Chronus Data Analysis with Prediction for Mixed cells

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os
```

In [2]:

```

# Read the data for cell A

# input_dir_dvv is the file location for dvv data folder
# input_dir_dv is the file location for dv data folder


input_dir_dvv = "/Users/jayashrijagannathan/Documents/chronus_project/An
input_dir_dv = "/Users/jayashrijagannathan/Documents/chronus_project/Ana

dvv_files = [f for f in os.listdir(input_dir_dvv)]
dv_files = [f for f in os.listdir(input_dir_dv)]

df_list = []  # initialize dataframes list

for dvv_f, dv_f in zip(dvv_files, dv_files):


    dvv_df = pd.read_csv(input_dir_dvv + "/" + dvv_f)
    dv_df = pd.read_csv(input_dir_dv + "/" + dv_f)


    # Here we combine the data of 106999 frequency dvv data with 2799999
    dvv_df = dvv_df[["real_time_106999","real_data_106999","imag_data_10
    dv_df = dv_df[["real_data_27999999","imag_data_27999999"]]

    # Scaling the dv data to match the dvv data
    dv_df = dv_df.apply(lambda x: x * 10)

    # Use the combined df data to proceed with analysis.
    combined_df = pd.concat([dvv_df,dv_df], axis=1, sort=False)
    df_list.append(combined_df)

concat_df = pd.concat(df_list, axis = 0)
```

In [3]:
```
concat_df.describe()
```

Out[3]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	422382.000000	422382.000000	422382.000000	422382.000000	422382.0
mean	5.120791	0.000078	0.000184	0.000429	0.0
std	2.693977	0.001900	0.003402	0.002649	0.0
min	0.009220	-0.008488	-0.013550	-0.011477	-0.0
25%	3.108961	-0.000701	-0.000916	-0.000718	-0.0
50%	5.176916	0.000006	0.000032	0.000013	0.0
75%	7.345117	0.000725	0.000991	0.000807	0.0
max	9.998788	0.008563	0.017257	0.019669	0.0

In [4]:
```
num_records = concat_df['real_time_106999'].count()
print(num_records)
```

```
422382
```

In [5]:
```
concat_df['cell_type'] = [0 for x in range(num_records)]
concat_df.describe()
```

Out[5]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	422382.000000	422382.000000	422382.000000	422382.000000	422382.0
mean	5.120791	0.000078	0.000184	0.000429	0.0
std	2.693977	0.001900	0.003402	0.002649	0.0
min	0.009220	-0.008488	-0.013550	-0.011477	-0.0
25%	3.108961	-0.000701	-0.000916	-0.000718	-0.0
50%	5.176916	0.000006	0.000032	0.000013	0.0
75%	7.345117	0.000725	0.000991	0.000807	0.0
max	9.998788	0.008563	0.017257	0.019669	0.0

In [6]:

```
# Read the data for cell B

# input_dir_dvv is the file location for dvv data folder
# input_dir_dv is the file location for dv data folder


input_dir_dvv = "/Users/jayashrijagannathan/Documents/chronus_project/An
input_dir_dv = "/Users/jayashrijagannathan/Documents/chronus_project/Ana

dvv_files = [f for f in os.listdir(input_dir_dvv)]
dv_files = [f for f in os.listdir(input_dir_dv)]

df_list = []  # initialize dataframes list

for dvv_f, dv_f in zip(dvv_files, dv_files):


    dvv_df = pd.read_csv(input_dir_dvv + "/" + dvv_f)
    dv_df = pd.read_csv(input_dir_dv + "/" + dv_f)


    # Here we combine the data of 106999 frequency dvv data with 2799999
    dvv_df = dvv_df[["real_time_106999","real_data_106999","imag_data_10
    dv_df = dv_df[["real_data_27999999","imag_data_27999999"]]

    # Scaling the dv data to match the dvv data
    dv_df = dv_df.apply(lambda x: x * 10)

    # Use the combined df data to proceed with analysis.
    combined_df = pd.concat([dvv_df,dv_df], axis=1, sort=False)
    df_list.append(combined_df)

concat_df1 = pd.concat(df_list, axis = 0)
```

In [7]:

```
concat_df1.describe()
```

Out[7]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	389250.000000	389250.000000	389250.000000	389250.000000	389250.0
mean	5.070524	0.000189	0.000651	0.001054	0.0
std	2.815230	0.004440	0.007685	0.005316	0.0
min	0.017045	-0.029131	-0.044101	-0.017850	-0.0
25%	3.126056	-0.000842	-0.000998	-0.000743	-0.0
50%	4.821506	-0.000002	0.000016	-0.000029	0.0
75%	7.451487	0.000831	0.001055	0.000788	0.0
max	9.999300	0.033728	0.062087	0.033611	0.0

In [8]:

```
num_records = concat_df1['real_time_106999'].count()
print(num_records)
```

```
389250
```

In [9]:

```
concat_df1['cell_type'] = [1 for x in range(num_records)]
concat_df1.describe()
```

Out[9]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	389250.000000	389250.000000	389250.000000	389250.000000	389250.0
mean	5.070524	0.000189	0.000651	0.001054	0.0
std	2.815230	0.004440	0.007685	0.005316	0.0
min	0.017045	-0.029131	-0.044101	-0.017850	-0.0
25%	3.126056	-0.000842	-0.000998	-0.000743	-0.0
50%	4.821506	-0.000002	0.000016	-0.000029	0.0
75%	7.451487	0.000831	0.001055	0.000788	0.0
max	9.999300	0.033728	0.062087	0.033611	0.0

In [10]:

```
# Read the data for cell C

# input_dir_dvv is the file location for dvv data folder
# input_dir_dv is the file location for dv data folder


input_dir_dvv = "/Users/jayashrijagannathan/Documents/chronus_project/An
input_dir_dv = "/Users/jayashrijagannathan/Documents/chronus_project/Ana

dvv_files = [f for f in os.listdir(input_dir_dvv)]
dv_files = [f for f in os.listdir(input_dir_dv)]

df_list = []  # initialize dataframes list

for dvv_f, dv_f in zip(dvv_files, dv_files):


    dvv_df = pd.read_csv(input_dir_dvv + "/" + dvv_f)
    dv_df = pd.read_csv(input_dir_dv + "/" + dv_f)


    # Here we combine the data of 106999 frequency dvv data with 2799999
    dvv_df = dvv_df[["real_time_106999","real_data_106999","imag_data_10
    dv_df = dv_df[["real_data_27999999","imag_data_27999999"]]

    # Scaling the dv data to match the dvv data
    dv_df = dv_df.apply(lambda x: x * 10)

    # Use the combined df data to proceed with analysis.
    combined_df = pd.concat([dvv_df,dv_df], axis=1, sort=False)
    df_list.append(combined_df)

concat_df2 = pd.concat(df_list, axis = 0)
```

In [11]:

```
concat_df2.describe()
```

Out[11]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	289621.000000	289621.000000	289621.000000	289621.000000	289621.0
mean	5.187166	0.000329	0.000617	0.001723	0.0
std	2.801033	0.007311	0.011412	0.009931	0.0
min	0.012892	-0.043792	-0.067887	-0.039366	-0.0
25%	2.402103	-0.000721	-0.000991	-0.000886	-0.0
50%	5.394943	0.000015	-0.000007	-0.000117	0.0
75%	7.749698	0.000770	0.001012	0.000717	0.0
max	9.997930	0.052416	0.080439	0.073876	0.0

In [12]:

```
num_records = concat_df2['real_time_106999'].count()
print(num_records)
```

```
289621
```

In [13]:

```
concat_df2['cell_type'] = [2 for x in range(num_records)]
concat_df2.describe()
```

Out[13]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	289621.000000	289621.000000	289621.000000	289621.000000	289621.0
mean	5.187166	0.000329	0.000617	0.001723	0.0
std	2.801033	0.007311	0.011412	0.009931	0.0
min	0.012892	-0.043792	-0.067887	-0.039366	-0.0
25%	2.402103	-0.000721	-0.000991	-0.000886	-0.0
50%	5.394943	0.000015	-0.000007	-0.000117	0.0
75%	7.749698	0.000770	0.001012	0.000717	0.0
max	9.997930	0.052416	0.080439	0.073876	0.0

In [14]:

```
frames = [concat_df, concat_df1, concat_df2]
final_df = pd.concat(frames)
```

In [15]:
```
final_df.describe()
```

Out[15]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_2799
count	1.101253e+06	1.101253e+06	1.101253e+06	1.101253e+06	1.10125
mean	5.120479e+00	1.832169e-04	4.627595e-04	9.900060e-04	1.61581
std	2.765931e+00	4.734893e-03	7.720983e-03	6.235154e-03	3.95202
min	9.220000e-03	-4.379175e-02	-6.788670e-02	-3.936649e-02	-4.11156
25%	2.946435e+00	-7.537055e-04	-9.647404e-04	-7.703195e-04	-6.92662
50%	5.077063e+00	6.025665e-06	1.718565e-05	-3.546471e-05	1.92559
75%	7.489495e+00	7.731317e-04	1.018462e-03	7.795763e-04	7.71254
max	9.999300e+00	5.241596e-02	8.043873e-02	7.387561e-02	4.21115

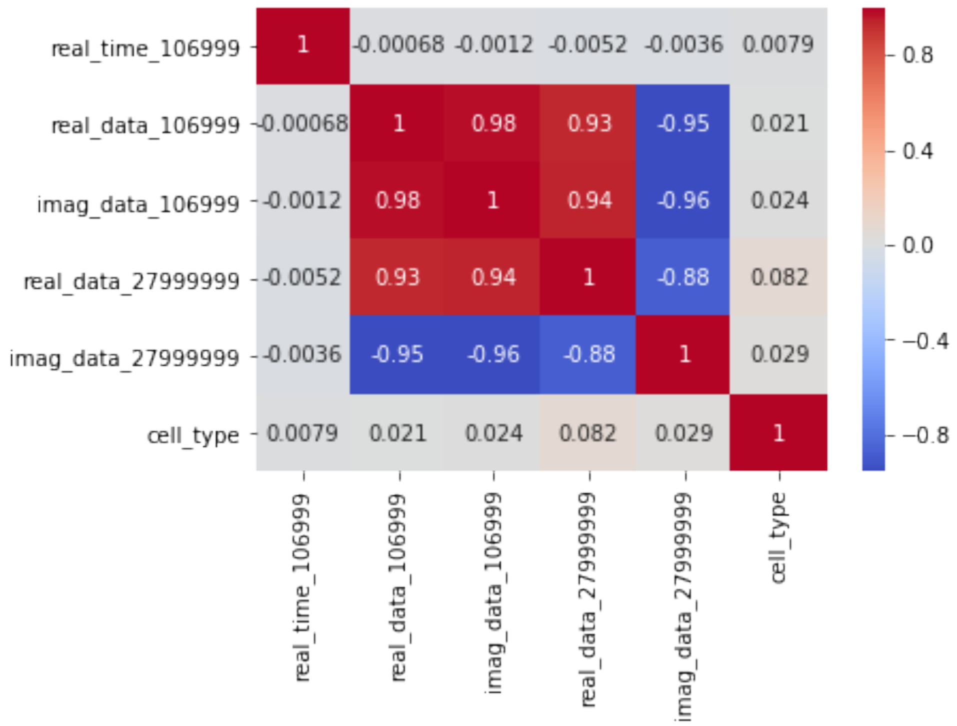
In [16]:
```
# Get the correlation matrix
corrMatrix = final_df.corr()
print(corrMatrix)
```

```
                    real_time_106999  real_data_106999  imag_data_1069
99  \
real_time_106999            1.000000         -0.000679         -0.0012
37
real_data_106999           -0.000679          1.000000          0.9833
39
imag_data_106999           -0.001237          0.983339          1.0000
00
real_data_27999999         -0.005224          0.933277          0.9435
11
imag_data_27999999         -0.003626         -0.953801         -0.9575
33
cell_type                   0.007922          0.020980          0.0240
67

                    real_data_27999999  imag_data_27999999  cell_type
real_time_106999             -0.005224           -0.003626   0.007922
real_data_106999              0.933277           -0.953801   0.020980
imag_data_106999              0.943511           -0.957533   0.024067
real_data_27999999            1.000000           -0.878623   0.082289
imag_data_27999999           -0.878623            1.000000   0.028896
cell_type                     0.082289            0.028896   1.000000
```

In [17]:
```
sns.heatmap(corrMatrix, annot = True, cmap = 'coolwarm')
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a18f16b70>
```



In [18]:
```
final_df.head()
```

Out[18]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_2799999
0	9.797785	-0.001080	0.000197	-0.000086	-0.00021
1	9.797788	-0.001103	0.000159	-0.000018	-0.00027
2	9.797790	-0.001122	0.000124	0.000066	-0.00034
3	9.797793	-0.001135	0.000093	0.000161	-0.00041
4	9.797795	-0.001143	0.000070	0.000259	-0.00048

In [19]:

```
# Find out if there is missing data
print(final_df.isnull().sum())
```

```
real_time_106999      0
real_data_106999      0
imag_data_106999      0
real_data_27999999    0
imag_data_27999999    0
cell_type             0
dtype: int64
```

Observation: There is no missing data

In [20]:

```
final_df.columns
```

Out[20]:

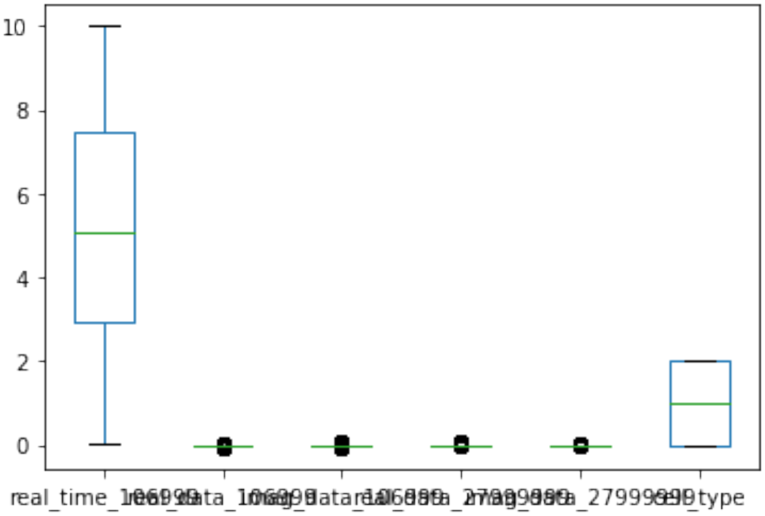
```
Index(['real_time_106999', 'real_data_106999', 'imag_data_106999',
       'real_data_27999999', 'imag_data_27999999', 'cell_type'],
      dtype='object')
```

In [21]:

```
final_df.plot.box()
```

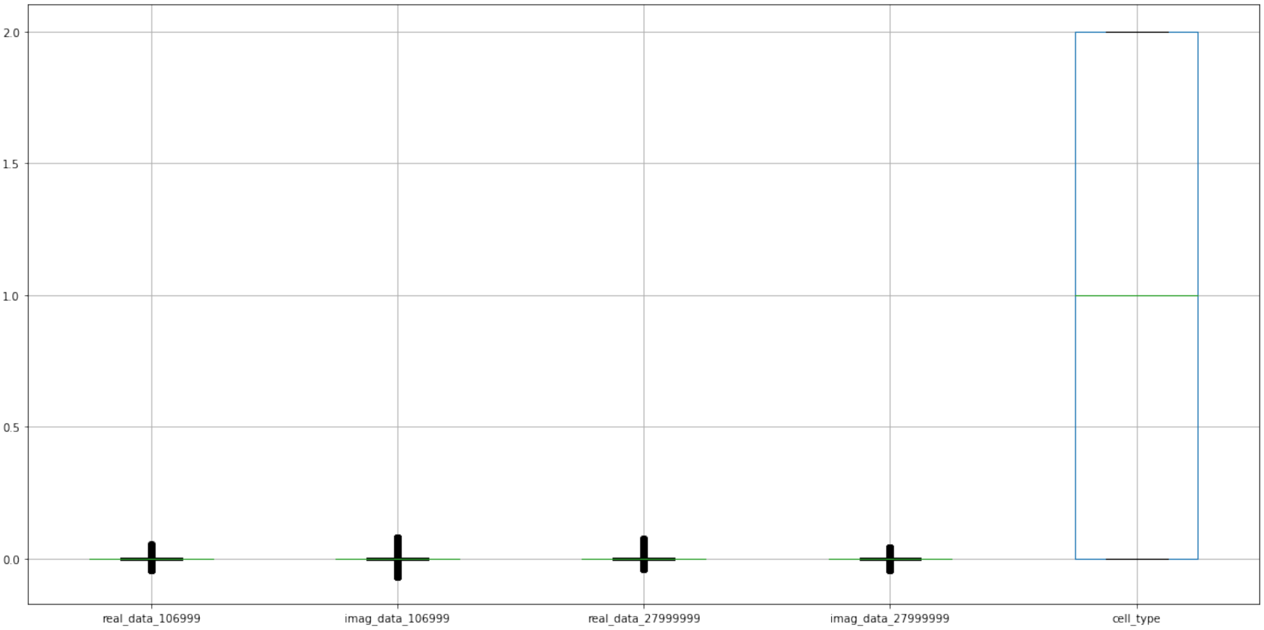
Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1a6a7fd0>
```



In [22]:
```
# Drop real_time_106999

boxplot = final_df.boxplot(column = ['real_data_106999', 'imag_data_1069
                                     'imag_data_27999999', 'cell_type'], figsize = (20, 1
```



In [23]:
```
#Preprocessing for data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [24]:
```
# Do the scaler fit for all variables except cell_type (i.e. the depende
# and real_time_106999
final_df = final_df[['real_data_106999', 'imag_data_106999','real_data_2
                     'imag_data_27999999', 'cell_type']]
```

In [25]:
```
final_df.columns
```

Out[25]:
```
Index(['real_data_106999', 'imag_data_106999', 'real_data_27999999',
       'imag_data_27999999', 'cell_type'],
      dtype='object')
```

In [26]:
```
# Separate the pandas dataframe into input and output components
X = final_df[['real_data_106999', 'imag_data_106999', 'real_data_2799999
Y = final_df['cell_type']
```

```
In [27]: scaler.fit(X)

Out[27]: StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
In [28]: scaled_features = scaler.transform(final_df[['real_data_106999', 'imag_d
                                                      'real_data_27999999','imag_
```

```
In [29]: df_feat=pd.DataFrame(scaled_features, columns = final_df.columns[:-1] )
         df_feat.head()

Out[29]:
```

	real_data_106999	imag_data_106999	real_data_27999999	imag_data_27999999
0	-0.266839	-0.034465	-0.172628	-0.095304
1	-0.271733	-0.039367	-0.161736	-0.110536
2	-0.275579	-0.043911	-0.148214	-0.127548
3	-0.278400	-0.047841	-0.132992	-0.145318
4	-0.280180	-0.050887	-0.117244	-0.162631

Our data is fitted and scaled and now it is ready

```
In [30]: # Define X and Y
         X = df_feat
         y = final_df['cell_type']
```

```
In [31]: # import train test split and metrics for evaluation
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import confusion_matrix, classification_report
```

```
In [32]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

```
In [33]: # Decision Tree Classifier
         from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier()
```

In [34]:
```
classifier.fit(X_train, y_train)
```

Out[34]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=N
one,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [35]:
```
y_pred = classifier.predict(X_test)
```

In [36]:
```
#Summary of predictions made by the Decision Tree Classifier
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.63      0.66      0.64    126569
           1       0.57      0.55      0.56    116658
           2       0.51      0.49      0.50     87149

    accuracy                           0.58    330376
   macro avg       0.57      0.57      0.57    330376
weighted avg       0.58      0.58      0.58    330376
```

In [37]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[83018 24985 18566]
 [28756 64671 23231]
 [21026 23304 42819]]
```

In [38]:
```
from sklearn.metrics import accuracy_score
print('accuracy = ', accuracy_score(y_pred, y_test))
```

```
accuracy =  0.5766399496331452
```

Prediction for Mixed Cell Types

In [39]:

```
# Read the data for cell Mixed_Cells_Type_1

# input_dir_dvv is the file location for dvv data folder
# input_dir_dv is the file location for dv data folder


input_dir_dvv = "/Users/jayashrijagannathan/Documents/chronus_project/An
input_dir_dv = "/Users/jayashrijagannathan/Documents/chronus_project/Ana

dvv_files = [f for f in os.listdir(input_dir_dvv)]
dv_files = [f for f in os.listdir(input_dir_dv)]

df_list = []  # initialize dataframes list

cell_num_list = [] # initialize the cell ID number list from Cell_<cell_

for dvv_f, dv_f in zip(dvv_files, dv_files):

    cell_num = int(dvv_f.split('.')[0].split('_')[1]) # cell ID from csv
    cell_num_list.append(cell_num)

    dvv_df = pd.read_csv(input_dir_dvv + "/" + dvv_f)
    dv_df = pd.read_csv(input_dir_dv + "/" + dv_f)


    # Here we combine the data of 106999 frequency dvv data with 2799999
    dvv_df = dvv_df[["real_time_106999","real_data_106999","imag_data_10
    dv_df = dv_df[["real_data_27999999","imag_data_27999999"]]

    # Scaling the dv data to match the dvv data
    dv_df = dv_df.apply(lambda x: x * 10)

    # Use the combined df data to proceed with analysis.
    combined_df = pd.concat([dvv_df,dv_df], axis=1, sort=False)

    #Add a new column to combined_df called cell_num to identify the spe
    num_records = combined_df["real_time_106999"].count()
    combined_df['cell_id'] = [cell_num for x in range(num_records)]

    df_list.append(combined_df)

concat_mixed_df1 = pd.concat(df_list, axis = 0)
```

In [40]:
```
concat_mixed_df1.describe()
```

Out[40]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	909420.000000	909420.000000	909420.000000	909420.000000	9.09420
mean	4.828389	0.000087	0.000345	0.000783	1.20319
std	2.834325	0.004388	0.007431	0.005996	3.84528
min	0.012960	-0.038856	-0.062456	-0.030852	-3.60458
25%	2.344528	-0.000741	-0.000900	-0.000749	-6.72135
50%	4.556157	-0.000018	-0.000016	-0.000053	3.89607
75%	7.463443	0.000705	0.000897	0.000699	6.95663
max	9.988067	0.040986	0.071008	0.061863	3.48860

In [41]:
```
concat_mixed_df1.head()
```

Out[41]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_2799999
0	5.479610	0.000987	-0.000073	0.001163	-0.00121
1	5.479613	0.000997	-0.000042	0.001204	-0.00118
2	5.479615	0.001006	-0.000013	0.001232	-0.00112
3	5.479618	0.001015	0.000015	0.001242	-0.00103
4	5.479620	0.001024	0.000042	0.001233	-0.00092

In [42]:
```
concat_mixed_df1.tail()
```

Out[42]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_27999
292	9.695752	0.000769	-0.000592	0.000789	-0.000
293	9.695755	0.000730	-0.000636	0.000840	-0.000
294	9.695757	0.000676	-0.000664	0.000869	-0.000
295	9.695760	0.000605	-0.000673	0.000873	-0.000
296	9.695763	0.000519	-0.000667	0.000853	-0.000

In [43]:
```
#Number of cell IDs
print(len(cell_num_list))
```

```
2314
```

In [44]:
```
# Number of cell IDs from the dataframe concat_mixed_df1
concat_mixed_df1['cell_id'].nunique()
```

Out[44]:
```
2314
```

In [45]:
```
# Get columns of concat_mixed_df1
concat_mixed_df1.columns
```

Out[45]:
```
Index(['real_time_106999', 'real_data_106999', 'imag_data_106999',
       'real_data_27999999', 'imag_data_27999999', 'cell_id'],
      dtype='object')
```

In [46]:
```
# Define the X variables for Mixed Cell Type 1 -- don't consider real_ti
X_mixed_1 = concat_mixed_df1[['real_data_106999', 'imag_data_106999',
                              'real_data_27999999', 'imag_data_27999999'
```

In [47]:
```
X_mixed_1.head()
```

Out[47]:

	real_data_106999	imag_data_106999	real_data_27999999	imag_data_27999999
0	0.000987	-0.000073	0.001163	-0.001215
1	0.000997	-0.000042	0.001204	-0.001189
2	0.001006	-0.000013	0.001232	-0.001126
3	0.001015	0.000015	0.001242	-0.001034
4	0.001024	0.000042	0.001233	-0.000924

In [48]:
```
# Use Standard Scaler on X_mixed_1
scaler.fit(X_mixed_1)
```

Out[48]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [49]:
```
scaled_features = scaler.transform(concat_mixed_df1[['real_data_106999',
                                                      'imag_data_106999',
                                                      'real_data_27999999
                                                      'imag_data_27999999
```

In [50]:
```
cols = ['real_data_106999', 'imag_data_106999','real_data_27999999',
                         'imag_data_27999999']

df_feat = pd.DataFrame(scaled_features, columns = cols)
```

In [51]:
```
# Prediction using Decision Tree model
X_mixed1 = df_feat
y_mixed_pred1 = classifier.predict(X_mixed1)
```

In [52]:
```
type(y_mixed_pred1)
```

Out[52]:
```
numpy.ndarray
```

In [53]:
```
#Convert y_mixed_pred1 (numpy.ndarray) to a column in pandas dataframe c
concat_mixed_df1['y_mixed_pred1'] = y_mixed_pred1
print(concat_mixed_df1.head())
```

```
   real_time_106999  real_data_106999  imag_data_106999  real_data_279
99999  \
0          5.479610          0.000987         -0.000073            0.0
01163
1          5.479613          0.000997         -0.000042            0.0
01204
2          5.479615          0.001006         -0.000013            0.0
01232
3          5.479618          0.001015          0.000015            0.0
01242
4          5.479620          0.001024          0.000042            0.0
01233

   imag_data_27999999  cell_id  y_mixed_pred1
0           -0.001215     1614              0
1           -0.001189     1614              0
2           -0.001126     1614              0
3           -0.001034     1614              0
4           -0.000924     1614              0
```

In [54]:
```
concat_mixed_df1['y_mixed_pred1'].value_counts()
```

Out[54]:
```
1    339890
0    332082
2    237448
Name: y_mixed_pred1, dtype: int64
```

In [55]:

```
# Now we have for the specific cell ID the corresponding y_mixed_pred1 v
# We will find the highest frequency value for each cell ID

def find_cell_type(alist):
    # Count the number of 0, 1, and 2
    the_dict = {}
    for item in alist:
        if item not in the_dict:
            the_dict[item] = 1
        else:
            the_dict[item] += 1
    cell_type = max(the_dict, key=the_dict.get)
    return(cell_type)
```

In [56]:

```
uniq_cell_id_list = list(concat_mixed_df1['cell_id'].unique())
print(len(uniq_cell_id_list))
```

```
2314
```

In [57]:

```
# Process each cell-- classify it into type 0,1,or 2 for cell A,B,C resp
cell_type_pred_list = []
for cell in uniq_cell_id_list:
    df1 = concat_mixed_df1[concat_mixed_df1['cell_id'] == cell]
    y_pred_list = df1['y_mixed_pred1'].tolist()
    cell_type_pred = find_cell_type(y_pred_list)
    cell_type_pred_list.append(cell_type_pred)
```

In [58]:

```
# Put cell_id and predicted cell type in a result dataframe
result_df1 = pd.DataFrame({'cell_id': uniq_cell_id_list,
                           'cell_type_predicted': cell_type_pred_list})
```

In [59]:

```
result_df1.count()
```

Out[59]:

```
cell_id                  2314
cell_type_predicted      2314
dtype: int64
```

In [60]:
```
result_df1.head()
```

Out[60]:

	cell_id	cell_type_predicted
0	1614	0
1	1172	2
2	901	2
3	915	1
4	1166	2

In [61]:
```
result_df1.to_excel('mixed_cell_type_1_result.xls')
```

In [62]:
```
result_df1['cell_type_predicted'].value_counts()
```

Out[62]:
```
0    1126
1     782
2     406
Name: cell_type_predicted, dtype: int64
```

In [63]:
```
result_df1['cell_type_predicted'].value_counts(normalize = True) * 100
```

Out[63]:
```
0    48.660328
1    33.794296
2    17.545376
Name: cell_type_predicted, dtype: float64
```

Mixed Cell Type 2

In [64]:

```
# Read the data for cell Mixed_Cells_Type_2

# input_dir_dvv is the file location for dvv data folder
# input_dir_dv is the file location for dv data folder


input_dir_dvv = "/Users/jayashrijagannathan/Documents/chronus_project/An
input_dir_dv = "/Users/jayashrijagannathan/Documents/chronus_project/Ana

dvv_files = [f for f in os.listdir(input_dir_dvv)]
dv_files = [f for f in os.listdir(input_dir_dv)]

df_list = []  # initialize dataframes list

cell_num_list = [] # initialize the cell ID number list from Cell_<cell_

for dvv_f, dv_f in zip(dvv_files, dv_files):

    cell_num = int(dvv_f.split('.')[0].split('_')[1]) # cell ID from csv
    cell_num_list.append(cell_num)

    dvv_df = pd.read_csv(input_dir_dvv + "/" + dvv_f)
    dv_df = pd.read_csv(input_dir_dv + "/" + dv_f)


    # Here we combine the data of 106999 frequency dvv data with 2799999
    dvv_df = dvv_df[["real_time_106999","real_data_106999","imag_data_10
    dv_df = dv_df[["real_data_27999999","imag_data_27999999"]]

    # Scaling the dv data to match the dvv data
    dv_df = dv_df.apply(lambda x: x * 10)

    # Use the combined df data to proceed with analysis.
    combined_df = pd.concat([dvv_df,dv_df], axis=1, sort=False)

    #Add a new column to combined_df called cell_num to identify the spe
    num_records = combined_df["real_time_106999"].count()
    combined_df['cell_id'] = [cell_num for x in range(num_records)]

    df_list.append(combined_df)

concat_mixed_df2 = pd.concat(df_list, axis = 0)
```

In [65]:
```
concat_mixed_df2.describe()
```

Out[65]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
count	2.163388e+06	2.163388e+06	2.163388e+06	2.163388e+06	2.16338
mean	4.988365e+00	-1.829240e-05	-3.428306e-05	4.711581e-04	1.39269
std	2.869387e+00	2.375823e-03	4.086100e-03	3.771979e-03	1.94537
min	2.425000e-04	-7.978595e-02	-1.382508e-01	-1.707221e-01	-4.18500
25%	2.497960e+00	-8.001132e-04	-1.338566e-03	-9.695339e-04	-7.84435
50%	4.904051e+00	6.451598e-06	7.429651e-06	-1.356636e-04	-1.33532
75%	7.428271e+00	8.013492e-04	1.367280e-03	9.023179e-04	8.27109
max	9.999300e+00	7.121290e-02	1.201685e-01	2.845747e-01	1.04440

In [66]:
```
concat_mixed_df2.tail()
```

Out[66]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_27999
172	2.694310	-0.001265	-0.000101	0.000506	-0.000
173	2.694313	-0.001244	-0.000052	0.000496	-0.000
174	2.694315	-0.001200	-0.000006	0.000489	-0.000
175	2.694318	-0.001138	0.000037	0.000482	-0.000
176	2.694320	-0.001064	0.000082	0.000476	-0.000

In [67]:
```
# Number of cell IDs
print(len(cell_num_list))
```

```
8661
```

In [68]:
```
# Verify -- Number of cell IDs from dataframe concat_mixed_df2
concat_mixed_df2['cell_id'].nunique()
```

Out[68]:
```
8661
```

In [69]:
```
# Get the columns of concat_mixed_df2
concat_mixed_df2.columns
```

Out[69]:
```
Index(['real_time_106999', 'real_data_106999', 'imag_data_106999',
       'real_data_27999999', 'imag_data_27999999', 'cell_id'],
      dtype='object')
```

In [70]:
```
# Define X variables
X_mixed_2 = concat_mixed_df2[['real_data_106999', 'imag_data_106999',
                              'real_data_27999999', 'imag_data_27999999'
```

In [71]:
```
X_mixed_2.head()
```

Out[71]:

	real_data_106999	imag_data_106999	real_data_27999999	imag_data_27999999
0	-0.004818	-0.007981	-0.003738	0.003641
1	-0.004883	-0.008234	-0.003872	0.003829
2	-0.004910	-0.008449	-0.003989	0.004000
3	-0.004905	-0.008627	-0.004088	0.004150
4	-0.004873	-0.008772	-0.004166	0.004275

In [72]:
```
# Use Standard Scaler on X_mixed_2
scaler.fit(X_mixed_2)
```

Out[72]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [73]:
```
scaled_features = scaler.transform(concat_mixed_df2[['real_data_106999',
                                                     'imag_data_106999',
                                                     'real_data_27999999
                                                     'imag_data_27999999
```

In [74]:
```
cols = ['real_data_106999', 'imag_data_106999',
        'real_data_27999999', 'imag_data_27999999']
df_feat = pd.DataFrame(scaled_features, columns = cols)
```

In [75]:
```
# Prediction using Decision Tree model
X_mixed2 = df_feat
y_mixed_pred2 = classifier.predict(X_mixed2)
```

In [76]:

```
# Convert y_mixed_pred2 to a column in concat_mixed_df2
concat_mixed_df2['y_mixed_pred2'] = y_mixed_pred2
print(concat_mixed_df2.head())
```

```
   real_time_106999  real_data_106999  imag_data_106999  real_data_279
99999  \
0          6.594275         -0.004818         -0.007981           -0.0
03738
1          6.594278         -0.004883         -0.008234           -0.0
03872
2          6.594280         -0.004910         -0.008449           -0.0
03989
3          6.594282         -0.004905         -0.008627           -0.0
04088
4          6.594285         -0.004873         -0.008772           -0.0
04166

   imag_data_27999999  cell_id  y_mixed_pred2
0            0.003641     5472              1
1            0.003829     5472              1
2            0.004000     5472              1
3            0.004150     5472              1
4            0.004275     5472              1
```

In [77]:

```
concat_mixed_df2['y_mixed_pred2'].value_counts()
```

Out[77]:

```
1    941567
2    628922
0    592899
Name: y_mixed_pred2, dtype: int64
```

In [78]:

```
uniq_cell_id_list = list(concat_mixed_df2['cell_id'].unique())
print(len(uniq_cell_id_list))
```

```
8661
```

In [79]:

```
# Process each cell - classify to type 0, 1 or 2 for cell A, B, C respec
cell_type_pred_list = []
for cell in uniq_cell_id_list:
    df2 = concat_mixed_df2[concat_mixed_df2['cell_id'] == cell]
    y_pred_list = df2['y_mixed_pred2'].tolist()
    cell_type_pred = find_cell_type(y_pred_list)
    cell_type_pred_list.append(cell_type_pred)
```

In [80]:

```
# Put cell_id and predicted cell type in a result dataframe
result_df2 = pd.DataFrame({'cell_id': uniq_cell_id_list,
                           'cell_type_predicted': cell_type_pred_list})

result_df2.count()
```

Out[80]:

```
cell_id                8661
cell_type_predicted    8661
dtype: int64
```

In [81]:

```
result_df2.head()
```

Out[81]:

	cell_id	cell_type_predicted
0	5472	1
1	3003	1
2	8156	0
3	7265	1
4	1614	1

In [83]:

```
result_df2.to_excel('mixed_cell_type_2_result.xls')
```

In [84]:

```
result_df2['cell_type_predicted'].value_counts()
```

Out[84]:

```
1    6839
2    1238
0     584
Name: cell_type_predicted, dtype: int64
```

In [86]:

```
result_df2['cell_type_predicted'].value_counts(normalize = True) * 100
```

Out[86]:

```
1    78.963168
2    14.293961
0     6.742870
Name: cell_type_predicted, dtype: float64
```

In [ ]:

```

```