTIF

df.describe()

MENJANA PLOT KOTAK

sns.boxplot(y='gpa',data=df)

# Add title and axis names

plt.title('GPA')

#plt.xlabel('masa')

plt.ylabel('GPA')

plt.savefig('GPA.png', dpi = 300)

sns.boxplot(data=df)

# Add title and axis names

plt.title('Masa untuk Isnin-Jumaat vs Sabtu-Ahad')

#plt.xlabel('categories')

plt.ylabel('Interval')

sns.boxplot(x='cluster', y='gpa',data=df)

# Add title and axis names

plt.title('Perbezaan GPA antara kelompok')

plt.xlabel('kelompok')

plt.ylabel('GPA')

plt.savefig('Perbezaan GPA antara kelompok.png', dpi = 300)

sns.boxplot(x='cluster', y='gpa', data=df)

ANALISIS KORELASI

df1= pandas.read\_csv('C:/Users/User/Desktop/withoutholiday2/2newmergedinterval-other2.csv')

df2= pandas.read\_csv('C:/Users/User/Desktop/withoutholiday2/2newmergedinterval-college.csv')

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

from pandas.tools.plotting import scatter\_matrix

axes = scatter\_matrix(df, alpha=0.5, diagonal='kde',figsize=(9,9))

corr = df.corr().as\_matrix()

for i, j in zip(\*plt.np.triu\_indices\_from(axes, k=1)):

axes[i, j].annotate("%.3f" %corr[i,j], (0.8, 0.8), xycoords='axes fraction', ha='center', va='center')

plt.show()

PENGELOMPOKAN K-MIN

import pandas as pd

import numpy as np

df = pd.read\_csv(r'C:\Users\User\Desktop\fretablemax.csv')

# Extract the Features from the Data

X = pd.DataFrame(df[["Bio", "GEO", "Kimia", "PPSM", "Sains Nuklear", "fizik"]])

import seaborn as sb

sb.set()

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

# Plot the Raw Data on 2D grids

sb.pairplot(X)

# Vary the Number of Clusters

min\_clust = 1

max\_clust = 40

init\_algo = 'k-means++'

# Compute Within Cluster Sum of Squares

within\_ss = []

for num\_clust in range(min\_clust, max\_clust+1):

kmeans = KMeans(n\_clusters = num\_clust, init = init\_algo, n\_init = 5)

kmeans.fit(X)

within\_ss.append(kmeans.inertia\_)

# Angle Plot : Within SS vs Number of Clusters

f, axes = plt.subplots(1, 1, figsize=(16,4))

plt.plot(range(min\_clust, max\_clust+1), within\_ss)

plt.xlabel('Number of Clusters')

plt.ylabel('Within Cluster Sum of Squares')

plt.xticks(np.arange(min\_clust, max\_clust+1, 1.0))

plt.grid(which='major', axis='y')

plt.show()

# Set "optimal" Clustering Parameters

num\_clust = 6

init\_algo = 'k-means++'

# Create Clustering Model using KMeans

kmeans = KMeans(n\_clusters = num\_clust,

init = init\_algo,

n\_init = 100)

# Fit the Clustering Model on the Data

kmeans.fit(X)

KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=300,

n\_clusters=6, n\_init=100, n\_jobs=None, precompute\_distances='auto',

random\_state=None, tol=0.0001, verbose=0)

# Predict the Cluster Labels

labels = kmeans.predict(X)

# Append Labels to the Data

X\_labeled = X.copy()

X\_labeled["Cluster"] = pd.Categorical(labels)

sb.pairplot(X\_labeled, vars = X.columns.values, hue = "Cluster")

# Boxplots for all Features against the Clusters

f, axes = plt.subplots(6, 1, figsize=(16,24))

sb.boxplot(x = 'Bio', y = 'Cluster', data = X\_labeled, ax = axes[0])

sb.boxplot(x = 'GEO', y = 'Cluster', data = X\_labeled, ax = axes[1])

sb.boxplot(x = 'Kimia', y = 'Cluster', data = X\_labeled, ax = axes[2])

sb.boxplot(x = 'PPSM', y = 'Cluster', data = X\_labeled, ax = axes[3])

sb.boxplot(x = 'Sains Nuklear', y = 'Cluster', data = X\_labeled, ax = axes[4])

sb.boxplot(x = 'fizik', y = 'Cluster', data = X\_labeled, ax = axes[5])

# Average Behaviour of each Cluster

cluster\_data = pd.DataFrame(X\_labeled.groupby(by = "Cluster").mean())

cluster\_data.plot.bar(figsize = (16,6))

# Show each student’s cluster

df['cluster'] = kmeans.labels\_

df

df.sort\_values('cluster', inplace=True, ascending=True)