Chronus Data Analysis

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os
```

In [7]:
```

# 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 [8]:
```
concat_df.describe()
```

Out[8]:

	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 [11]:
```
num_records = concat_df['real_time_106999'].count()
print(num_records)
```

422382

In [12]:
```
concat_df['cell_type'] = [0 for x in range(num_records)]
concat_df.describe()
```

Out[12]:

	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 [13]:
```
# 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 [14]:
```
concat_df1.describe()
```

Out[14]:

	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 [16]:
```
num_records = concat_df1['real_time_106999'].count()
print(num_records)
```

```
389250
```

In [17]:
```
concat_df1['cell_type'] = [1 for x in range(num_records)]
concat_df1.describe()
```

Out[17]:

	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 [18]:
```
# 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 [19]:
```
concat_df2.describe()
```

Out[19]:

	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 [20]:
```
num_records = concat_df2['real_time_106999'].count()
print(num_records)
```

```
289621
```

In [24]:
```
concat_df2['cell_type'] = [2 for x in range(num_records)]
concat_df2.describe()
```

Out[24]:

	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 [25]:
```
frames = [concat_df, concat_df1, concat_df2]
final_df = pd.concat(frames)
```

In [26]:
```
final_df.describe()
```

Out[26]:

	real_time_106999	real_data_106999	imag_data_106999	real_data_27999999	imag_data_279
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

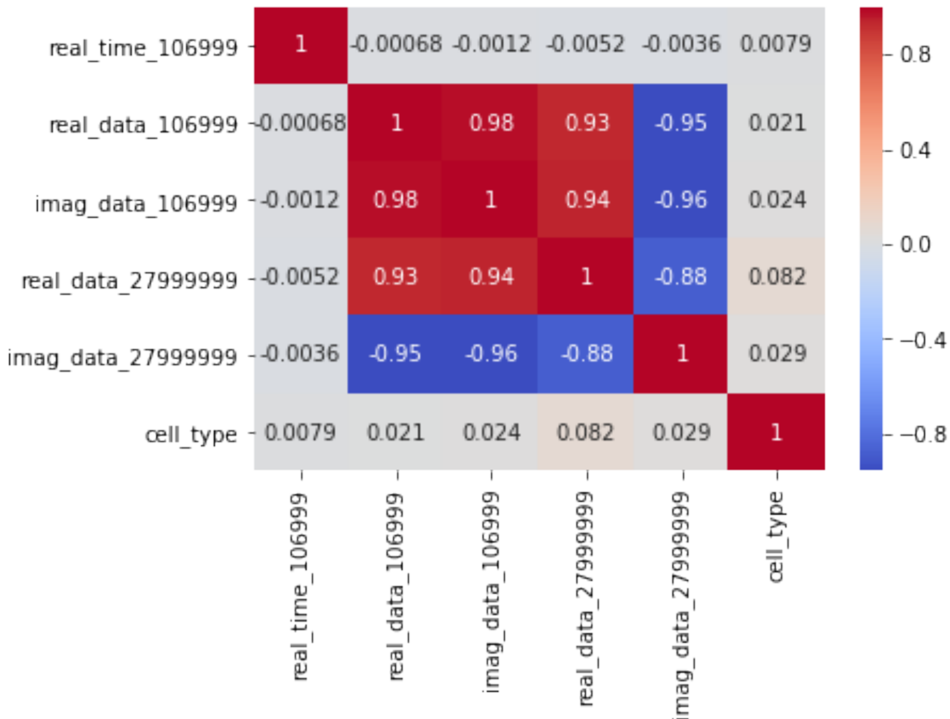
In [31]:
```
# Get the correlation matrix
corrMatrix = final_df.corr()
print(corrMatrix)
```

```
                    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 [32]:
```
sns.heatmap(corrMatrix, annot = True, cmap = 'coolwarm')
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a333b8160>
```



In [33]:
```
final_df.head()
```

Out[33]:

	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 [34]:
```
# 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
```

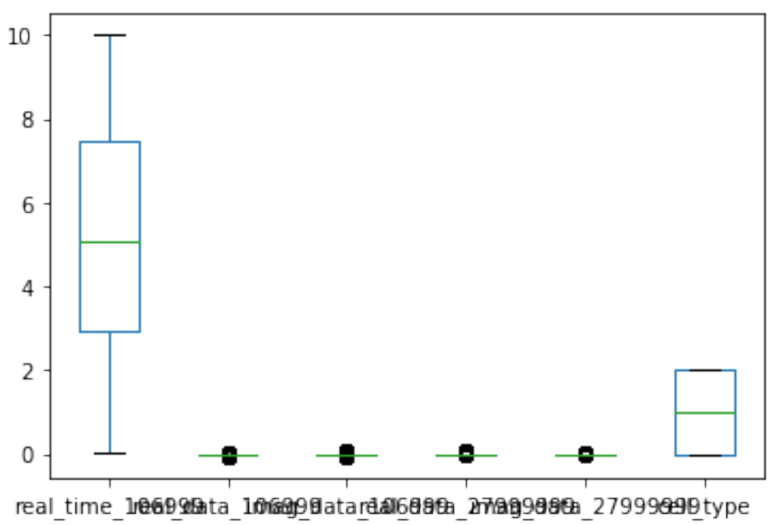
Observation: There is no missing data

In [37]:
```
final_df.columns
```

Out[37]:
```
Index(['real_time_106999', 'real_data_106999', 'imag_data_106999',
       'real_data_27999999', 'imag_data_27999999', 'cell_type'],
      dtype='object')
```

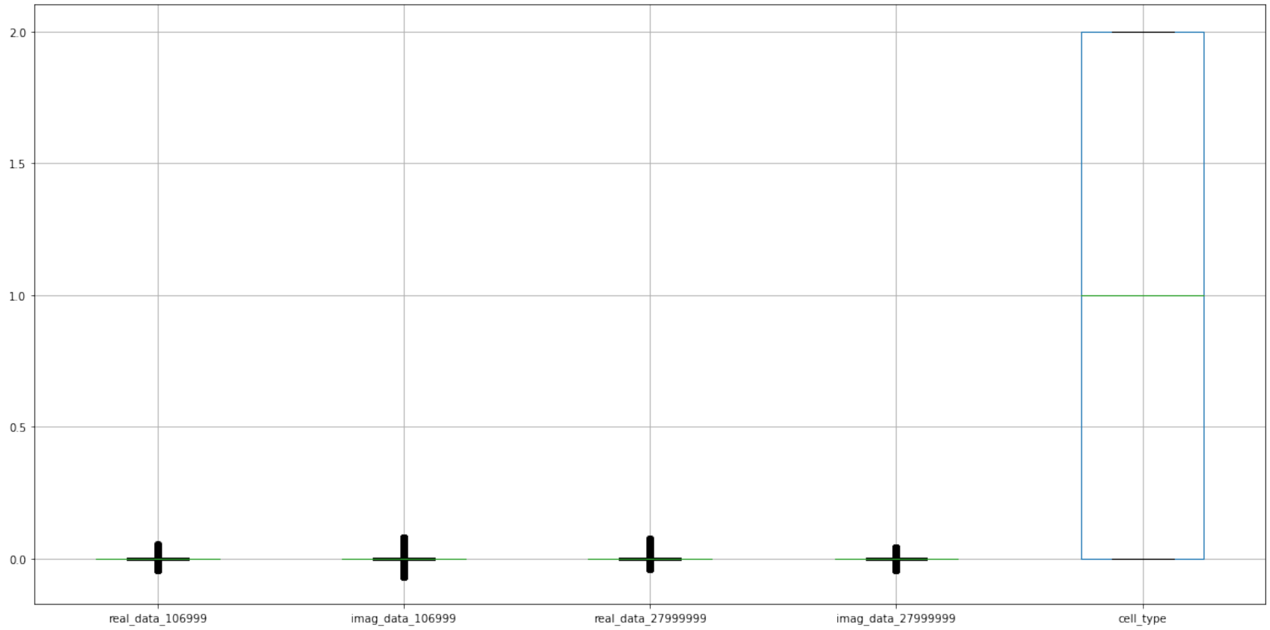
In [42]:
```
final_df.plot.box()
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a344b8198>
```



In [48]:
```
# Drop real_time_106999

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



In [50]:
```
#Preprocessing for data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [52]:
```
# 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 [53]:
```
final_df.columns
```

Out[53]:
```
Index(['real_data_106999', 'imag_data_106999', 'real_data_27999999',
       'imag_data_27999999', 'cell_type'],
      dtype='object')
```

In [57]:
```
# 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 [58]:
```
scaler.fit(X)
```

Out[58]: StandardScaler(copy=True, with_mean=True, with_std=True)

In [59]:
```
scaled_features = scaler.transform(final_df[['real_data_106999', 'imag_d
                                             'real_data_27999999','imag_
```

In [62]:
```
df_feat=pd.DataFrame(scaled_features, columns = final_df.columns[:-1] )
df_feat.head()
```

Out[62]:

	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 [63]:
```
# Define X and Y
X = df_feat
y = final_df['cell_type']
```

In [64]:
```
# 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 [65]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
```

In [66]:
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression()
```

In [67]:
```
log_model.fit(X_train, y_train)
```

```
/Users/jayashrijagannathan/anaconda3/lib/python3.7/site-packages/sklea
rn/linear_model/logistic.py:432: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/jayashrijagannathan/anaconda3/lib/python3.7/site-packages/sklea
rn/linear_model/logistic.py:469: FutureWarning: Default multi_class wi
ll be changed to 'auto' in 0.22. Specify the multi_class option to sil
ence this warning.
  "this warning.", FutureWarning)
```

Out[67]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [68]:
```
y_pred = log_model.predict(X_test)
```

In [69]:
```
# Summary of the predictions made by the classifier
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.43      0.75      0.55    126569
           1       0.45      0.29      0.35    116658
           2       0.55      0.24      0.33     87149

    accuracy                           0.45    330376
   macro avg       0.48      0.42      0.41    330376
weighted avg       0.47      0.45      0.42    330376
```

In [70]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[94880 27674  4015]
 [70761 33366 12531]
 [53684 12953 20512]]
```

In [72]:
```
from sklearn.metrics import accuracy_score
print('accuracy = ', accuracy_score(y_pred, y_test))
```

```
accuracy =  0.45026878465748116
```

In [75]:
```
# Support Vector Machines(SVM)
from sklearn.svm import SVC

svm_model = SVC()
```

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

In [ ]: