Author Name: Vangari Prashanth

Student Id: 11645119

- TASK 1

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
def getData(filename):
  canData=[] #List to store CAN data
  f = open(filename)
  #read_file = reader(f)
  read_file =f.readlines()
  #file = list(read_file)
  speed = []
  rpm = []
  i = 0
  for row in read_file:
    #Change the positions of the values if needed
    record = {'stamp':row[1:18], 'PID':row[25:28], 'const1':row[29:33],
'change':row[33:41],'value':int(row[41:45], 16), 'value2':0 ,'attack':0}
    if record["PID"] == '254': #Processing of speed
      if record["value"] >= 4095:
        record["attack"] = 1
      record['value'] = (record['value'] * 0.62137119) /100
      speed.append(record['value'])
    if record["PID"] == '115': #Processing of RPM
      if record["value"] >= 65535:
        record["attack"] = 1
      record['value'] = (record['value'] * 2)
      rpm.append(record['value'])
    i = i+1
    canData.append(record)
    record={}
  f.close()
  return canData
def dict to df(dict):
  #load dictionary to dataframe
  df = pd.DataFrame.from_dict(dict)
  df = df.reset_index(drop=True)  # Dropping the Column Attack
  df_RPM = df.loc[(df['PID'] == '115')]
  df_Speed = df.loc[(df['PID'] == '254')]
  return df_RPM,df_Speed
#edit file name with file directory of downloaded log files
f_inj_RPM, f_inj_Speed = dict_to_df(getData("CAN Bus log - injection of FFF as the speed reading.log"))
\verb|r_inj_RPM|, | r_inj_Speed=| dict_to_df(getData("CAN Bus log - injection of RPM readings.log")|| \\
no_inj_RPM, no_inj_Speed = dict_to_df(getData("CAN bus log - no injection of messages.log"))
```

▼ TASK 2: K - Means Clustering

```
f_inj_RPM.drop(['attack'], axis = 1)
f_inj_Speed.drop(['attack'], axis = 1)
r_inj_RPM.drop(['attack'], axis = 1)
r_inj_Speed.drop(['attack'], axis = 1)
no_inj_RPM.drop(['attack'], axis = 1)
```

20 1522004165.996440 254 918F D364239E 0.000000 0 55 1522004166.08373 254 918F C364339E 0.000000 0 64 1522004166.036373 254 918F 336439E 0.000000 0 124 1522004166.056373 254 918F 336439E 0.000000 0 125 157 1522004166.076394 254 918F D36639E 0.000000 0 157 1522004160.076394 254 918F D36639E 0.000000 0 157 1522004180.036277 254 918F D36639E 3.000000 0 157 1522004180.036277 254 918F D36639E 3.249771 0 23825 1522004180.055837 254 918F D366239F 3.249771 0 23825 1522004180.056872 254 918F D366239F 3.249771 0 23825 1522004180.05697 254 918F D366239F 3.249771 0 23825 152204180.05697 254 918F		stamp	PID	const1	change	value	value2	3							
### ### ### ### ### ### ### ### ### ##	20	1522004165.996440	254	918F	D364239E	0.000000	0 11	1							
124 152204166.056434 254 918F A364538E 0.000000 0 157 1522004160.076394 254 918F 9384639E 0.000000 0	55	1522004166.016375	254	918F	C364339E	0.000000	0	•							
157 1522004180.076394 254 918F 9384639E 0.000000 0	84	1522004166.036373	254	918F	B364439E	0.000000	0								
23823 1522004180.036277 254 918F D096039F 3.386473 0 23880 1522004180.055837 254 918F D096039F 3.444736 0 23885 1522004180.056837 254 918F D096039F 3.144136 0 23925 1522004180.066297 254 918F D096039F 3.038505 0 23926 1522004180.066297 254 918F D096039F 3.038505 0 23926 1522004180.165809 254 918F D096039F 2.932872 0 707 rows ×6 columns f_ini_Speed_reset_index(drop=True, inplace=True)	124	1522004166.056434	254	918F	A364539E	0.000000	0								
23823 1522004180.056387 254 918F D086039F 3.364473 0 23860 1522004180.055837 254 918F D086639F 3.249771 0 23885 1522004180.056837 254 918F D086339F 3.444138 0 23825 1522004180.056287 254 918F D086339F 3.43636 0 23826 1522004180.015850 254 918F D086339F 3.036505 0 23961 1522004180.15850 254 918F D086339F 2.932872 0 707 rows × 6 columns f_inj_Sped_reset_index(drop=True, inplace=True) f_inj_Sped_reset_index(drop=True, inplace=True) r_inj_Sped_reset_index(drop=True, inplace=True) r_inj_Sped_reset_index(drop=True, inplace=True) n_inj_Sped_reset_index(drop=True, inplace=True) n_inj_Sped_keans.cluster nosinj_Sped_reset_index(drop=True, inplace=True) n_inj_Sped_keans.cluster n_inj_Sped_keans.cluster scaler = StandardScaler() #1 f_inj_Sped_keans.fit(f_inj_Sped_keans.labels_ f_inj_Sped_keans.fit(f_inj_Sped_keans.labels_ f_inj_Sped_keans.centroids = f_inj_Sped_keans.cluster_centers_ f_inj_Sped_keans.centroid = n_inj_Sped_keans.centroids) /usr/local/lib/python3.18/dist-packages/sklearn/cluster/_kmeans.py:878: FutureWarning: The default value warnings.warn(#2 #3 r_inj_Sped_keans.centroids = f_inj_RPM_keans.centroids) /usr/local/lib/python3.18/dist-packages/sklearn/cluster/_kmeans.py:878: FutureWarning: The default value warnings.warn(#3 r_inj_Sped_keans.centroids = f_inj_Sped_keans.labels_ f_inj_Sped_keans.sentroids = n_inj_Sped_keans.labels_ f_inj_Sped_keans.sentroids = n_inj_Sped_keans.labels_ r_inj_Sped_keans.sentroids	157	1522004166.076394	254	918F	9364639E	0.000000	0								
23860 1522004180.05837 254 918F D696139F 3.249771 0 23885 1522004180.05837 254 918F D696339F 3.144138 0 23925 1522004180.096297 254 918F D696339F 3.038505 0 23961 1522004180.096297 254 918F D696339F 3.038505 0 707 rows x 6 columns f_inj_RPM.reset_index(drop=True, inplace=True) f_inj_Speed.reset_index(drop=True, inplace=True) f_inj_Speed.reset_index(drop=True, inplace=True) n_inj_RPM.reset_index(drop=True, inplace=True) n_inj_RPM.reset_index(drop=True, inplace=True) n_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_Speed.reset_index(drop=True, inplace=True) no_inj_Speed.reset_index(drop=True, inplace=True) from sklearn.cluster import KMeans from sklearn.preprocessing import StandardScaler scaler = StandardScaler() #1 f_inj_Speed(['stamp', 'value']] = scaler.fit_transform(f_inj_Speed[['stamp', 'value']]) f_inj_Speed(seans = RMeans(n_clusters=2) f_inj_Speed(seans.sit(f_inj_Speed(['stamp', 'value'])) f_inj_Speed(seans.sit(f_inj_Speed(seans.cluster_centers_f_inj_Speed_seans.cluster_f_index_															
23925 1522004180.096297 254 918F D896239F 3.144138 0 23925 1522004180.096297 254 918F D996339F 3.038505 0 23961 1522004180.115850 254 918F DA96439F 2.932872 0 707 rows × 6 columns f.inj_RPM.reset_index(drop=True, inplace=True) f.inj_Speed_reset_index(drop=True, inplace=True) f.inj_Speed_reset_index(drop=True, inplace=True) r.inj_Speed_reset_index(drop=True, inplace=True) r.inj	23823	1522004180.036277	254	918F	D096039F	3.386473	0								
23925 152204180.096297 254 918F D996339F 3.038505 0 23961 152204180.116850 254 918F DA96439F 2.932872 0 707 rows × 6 columns f_inj_RPM.reset_index(drop=True, inplace=True) f_lnj_Speed.reset_index(drop=True, inplace=True) f_lnj_Speed.reset_index(drop=True, inplace=True) r_lnj_Speed.reset_index(drop=True, inplace=True) r_lnj_Speed.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) from sklearn.cluster import KMeans from sklearn.preprocessing import StandardScaler scaler = StandardScaler() #1 f_inj_Speed(['stamp', 'value']] = scaler.fit_transform(f_inj_Speed['stamp', 'value']]) f_inj_Speed_kmeans = KMeans(n_clusters=2) f_inj_Speed_kmeans.fit(f_inj_Speed_kmeans.labels, f_inj_Speed_kmeans_entroids = f_inj_Speed_kmeans_rentroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureNarning: The default value warnings.warn(#2 f_inj_RPM_lmeans = KMeans(n_clusters=2) f_inj_RPM_lme	23860	1522004180.055837	254	918F											
23961 152204180.115850 254 918F DA96439F 2.932872 0 707 rows × 6 columns f_inj_RPM.reset_index(drop=True, inplace=True) f_inj_Sped.reset_index(drop=True, inplace=True) r_inj_RPM.reset_index(drop=True, inplace=True) r_inj_RPM.reset_index(drop=True, inplace=True) no_inj_Sped.reset_index(drop=True, inplace=True) no_inj_Sped.reset_index(drop=True, inplace=True) no_inj_Sped.reset_index(drop=True, inplace=True) no_inj_Sped.reset_index(drop=True, inplace=True) from sklearn.cluster import KMeans from sklearn.cluster import KMeans from sklearn.preprocessing import StandardScaler scaler = StandardScaler() #1 f_inj_Sped_(istamp', 'value']] = scaler.fit_transform(f_inj_Sped[['stamp', 'value']]) f_inj_Sped_kmeans = KMeans(n_clusters=2) f_inj_Sped_kmeans.fit(f_inj_Sped_kmeans.labels_f_inj_Sped_kmeans.cluster_centers_f_inj_Sped_centroid = np.array(f_inj_Sped_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value = warnings.warn(#2 f_inj_RPM_kmeans = KMeans(n_clusters=2) f_inj_RPM_kmeans = f_inj_RPM_kmeans.cluster_centers_f_inj_RPM_kmeans_f_inj_RPM_kmeans_centroids = f_inj_RPM_kmeans.cluster_centers_f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value = warnings.warn(#3 r_inj_Sped_kmeans = KMeans(n_clusters=2) r_inj_Sped_kmeans = KMeans(n_clusters=2) r_inj_Sped_kmeans = KMeans(n_clusters=2) r_inj_Sped_kmeans = KMeans(n_clusters=2) r_inj_Sped_kmeans = RMeans(n_clusters=2) r_inj_Sped_kmeans_centroids = r_inj_Sped_kmeans.cluster_centers_r_inj_Sped_kmeans_centroids = r_inj_Sped_kmeans.cluster_centers_r_inj_Sped_kmeans_centroids = r_inj_Sped_kmeans.cluster_centers_r_inj_Sped_kmeans_centroids = r_inj_Sped_kmeans_centroids = r_inj_Sped_kmeans_				918F											
<pre>f_inj_RPM.reset_index(drop=True, inplace=True) f_inj_RPM.reset_index(drop=True, inplace=True) f_inj_RPM.reset_index(drop=True, inplace=True) r_inj_Speed.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) from sklearn.cluster import KNeans from sklearn.cluster import KNeans from sklearn.preprocessing import StandardScaler scaler = StandardScaler() #1 f_inj_Speed['stamp', 'value']] = scaler.fit_transform(f_inj_Speed[['stamp', 'value']]) f_inj_Speed_kmeans.fit(f_inj_Speed['stamp', 'value']]) f_inj_Speed_kmeans.fit(f_inj_Speed_kmeans.labels_f_inj_Speed_kmeans_cuntroids) //usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#2 f_inj_RPM_('stamp', 'value')] = scaler.fit_transform(f_inj_RPM[('stamp', 'value')]) f_inj_RPM_kmeans.fit(f_inj_RPM_kmeans.labels_f_inj_RPM_kmeans.sentroids = f_inj_RPM_kmeans.sentroids //usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#3 r_inj_Speed_kmeans = KMeans(_clusters=2) r_inj_Speed_kmeans.sentroids = f_inj_RPM_kmeans.cluster_centers_f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans.cluster_centers_f_inj_RPM_centroid = np.array(f_inj_Speed_kmeans.centroids) //usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#3 r_inj_Speed_kmeans = KMeans(_clusters=2) r_inj_Speed_kmeans_centroids = f_inj_Speed_kmeans.cluster_centers_r_inj_Speed_kmeans.rateroids = r_inj_Speed_kmeans.centroids = r_inj_Speed_kmeans.centr</pre>															
<pre>f_inj_RPM.reset_index(drop=True, inplace=True) f_inj_Speed.reset_index(drop=True, inplace=True) r_inj_RPM.reset_index(drop=True, inplace=True) r_inj_Speed.reset_index(drop=True, inplace=True) no_inj_RPM.reset_index(drop=True, inplace=True) no_inj_Speed.reset_index(drop=True, inplace=True) no_inj_Speed.reset_index(drop=True, inplace=True) from sklearn.cluster import KMeans from sklearn.cluster scaler = StandardScaler() ### f_inj_Speed_kmeans = KMeans(n_clusters=2) f_inj_Speed_kmeans = KMeans(n_cluster) f_inj_Speed_kmeans.cluster f_inj_RPM_Kmeans = KMeans(n_cluster) #### f_inj_RPM_kmeans = KMeans(n_cluster) f_inj_Speed_kmeans = KMeans(n_cluster) f_inj_Spe</pre>			254	918F	DA96439F	2.932872	0								
<pre>grow sklearn.preprocessing import StandardScaler scaler = StandardScaler() #1 f_inj_Speed[['stamp', 'value']] = scaler.fit_transform(f_inj_Speed[['stamp', 'value']]) f_inj_Speed_kmeans = kMeans(n_clusters=2) f_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ f_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ f_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ f_inj_Speed_cmtroid = np.array(f_inj_Speed_kmeans.custer_centers_ f_inj_Speed_cmtroid = np.array(f_inj_Speed_kmeans.custer_kmeans.py:870: FutureWarning: The default value of warnings.warn() #2 f_inj_RPM[['stamp', 'value']] = scaler.fit_transform(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans = kMeans(n_clusters=2) f_inj_RPM_kmeans.fit(f_inj_RPM]_kmeans.labels_ f_inj_RPM_kmeans_centroids = f_inj_RPM_kmeans.custer_centers_ f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans.centroids) #3 r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.custer_centers_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.custer_centers_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.custer_centers_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.custer_centers_ r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.custer_centers_ r_inj_Speed_cmtroid = np.array(r_inj_Speed_kmeans.custer_kmeans.py:870: FutureWarning: The default value of the part of th</pre>	f_inj_Speed. r_inj_RPM.re r_inj_Speed. no_inj_RPM.r	reset_index(drop=T set_index(drop=Tru reset_index(drop=T eset_index(drop=Tr	rue, e, in rue, ue, i	inplace= place=Tr inplace= nplace=T	True) ue) True) rue)										
#1 f_inj_Speed[['stamp', 'value']] = scaler.fit_transform(f_inj_Speed[['stamp', 'value']]) f_inj_Speed_kmeans = KMeans(n_clusters=2) f_inj_Speed_kmeans.fit(f_inj_Speed[['stamp', 'value']]) f_inj_Speed_kmeans.fit(f_inj_Speed_kmeans.labels_ f_inj_Speed_kmeans_centroids = f_inj_Speed_kmeans.cluster_centers_ f_inj_Speed_centroid = np.array(f_inj_Speed_kmeans.cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#2 #2 #2 #2 #3 #4 #5 #6 #6 #7 #6 #7 #7 #7 #8 #8 #8 #8 #8 #8 #8		•		tandardS	caler										
<pre>f_inj_Speed[['stamp', 'value']] = scaler.fit_transform(f_inj_Speed[['stamp', 'value']]) f_inj_Speed_kmeans = KMeans(n_clusters=2) f_inj_Speed_kmeans.fit(f_inj_Speed['stamp', 'value']]) f_inj_Speed_kmeans.fit(f_inj_Speed_Kmeans.labels_ f_inj_Speed_kmeans_centroids = f_inj_Speed_kmeans.cluster_centers_ f_inj_Speed_centroid = np.array(f_inj_Speed_kmeans.cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#2 f_inj_RPM[['stamp', 'value']] = scaler.fit_transform(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans = KMeans(n_clusters=2) f_inj_RPM_kmeans.fit(f_inj_RPM_kmeans.labels_ f_inj_RPM_kmeans.fit(f_inj_RPM_kmeans.labels_ f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans.centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#3 r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_entroids = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_ent</pre>	scaler = Sta	ndardScaler()													
#2 f_inj_RPM[['stamp', 'value']] = scaler.fit_transform(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans = KMeans(n_clusters=2) f_inj_RPM_kmeans.fit(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans.fit(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans.fit(f_inj_RPM_kmeans.labels_ f_inj_RPM_kmeans_centroids = f_inj_RPM_kmeans.cluster_centers_ f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#3 r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of the control o	<pre>f_inj_Speed[f_inj_Speed_ f_inj_Speed_ f_inj_Speed[f_inj_Speed_</pre>	kmeans = KMeans(n_ kmeans.fit(f_inj_S 'KMeans'] = f_inj_ kmeans_centroids =	clust peed[Speed f_in	ers=2) ['stamp' _kmeans. j_Speed_	_ ,'value']]) labels_ kmeans.clus	ster_cente		, 'value']])							
<pre>#2 f_inj_RPM[['stamp', 'value']] = scaler.fit_transform(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans = KMeans(n_clusters=2) f_inj_RPM_kmeans.fit(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans.fit(f_inj_RPM_kmeans.labels_ f_inj_RPM_kmeans_centroids = f_inj_RPM_kmeans.cluster_centers_ f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value warnings.warn(#3 r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed_['stamp', 'value']]) r_inj_Speed['KMeans'] = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value</pre>			/dist	-package	s/sklearn/o	:luster/_k	means.py:870	: FutureWarning:	The default	value	of	of `n_init`	of `n_init` will	of `n_init` will chan	of `n_init` will chang
<pre>f_inj_RPM[['stamp', 'value']] = scaler.fit_transform(f_inj_RPM[['stamp', 'value']]) f_inj_RPM_kmeans = KMeans(n_clusters=2) f_inj_RPM_kmeans.fit(f_inj_RPM[['stamp', 'value']]) f_inj_RPM['KMeans'] = f_inj_RPM_kmeans.labels_ f_inj_RPM_kmeans_centroids = f_inj_RPM_kmeans.cluster_centers_ f_inj_RPM_centroid = np.array(f_inj_RPM_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of warnings.warn(#3 r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans.fit(r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of the complex contents and the complex contents are contents. // **Complex contents are contents and the complex contents are contents are contents. // **Complex contents are contents and contents are contents are contents. // **Complex contents are contents are contents are contents. // **Complex contents are contents are contents are contents are contents. // **Complex contents are contents are contents are contents are contents are contents are contents. // **Complex contents are contents. // **Complex contents are contents are contents are co</pre>		·													
#3 r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed[['stamp', 'value']]) r_inj_Speed['KMeans'] = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of	f_inj_RPM[[' f_inj_RPM_km f_inj_RPM_km f_inj_RPM['K f_inj_RPM_km	eans = KMeans(n_cl eans.fit(f_inj_RPM Means'] = f_inj_RPI eans_centroids = f	uster [['st M_kme _inj_	s=2) amp','va ans.labe RPM_kmea	lue']]) ls_ ns.cluster_	_centers_	['stamp', 'v	alue']])							
<pre>r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed[['stamp','value']]) r_inj_Speed['KMeans'] = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value or the content of the con</pre>			/dist	-package	s/sklearn/o	:luster/_k	means.py:870	: FutureWarning:	The default	value o	f	f `n_init`	f `n_init` will	f `n_init` will chan	f `n_init` will chang
<pre>r_inj_Speed[['stamp', 'value']] = scaler.fit_transform(r_inj_Speed[['stamp', 'value']]) r_inj_Speed_kmeans = KMeans(n_clusters=2) r_inj_Speed_kmeans.fit(r_inj_Speed[['stamp','value']]) r_inj_Speed['KMeans'] = r_inj_Speed_kmeans.labels_ r_inj_Speed_kmeans_centroids = r_inj_Speed_kmeans.cluster_centers_ r_inj_Speed_centroid = np.array(r_inj_Speed_kmeans_centroids) /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value or the content of the con</pre>	4														
	r_inj_Speed[r_inj_Speed_ r_inj_Speed_ r_inj_Speed[r_inj_Speed_ r_inj_Speed_	kmeans = KMeans(n_ kmeans.fit(r_inj_S 'KMeans'] = r_inj_ kmeans_centroids = centroid = np.arra	clust peed[Speed r_in y(r_i	ers=2) ['stamp' _kmeans. j_Speed_ nj_Speed	,'value']]; labels_ kmeans.clus _kmeans_cer	ster_cente	ers_		The default	value of	=	· `n_init`	· `n_init` wil:	· `n_init` will chan	· `n_init` will chang
				_		_		3				_	_	_	_

```
#4
r_inj_RPM[['stamp', 'value']] = scaler.fit_transform(r_inj_RPM[['stamp', 'value']])
r_inj_RPM_kmeans = KMeans(n_clusters=2)
r_inj_RPM_kmeans.fit(r_inj_RPM[['stamp','value']])
r_inj_RPM['KMeans'] = r_inj_RPM_kmeans.labels_
r_inj_RPM_kmeans_centroids = r_inj_RPM_kmeans.cluster_centers_
r inj RPM centroid = np.array(r inj RPM kmeans centroids)
     /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
      warnings.warn(
#5
no_inj_Speed[['stamp', 'value']] = scaler.fit_transform(no_inj_Speed[['stamp', 'value']])
no inj Speed kmeans = KMeans(n clusters=2)
no_inj_Speed_kmeans.fit(no_inj_Speed[['stamp','value']])
no_inj_Speed['KMeans'] = no_inj_Speed_kmeans.labels_
no_inj_Speed_kmeans_centroids = no_inj_Speed_kmeans.cluster_centers_
no_inj_Speed_centroid = np.array(no_inj_Speed_kmeans_centroids)
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
       warnings.warn(
no_inj_RPM[['stamp', 'value']] = scaler.fit_transform(no_inj_RPM[['stamp', 'value']])
no_inj_RPM_kmeans = KMeans(n_clusters=2)
no_inj_RPM_kmeans.fit(no_inj_RPM[['stamp','value']])
no_inj_RPM['KMeans'] = no_inj_RPM_kmeans.labels_
no_inj_RPM_kmeans_centroids = no_inj_RPM_kmeans.cluster_centers_
no_inj_RPM_centroid = np.array(no_inj_RPM_kmeans_centroids)
     /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 1
      warnings.warn(
# printing the centroids
print("f_inj_Speed_centroid:",f_inj_Speed_centroid)
print("f_inj_RPM_centroid:",f_inj_RPM_centroid)
print("r_inj_Speed_centroid:",r_inj_Speed_centroid)
print("r_inj_RPM_centroid:",r_inj_RPM_centroid)
print("no_inj_Speed_centroid:",no_inj_Speed_centroid)
print("no_inj_RPM_centroid:",no_inj_RPM_centroid)
    f_inj_Speed_centroid: [[ 0.0147652     0.50089906]
     [-0.05743875 -1.94856666]]
    f_inj_RPM_centroid: [[-0.86156837 0.61731336]
     [ 0.8489947 -0.60830435]]
    [-0.77899649 -0.85721514]]
    r inj RPM centroid: [[ 0.10991432  0.64166953]
     [-0.26694906 -1.55842462]]
    no_inj_Speed_centroid: [[-1.185739
                                         -0.93419799]
     [ 0.54632192  0.43042593]]
    no_inj_RPM_centroid: [[ 1.1050571 -0.95013065]
     [-0.62699574 0.53909243]]
```

- Compare the graphs for speed

```
plt.scatter(x=f_inj_Speed['stamp'],y=f_inj_Speed['value'],c=f_inj_Speed['KMeans'],cmap='coolwarm')
plt.scatter(x=np.array(f_inj_Speed_centroid[:,0]),y=np.array(f_inj_Speed_centroid[:,1]),s = 60,marker="x")
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title(" fff injection")
plt.colorbar(label='Cluster')
plt.show()
```

```
fff injection

0.5

0.0

-0.5

-0.6

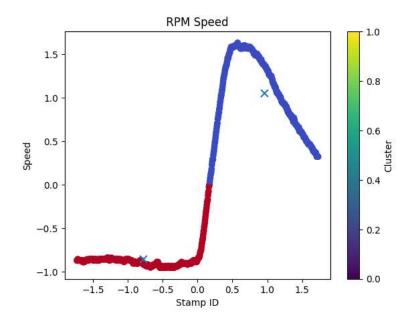
-0.6

-0.7

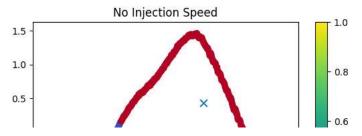
-0.4

-0.2
```

```
plt.scatter(x=r_inj_Speed['stamp'],y=r_inj_Speed['value'],c=r_inj_Speed['KMeans'],cmap='coolwarm')
plt.scatter(x=np.array(r_inj_Speed_centroid[:,0]),y=np.array(r_inj_Speed_centroid[:,1]),s = 60,marker="x")
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title(" RPM Speed ")
plt.colorbar(label='Cluster')
plt.show()
```

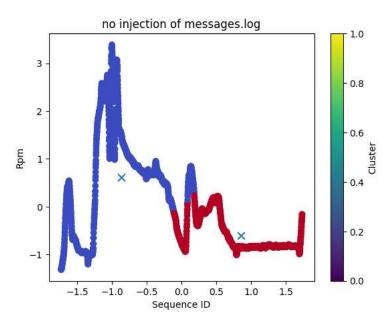


```
plt.scatter(x=no_inj_Speed['stamp'], y=no_inj_Speed['value'], c=no_inj_Speed['KMeans'], cmap='coolwarm')
plt.scatter(x=np.array(no_inj_Speed_centroid[:,0]), y=np.array(no_inj_Speed_centroid[:,1]), s=60, marker="x")
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title("No Injection Speed")
plt.colorbar(label='Cluster')
plt.show()
```



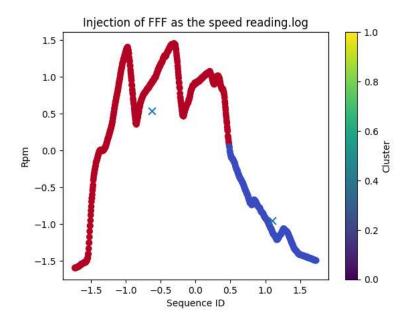
- Compare the plots for RPM

```
plt.scatter(x=f_inj_RPM['stamp'],y=f_inj_RPM['value'],c=f_inj_RPM['KMeans'],cmap='coolwarm')
plt.scatter(x=np.array(f_inj_RPM_centroid[:,0]),y=np.array(f_inj_RPM_centroid[:,1]),s = 60,marker="x")
plt.xlabel("Sequence ID")
plt.ylabel("Rpm")
plt.colorbar(label='Cluster')
plt.title("no injection of messages.log")
plt.show()
```



```
plt.scatter(x=r_inj_RPM['stamp'],y=r_inj_RPM['value'],c=r_inj_RPM['KMeans'],cmap='coolwarm')
plt.scatter(x=np.array(r_inj_RPM_centroid[:,0]),y=np.array(r_inj_RPM_centroid[:,1]),s = 60,marker="x")
plt.xlabel("Sequence ID")
plt.ylabel("Rpm")
plt.colorbar(label='Cluster')
plt.title("Injection of RPM readings.log")
plt.show()
```

```
plt.scatter(x=no_inj_RPM['stamp'],y=no_inj_RPM['value'],c=no_inj_RPM['KMeans'],cmap='coolwarm')
plt.scatter(x=np.array(no_inj_RPM_centroid[:,0]),y=np.array(no_inj_RPM_centroid[:,1]),s = 60,marker="x")
plt.xlabel("Sequence ID")
plt.ylabel("Rpm")
plt.colorbar(label='Cluster')
plt.title("Injection of FFF as the speed reading.log")
plt.show()
```



- TASK 3

import math

import seaborn as sns

from sklearn.ensemble import IsolationForest

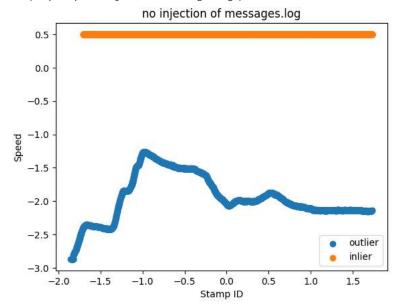
```
IF_f_inj_Speed = IsolationForest(contamination=0.5)
IF_f_inj_Speed.fit(f_inj_Speed[['value']])
IF_r_inj_Speed = IsolationForest(contamination=0.5)
IF_r_inj_Speed.fit(r_inj_Speed[['value']])
IF_no_inj_Speed = IsolationForest(contamination=0.5)
IF_no_inj_Speed.fit(no_inj_Speed[['value']])
IF_f_inj_RPM = IsolationForest(contamination=0.5)
IF_f_inj_RPM.fit(f_inj_RPM[['value']])
IF_r_inj_RPM = IsolationForest(contamination=0.5)
IF_r_inj_RPM.fit(r_inj_RPM[['value']])
IF_no_inj_RPM = IsolationForest(contamination=0.5)
IF_no_inj_RPM.fit(no_inj_RPM[['value']])
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
     /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but IsolationForest was f
       warnings.warn(
                IsolationForest
     IsolationForest(contamination=0.5)
```

```
f_inj_Speed['scores'] = IF_f_inj_Speed.decision_function(f_inj_Speed[['value']])
f_inj_Speed['anomaly'] = IF_f_inj_Speed.predict(f_inj_Speed[['value']])
r_inj_Speed['scores'] = IF_r_inj_Speed.decision_function(r_inj_Speed[['value']])
r_inj_Speed['anomaly'] = IF_r_inj_Speed.predict(r_inj_Speed[['value']])
no_inj_Speed['scores'] = IF_no_inj_Speed.decision_function(no_inj_Speed[['value']])
no_inj_Speed['anomaly'] = IF_no_inj_Speed.predict(no_inj_Speed[['value']])
f_inj_RPM['scores'] = IF_f_inj_RPM.decision_function(f_inj_RPM[['value']])
f_inj_RPM['scores'] = IF_r_inj_RPM.decision_function(r_inj_RPM[['value']])
r_inj_RPM['anomaly'] = IF_r_inj_RPM.predict(r_inj_RPM[['value']])
no_inj_RPM['scores'] = IF_no_inj_RPM.decision_function(no_inj_RPM[['value']])
no_inj_RPM['anomaly'] = IF_no_inj_RPM.predict(no_inj_RPM[['value']])
```

Comparing graph for Speed

```
# Plotting the graph NO injection as the speed reading.log and speed data
plt.scatter(data=f_inj_Speed.loc[f_inj_Speed['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=f_inj_Speed.loc[f_inj_Speed['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title("no injection of messages.log")
```

Text(0.5, 1.0, 'no injection of messages.log')

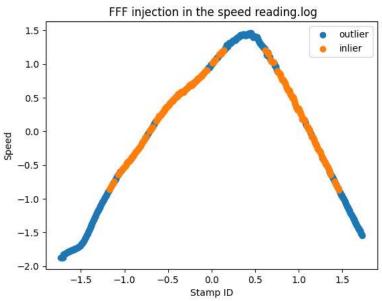


```
# Plotting the graph RPM injection as the speed reading.log and speed data
plt.scatter(data=r_inj_Speed.loc[r_inj_Speed['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=r_inj_Speed.loc[r_inj_Speed['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title("injection of RPM readings.log")
```

injection of RPM readings.log 1.5 - outlier inlier

```
# Plotting the graph FFF injection as the speed reading.log and speed data
plt.scatter(data=no_inj_Speed.loc[no_inj_Speed['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=no_inj_Speed.loc[no_inj_Speed['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title(" FFF injection in the speed reading.log")
```

Text(0.5, 1.0, ' FFF injection in the speed reading.log')



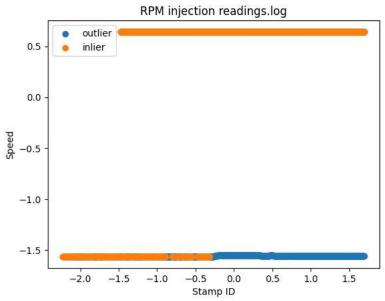
Comparing the graphs for RPM

```
# Plotting the graph for NO Injection
plt.scatter(data=f_inj_RPM.loc[f_inj_RPM['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=f_inj_RPM.loc[f_inj_RPM['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title(" no injection in messages.log")
```

no injection in messages.log

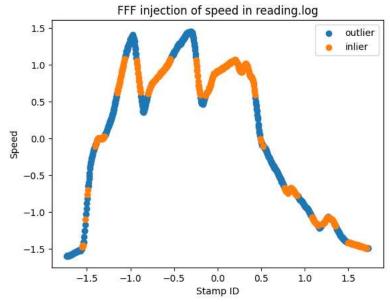
```
# Plotting the graph for RPM Injection
plt.scatter(data=r_inj_RPM.loc[r_inj_RPM['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=r_inj_RPM.loc[r_inj_RPM['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title("RPM injection readings.log")
```

Text(0.5, 1.0, 'RPM injection readings.log')



```
# Plotting the graph for FFF Injection
plt.scatter(data=no_inj_RPM.loc[no_inj_RPM['anomaly']==-1],x="stamp",y="value")
plt.scatter(data=no_inj_RPM.loc[no_inj_RPM['anomaly']==1],x="stamp",y="value")
plt.legend(["outlier","inlier"])
plt.xlabel("Stamp ID")
plt.ylabel("Speed")
plt.title("FFF injection of speed in reading.log")
```

Text(0.5, 1.0, 'FFF injection of speed in reading.log')



```
Requirement already satisfied: hmmlearn in /usr/local/lib/python3.10/dist-packages (0.3.0)
  Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.23.5)
  Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.2.2)
  Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.10/dist-packages (from hmmlearn) (1.11.3)
  Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmlearn) (1
  Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.22.0,>=0.16->hmmle
from hmmlearn.hmm import GaussianHMM
from sklearn.metrics import accuracy_score
def HMModel(data):
 hMmodel = GaussianHMM(n_components=2, random_state=42)
 dataframe_ = np.array(data["value"]).reshape(-1,1)
 hMmodel.fit(dataframe_)
 output = list(hMmodel.predict(dataframe_))
 print("Actual Value:", list(data['attack']))
 print("Predicted Value:", output)
 print("Accuracy:", accuracy_score(list(data['attack']), output))
# Hidden Markov Model for fff Injection, RPM
HMModel(f inj RPM)
  Accuracy: 0.5797379354426334
   4
# Hidden Markov Model for fff Injection, Speed
HMModel(f_inj_Speed)
  Accuracy: 0.0
   4 -
# Hidden Markov Model for RPM Injection, RPM
HMModel(r_inj_RPM)
  Accuracy: 0.0
# Hidden Markov Model for RPM Injection, Speed
HMModel(r_inj_Speed)
  Accuracy: 0.5367647058823529
# Hidden Markov Model for NO Injection RPM
HMModel(no_inj_RPM)
  Accuracy: 0.6093418259023354
   4
# Hidden Markov Model for NO Injection Speed
HMModel(no_inj_Speed)
  Accuracy: 0.7906647807637907
```

→ Task 5

1. What were the shortcomings/limitations of using unsupervised ML on this problem?

Unsupervised learning (K-means and Isolation Forest) has several limitations.

- a. In unsupervised learning, the labeled data is not present which makes model peformance evaluation challenging. b. The assumption that similar data points should belong to the same cluster may not be true in the complex datasets. c. In K-means, K value should be chosen very correctly. Else, if the k value is not chosen correctly the entire model may not give accurate results.
 - 2. What could be done to make the models perform better? Dimensionality reduction techniques like PCA can be applied before clustering to improve the accuracy of the model. Using multiple clustering algorithms and ensemble methods can also improve the results. Also, employing feature engineering techniques helps in creating more meaningful features.
 - 3. What do you suggest doing to solve the problem? I would suggest that refining data pre-processing methods are helpful, to ensure that the data is properly cleaned and prepared. Trying out various clustering algorithms and optimizing their parameters can help in achieving more accurate results. For unsupervised learning, visualizations and exploratory data analysis are important to understand the nature of the data. Finally, trying out the different models, tuning parameters, and comparing results with the other models can help in increasing the accuracy.

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.