

Final Project Submission

STATS 202: Data Mining and Analysis

Stanford University, Summer Session 2023

Instructor: Dr. Linh Tran

Submitted by:

Adam Kainikara

adamkainikara@gmail.com

Project Overview

This document contains my final project for STATS 202, completed in the Summer of 2023 at Stanford University. The project is divided into two parts:

- **Part I** – Final report
- **Part II** – Python code (approximately 1000 lines)

The project applies various machine learning and data analysis techniques to a clinical dataset of patients diagnosed with schizophrenia. The dataset includes 10 attributes and a total of 110,047 observations (80,046 for training and 30,001 for testing).

Model outputs were submitted to two internal Kaggle competitions:

1. **PANSS Score Prediction:** Predicting the Positive and Negative Syndrome Scale (PANSS) score to assess the severity of schizophrenia.
2. **Treatment Classification:** Classifying patients into categories such as continuing treatment, switching treatment, or discharge.

I placed 2nd in the PANSS prediction task and 7th in the classification task, resulting in a 4th overall ranking out of 40 students who participated in the competition.

Thank you for taking the time to review my final project. To contact or discuss any material of the project, please contact me at: adamkainikara@gmail.com

Please continue to the following pages.

STATS 202: DATA MINING AND ANALYSIS FINAL

INSTRUCTOR: LINH TRAN

FINAL PROJECT

DUE DATE: AUGUST 2, 2023

STANFORD UNIVERSITY

ADAM KAINIKARA

Kaggle Reference

Team Name: Adam Kainikara.

Work done by Adam Kainikara.

Part 1. Treatment Effect

Using Python I loaded the data from all available CSV sets. Created a function called find_patients which took the input of the loaded data and returned every unique patient id. After collecting every patient id, I created a blank dictionary. The dictionary would have the patient id as the key. For values, I stored all the associated values of each patient in an array. For patients that had one or more visit days, the information would also be stored. The following is an example of how I stored the data.

```
dtype=[('study', '<U10'), (country, '<U10'), ('txgroup', '<U10'), ('assessmentid', '<f8'), ('patientid', '<f8'), ('visitday', '<i4'), ('xvalues', '<f8', (30,)), ('panss', '<f8'), ('leadstatus', '<U10')], 30951.0: array([('C', "China", "Control", 304958., 30951., 0, [4., 3., 2., 1., 3., 4., 2., 5., 4., 5., 3., 1., 2., 1., 4., 1., 4., 4., 3., 3., 4., 6., 3., 1., 5., 2.], 94., "Assign to"), ("C", "China", "Control", 301327., 30951., 7, [4., 3., 2., 1., 2., 4., 2., 4., 5., 4., 5., 3., 1., 1., 1., 4., 1., 1., 4., 4., 3., 3., 4., 6., 3., 1., 4., 2.], 91., "Assign to"), ("C", "China", "Control", 303725., 30951., 14, [4., 3., 2., 1., 1., 4., 2., 4., 5., 4., 5., 4., 5., 3., 1., 1., 1., 4., 2., 4., 3., 3., 4., 6., 3., 1., 4., 2.], 91., "Passed"), ("C", "China", "Control", 304954., 30951., 42, [4., 3., 4., 1., 1., 4., 2., 4., 5., 5., 4., 5., 3., 1., 2., 1., 1., 4., 3., 4., 3., 3., 4., 6., 3., 2., 5., 2.], 98., "Passed"), ("C", "China", "Control", 307645., 30951., 70, [4., 2., 2., 3., 1., 4., 3., 5., 6., 5., 5., 4., 2., 3., 1., 1., 4., 3., 5., 5., 3., 3., 4., 6., 3., 3., 5., 2.], 108., "Passed")]
```

After this was done, patients were further filtered into a control group and a treatment group. In addition, patients with fewer than one visit day were removed. Below is a graph of treatment and control groups. The X axis is visit day and Y axis is PANSS score. To reduce clutter on the graph, only Study E is shown.

After, I calculated the difference between each patient's final PANNS score and their initial score. The distribution of the difference in scores can be seen below.

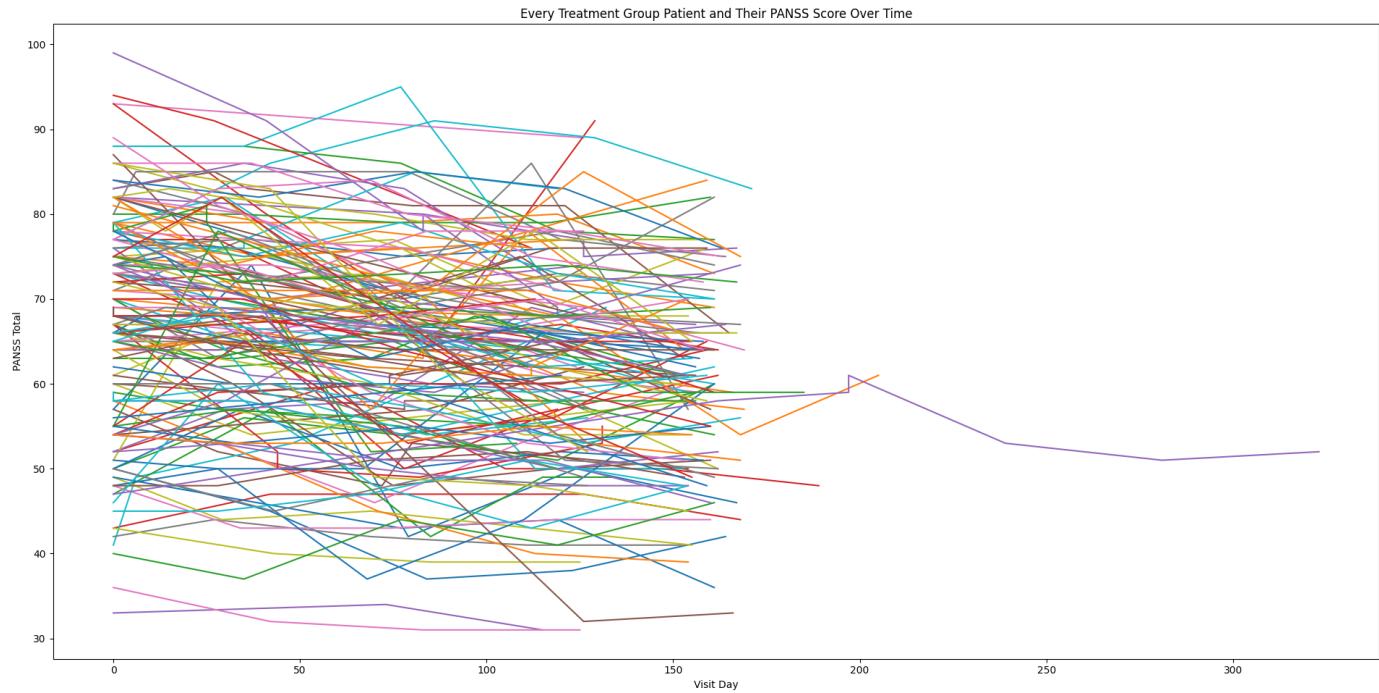


FIGURE 0.1. Treatment Group: Study E

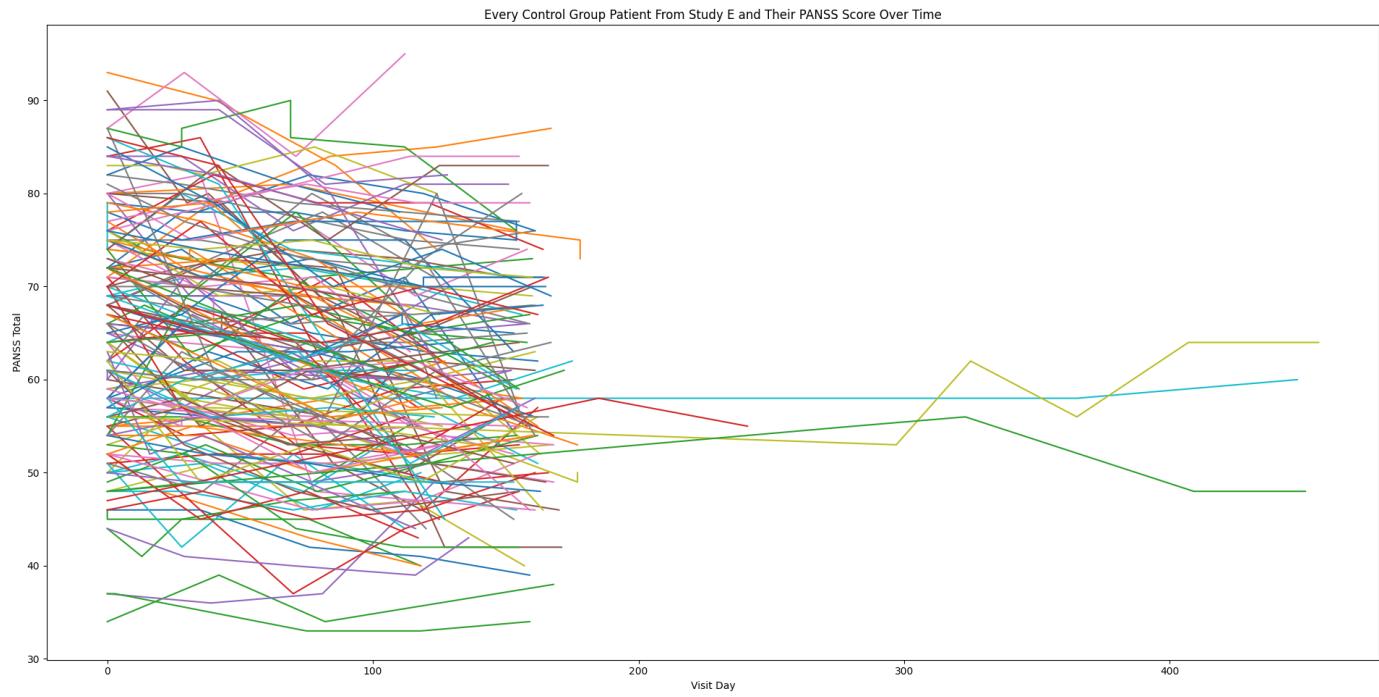


FIGURE 0.2. Control Group: Study E

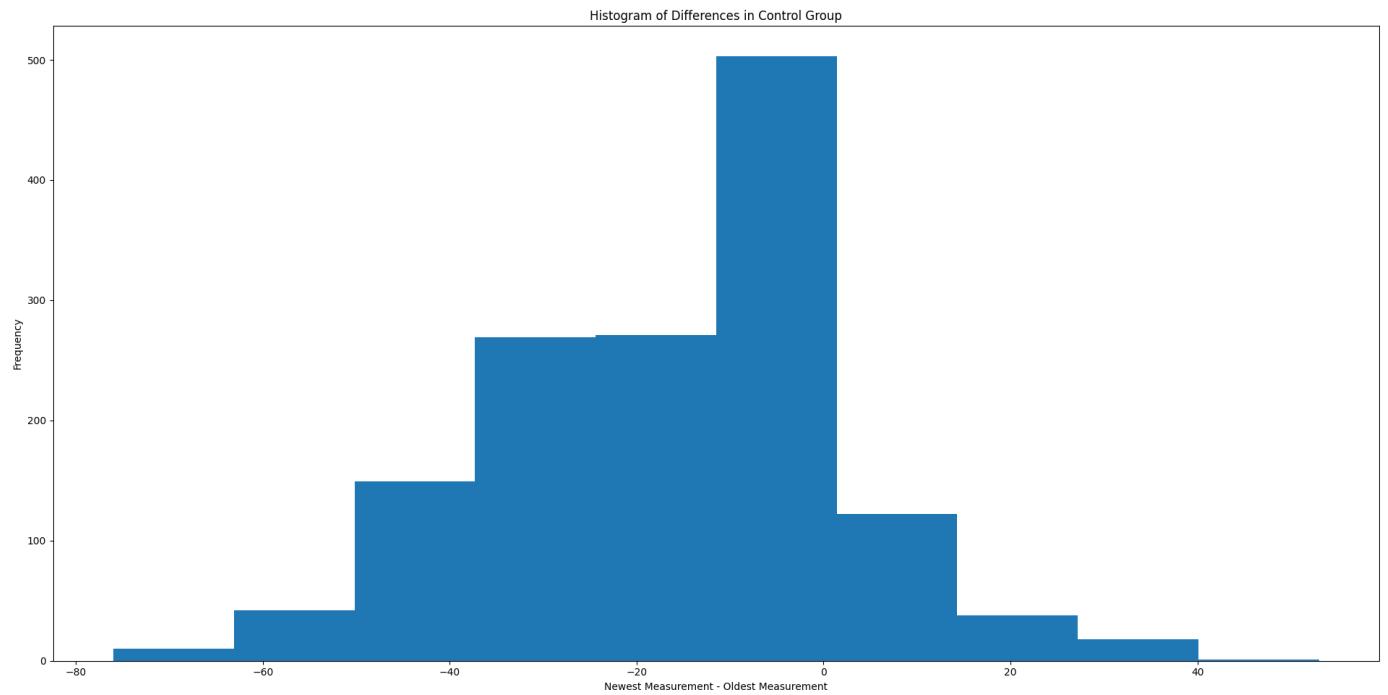


FIGURE 0.3. Differences in Control Group

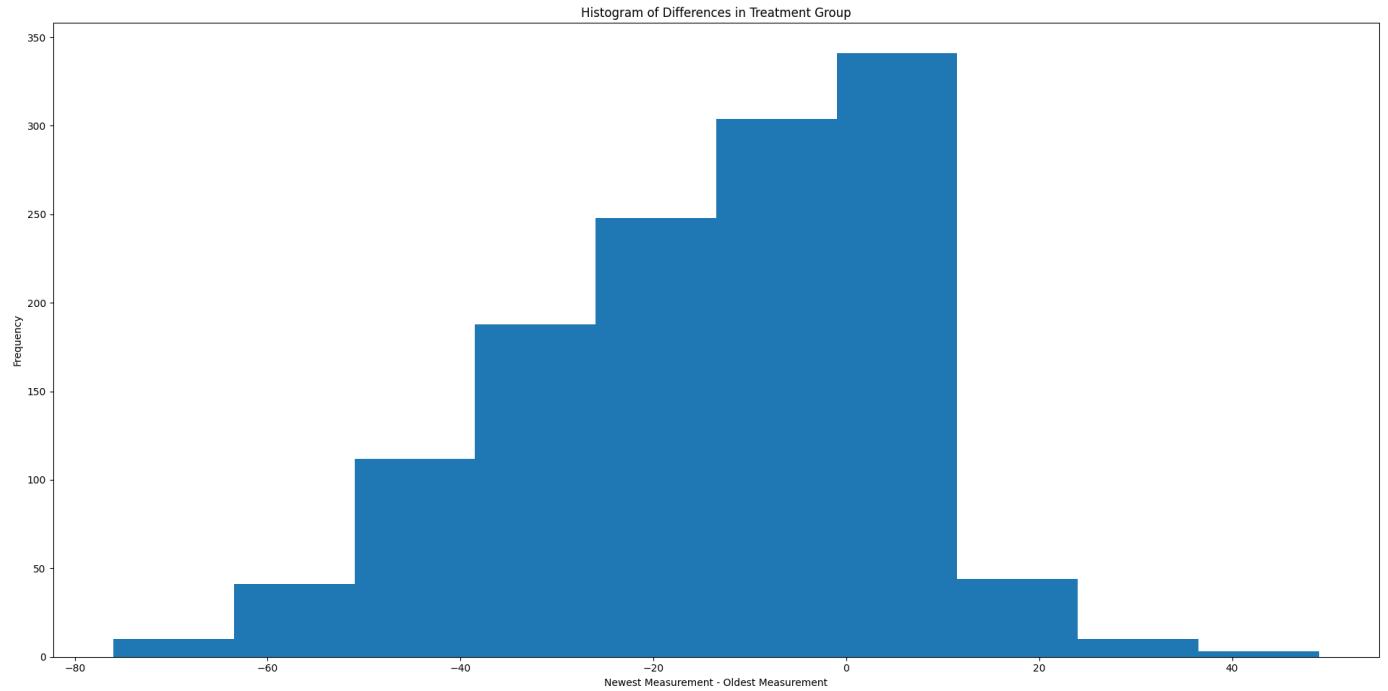


FIGURE 0.4. Differences in Treatment Group

The distribution of difference scores in the control group is somewhat normal where as the distribution of difference of scores in the treatment group is skewed. The mean difference of patients in the control group was -16.13 with a standard deviation of 19.40. That is on average, patients in the control group on average saw their PANSS score decrease by 16 at the end of the study. The mean difference of patients in the treatment group was -15.98 with a standard deviation of 19.24.

To see if the treatment had any affect, a t test will be conducted where H_0 is there is no significant difference in the mean change in score between the treatment and control groups and H_a is there is a significant difference in the mean change in score between the treatment and control groups. Patients were randomly assigned to either treatment or control. With $\alpha = 0.05$ a test statistic of 0.2105 and 2945 degrees of freedom. The p value is 0.416. The result is not significant. The treatment affect does not make a significant impact on the decrease in PANSS Scores.

Part 2. Patient Segmentation

For this problem I used two forms of clustering. One being k means clustering, and the other being hierarchical clustering so I could see a dendrogram. For K means clustering, I began by clustering on various components that made up the PANSS score. For example I tried P1 with N1, P1 with G1 etc. I choose 3 centroids. I also used cross validation to choose the best k. In the scatter plot below, it shows the locations of the centroids of the clusters when clustering using P1 and N1. The centroids are in orange. The blue dots are the xvalue combinations with P1 and N1.

I then wanted to know what a dendrogram of this would look like, so one was constructed. There were two clusters. One thing I found quite interesting is that one of the clusters (green is N1) is larger than the other cluster. I thought that the clusters would be of similar size. Also the clusters don't meet for a while.

Part 3. Forecasting

Kaggle Reference. Team Name: Adam Kainikara.

Work done by Adam Kainikara.

There were many data wrangling steps involved in this part of the project. I used data from set E for this part. At first, I created a structured array which was organized by study, country, txgroup, assesment id, patient id, visit day, xvalues, panss score and lead status. Then I made a function that found every unique patient id. This was to figure out how many patients were in the study. After collecting every patient id, I created a blank dictionary. The dictionary would have the patient id as the key. For values, I stored all the associated values of each patient in an array. These associated values include: PANSS score, country, etc. For patients that had one or more visit days, the information would also be stored in arrays. It is stored in the same way as seen in section 1. Patients who did not have at least two visit days were removed from the study. In a spread sheet, I did some basic calculations. These calculations included figuring out the average % change for each patient from the initial reading in the study to the last reading in the study.

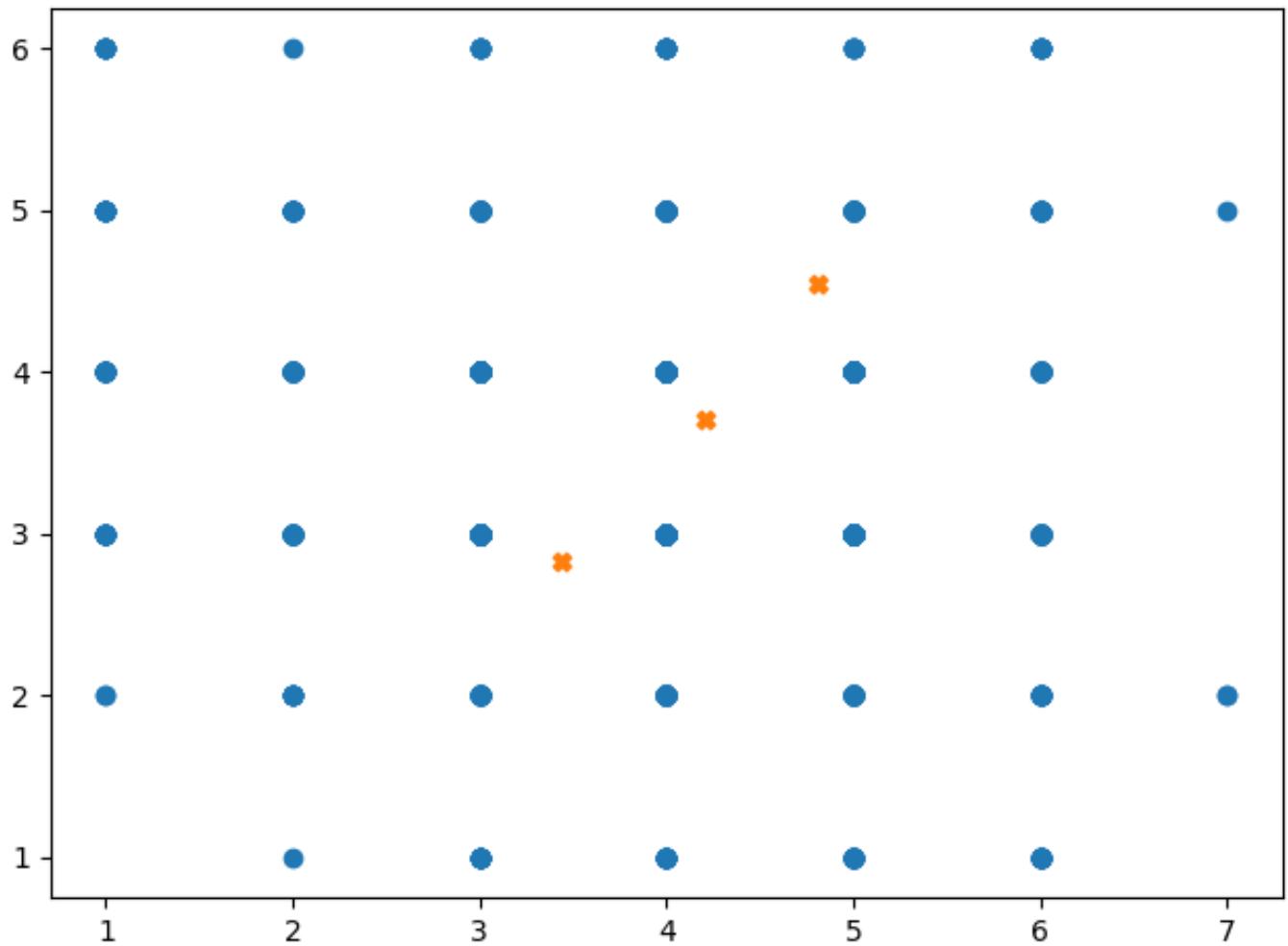


FIGURE 0.5. K Means Clustering

For my first prediction, I uploaded a CSV that contained the most recent visit day's PANSS score. I did this to see if the most recent visit PANSS score would be similar to the real 18th week score. This resulted in a score of 6.61736.

My next submission was the score decreased by the average decrease calculated in the spread sheet. To do this, I calculated each patient's change in PANSS score from the first week to the last week. I found the average % drop across the data set. I then changed the initial reading (first day's reading) by the average % change. The average % change across the data set was about 10%. This did not change my score significantly.

For my final few submissions I created a function that did linear interpolation where:

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1} \text{ and}$$

$$\hat{y} = (1 - \alpha)y_1 + \alpha y_2$$

The linear interpolation resulted in better results. My best score was 6.5099. In my created function, I tried several variations. The first variation I did was predict the average PANSS score if a patient did not get to day 129.5. I used day 129.5 as it is 18.5 weeks. In my spreadsheet I calculated that the average visit day turns into about 17.6 weeks. Although day

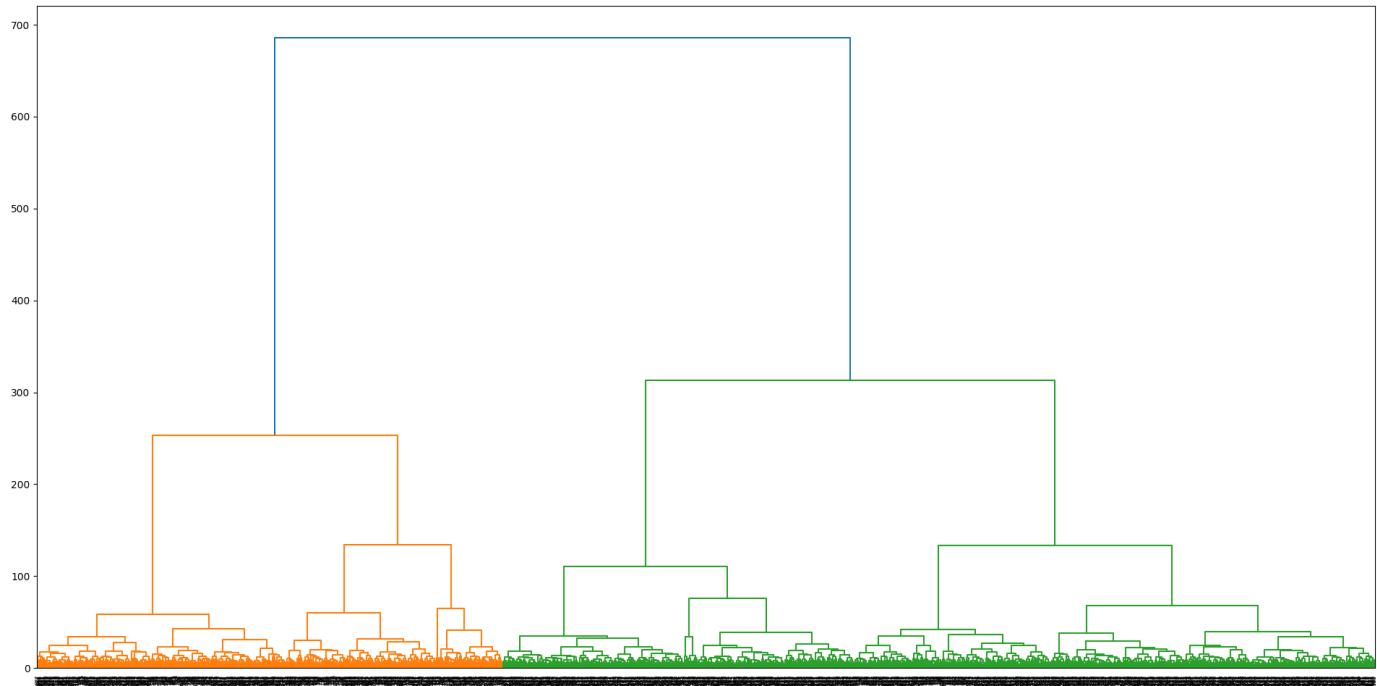


FIGURE 0.6. Dendrogram

126 (18 weeks) does not necessarily mean that that was the patient's 18th week, I assumed that most patients would be around this time period. This assumption was made because the average patient stayed for 17.6 weeks.

Another variation I did if a patient did not make it to day 129.5 was to calculate the slope between the last two points, fit a line, and predict what the score for day 129.5 would have been.

The last attempt I did involved extrapolation. When I used extrapolation though, my score got worse (7.74936) so I did not use extrapolation. In the end using linear interpolation with the day as 129.5 days produced the best results for me. The code for this section is the function(s) `desired_data` in the `interpol3.py` file.

Part 4. 4 Binary Classification

Kaggle Reference

Team Name: Adam Kainikara.

Work done by Adam Kainikara.

There were many data wrangling steps involved in this part of the project. I used data sets A,B,C,D for training and E as the test set. At first, I created a structured array which was organized by study, country, txgroup, assesment id, patient id, visit day, xvalues, panss score and lead status. Then made a function that found every unique patient id. This was to figure out how many patients were in the study. After collecting every patient id, I created a blank dictionary. The dictionary would have the patient id as the key. For values, I stored all the associated values of each patient in an array. For patients that had one or more visit

days, the information would also be stored. Patients who did not have at least two visit days were removed from the study.

For classification I used many classification methods. These include knn, naive bayes, decision trees, random forests and support vector machines. When using knn, I choose many different K values, however my accuracy was never very good so I decided not to use it. My score on kaggle was 16. The others (naive bayes, decision trees, random forests) produced poor training accuracy.

My best scores occurred while using a support vector machine. The support vector machine gave me a score of 0.858 with a training accuracy of 0.812.

To see if I could improve my score, I tried using different kernels such as linear or sigmoid however these did not improve my score. I did add a regularization parameter of 0.95. This changed my score to 0.854

finalproj2/

finalproj2/	1
—— finalp3.py	1
—— interplop3.py	3
—— p3class.py	6
—— prob2.py	9
—— realp3.py	12


```
# finalp3.py
```

```
1 from numpy import *
2 import sys
3 import matplotlib.pyplot as plt
4 patient_visit_dt = dtype([('study', 'U10'), ('country', 'U10'), ('txgroup', 'U10'), ('patientid', float64), ('visitday', int32), ('xvalues', float64), ('visitday', int32), ('xvalues', float64), ('panss', float64)])
5
6 def data_loader(fname):
7     #Study      Country PatientID      SiteID  RaterID AssessmentID  TxGroup VisitDay      P1      P2      P3      P4      P5      P6      P7      N1      N2      N3      N4      N5
8     # 0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19      20      21      22      23      24      25      26      27      28      29      30      31      32      33      34      35      36      37      38
9     # G7      G8      G9      G10      G11      G12      G13      G14      G15      G16      PANSS_Total
10    # 28      29      30      31      32      33      34      35      36      37      38
11
12    #data_a = loadtxt(fname, skiprows=1, usecols=(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',')
13    cata_data_a = loadtxt(fname, skiprows=1, usecols=(0,1,6), delimiter=',', dtype=str)
14    quant_data_a = loadtxt(fname, skiprows=1, usecols=(2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',', dtype=float64)
15    #print(cata_data_a, quant_data_a)
16    #print(cata_data_a.shape, quant_data_a.shape)
17    #print(type(cata_data_a), type(quant_data_a))
18
19
20    data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
21    #print(data_a.shape)
22
23    #raise SystemExit
24    #visit_a = array((visit_a[['study', 'country', 'txgroup']], visit_a[['xvalues']], visit_a[['yvalues']]), dtype = patient_visit_dt)
25    #x_v = quant_data_a[:,5:35]
26
27    #y_v = quant_data_a[:,35]
28    #print(data_a[['study', 'country', 'txgroup']].shape, cata_data_a.shape)
29    data_a[['study']] = cata_data_a[:, 0]
30    data_a[['country']] = cata_data_a[:, 1]
31    data_a[['txgroup']] = cata_data_a[:, 2]
32    data_a[['patientid']] = quant_data_a[:, 0]
33    data_a[['visitday']] = quant_data_a[:, 4]
34    #print("eeeeee", quant_data_a[:,4])
35    #print("aaaaa", quant_data_a[:,0])
36
37    #print(quant_data_a[:,5:35])
38    #print(quant_data_a[:,35])
39
40    data_a[['xvalues']] = quant_data_a[:,5:35]
41    data_a[['panss']] = quant_data_a[:,35]
42    #print(data_a[['xvalues']].shape, data_a[['panss']].shape)
43    #print(data_a)
44
45    #data_a[['study', 'country', 'txgroup']] = tuple(cata_data_a[:, i] for i in range(3))
46    #print('!!!', cata_data_a[:,2])
47    #data_a[['study', ]] = cata_data_a[:,0, :2]
48    #cata_data_a[:, :, 2]
49
50    #visit_a[['xvalues']] = x_v
51    #visit_a[['yvalues']] = y_v
52    #visit_a[['country', 'study']] = cata_data_a[:,0], cata_data_a[:,1]
53    return data_a, cata_data_a, quant_data_a
54
55 def trial():
56     num_l = []
57     for i in range(100):
58         num_l.append(i)
59         #print(i)
60     return num_l
61
62 def find_patients(data_a):
63     d = {}
64     #rows = range(data_a.shape[0])
65     for i in range(data_a.shape[0]):
66         key = data_a[i]['patientid']
67         if key in d:
68             d[key].append(i)
69         else:
70             d[key] = [i]
71
72     patient_d = {}
73
74     for patientid, index_l in d.items():
75         patient_a = data_a[index_l]
76         index_v = patient_a['visitday'].argsort()
77         patient_d[patientid] = patient_a[index_v]
78
79     print(type(patient_d))
80     print("test", patient_d)
81     return patient_d
82
83 def separate_control_treatment(patient_d):
84     control_l = []
85     treatment_l = []
86
87     for patient_a in patient_d.values():
88         if patient_a[0]['txgroup'] == "Control":
89             control_l.append(patient_a)
90         else:
91             treatment_l.append(patient_a)
92     #print(control_l, treatment_l)
93     return control_l, treatment_l
94
95 def plot_patient_data(data_l):
96     print(type(data_l))
97     print(len(data_l))
98
99     fig, ax = plt.subplots()
100
101    for patient_a in data_l:
102
103        x_l = patient_a['visitday']
104        y_l = patient_a['panss']
105        ax.plot(x_l, y_l)
106    plt.xlabel('Visit Day')
107    plt.ylabel('PANSS Total')
108    plt.title('Every Treatment Group Patient and Their PANSS Score Over Time')
109    plt.show()
110
111 def filter_patients(patient_d, day_limit=126):
112     print(type(patient_d))
113     print(patient_d)
114     #delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
115     delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
116     for patientid in delete_patient_l:
117         del patient_d[patientid]
118
119 def fit_linearmodel():
120     pass
121
122 data_a, cata_data_a, quant_data_a = data_loader("Study_E.csv")
123 patientid_v = data_a['patientid']
124 upatientid_v = unique(patientid_v)
125 print('Patient id:', upatientid_v.shape[0])
126
127 print("just viewing", patientid_v)
128 #print(da)
129
```

```

130 # day_v = data_a['visitday']
131 # # uday_v = unique(day_v)
132 # week_v = uday_v/7
133 # print(uday_v)
134 # print('-----')
135 # print('-----')
136 # print(week_v)
137 # print('-----')
138
139 # print(f'data a:10 {data_a[10:]}')
140 m_v = data_a['txgroup'] == "Treatment"
141 day_v = data_a[m_v]['visitday']
142 print('treatment day', sorted(day_v))
143 uday_v = unique(day_v)
144 # week_v = uday_v//7
145 # print(f'uday_v treatments: {uday_v}')
146 # print('week_v treatments: {week_v}')
147 # #week_v = int32(week_v)
148 # print('week count:', list(zip(arange(week_v.shape[0]), bincount(week_v))))
149 patient_d = find_patients(data_a)
150 print('Before delete:', len(patient_d))
151 filter_patients(patient_d, day_limit=105)
152 print('After delete:', len(patient_d))
153
154 #print(len(a))
155 print('-----')
156 #print(d)
157
158 control_l, treatment_l = seperate_control_treatment(patient_d)
159 print("control", control_l)
160 print('-----')
161 print('-----')
162 print('-----')
163 print('-----')
164 print('-----')
165 print("treatment", treatment_l)
166
167 plot_patient_data(treatment_l)
168 plot_patient_data(control_l)
169 raise SystemExit
170 plt.xlabel('Visit Day')
171 plt.ylabel('PANSS Total')
172 plt.title('Every Treatment Group Patient and Their PANSS Score Over Time')
173 if __name__ == '__main__':
174     main()

```

```
# interp03.py
```

```
1 from cgi import test
2 from numpy import *
3 import sys
4 import matplotlib.pyplot as plt
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.naive_bayes import CategoricalNB
7 from sklearn.naive_bayes import GaussianNB
8 from sklearn import tree
9 from sklearn.ensemble import RandomForestClassifier
10
11 patient_visit_dt = dtype([('study','U10'),('country','U10'),('txgroup','U10'),('assesmentid', float64),('patientid', float64),('visitday', int32),('xvalues',float64,(30)),('panss',float64),
12
13 def data_loader(fname):
14     #Study   Country PatientID      SiteID RaterID AssessmentID   TxGroup VisitDay      P1      P2      P3      P4      P5      P6      P7      N1      N2      N3      N4      N5
15     # 0       1       2       3       4       5       6       7       8       9       10      11      12      13      14      15      16      17      18      19      20      21      22      23      24      25      26      27
16     # G7      G8      G9      G10     G11     G12     G13     G14     G15     G16     PANSS_Total
17     # 28      29      30      31      32      33      34      35      36      37      38
18
19     if 'Study_E.csv' in fname:
20         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6), delimiter=',', dtype=str )
21     else:
22         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6,39), delimiter=',', dtype=str )
23     quant_data_a = loadtxt(fname,skiprows=1, usecols=(2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',', dtype=float64)
24
25
26     data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
27     #print(data_a.shape)
28
29
30     data_a[['study']] = cata_data_a[:, 0]
31     data_a[['country']] = cata_data_a[:, 1]
32     data_a[['txgroup']] = cata_data_a[:, 2]
33     data_a[['assesmentid']] = quant_data_a[:,3]
34     data_a[['patientid']] = quant_data_a[:,0]
35     data_a[['visitday']] = quant_data_a[:,4]
36
37
38     data_a[['xvalues']] = quant_data_a[:,5:35]
39     data_a[['panss']] = quant_data_a[:,35]
40     if 'Study_E.csv' in fname:
41         data_a[['leadstatus']] = 0
42     else:
43         data_a[['leadstatus']] = cata_data_a[:,3]
44
45     return data_a
46
47 def trial():
48     num_l = []
49     for i in range(100):
50         num_l.append(i)
51     return num_l
52
53 def find_patients(data_a):
54     d = {}
55     for i in range(data_a.shape[0]):
56         key = data_a[i]['patientid']
57         if key in d:
58             d[key].append(i)
59         else:
60             d[key] = [i]
61
62     patient_d = {}
63     for patientid, index_l in d.items():
64         patient_a = data_a[index_l]
65         index_v = patient_a['visitday'].argsort()
66         patient_d[patientid] = patient_a[index_v]
67
68     print(type(patient_d))
69     return patient_d
70
71 def separate_control_treatment(patient_d):
72     control_d = {}
73     treatment_d = {}
74
75     for patient_a in patient_d.values():
76
77         if patient_a[0]['txgroup'] == 'Control':
78             control_d[patient_a[0]['patientid']] = patient_a
79         else:
80             #print('test')
81             treatment_d[patient_a[0]['patientid']] = patient_a
82
83     return control_d, treatment_d
84
85
86 def plot_patient_data(data_l):
87     print(type(data_l))
88     print(len(data_l))
89
90     fig, ax = plt.subplots()
91
92     for patient_a in data_l:
93
94         x_l = patient_a['visitday']
95         y_l = patient_a['panss']
96         ax.plot(x_l, y_l)
97     plt.show()
98
99 def filter_patients(patient_d, day_limit=126):
100    print(type(patient_d))
101    print(patient_d)
102    #delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
103    delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
104    for patientid in delete_patient_l:
105        del patient_d[patientid]
106
107 def difference_in_fields(patient_d, field_name):
108     difference_values_l = []
109     for patient_a in patient_d.values():
110         dif1 = patient_a[1][field_name]
111         dif2 = patient_a[0][field_name]
112         dif = dif1-dif2
113         #dif = patient_a[-1]['panss'] - patient_a[1]['panss']
114         difference_values_l.append(dif)
115     difference_values_a = array(difference_values_l)
116     return difference_values_a
117
118 def difference_in_scores(patient_d):
119     assert 0
120     difference_values_l = []
121     for patient_a in patient_d.values():
122         dif1 = patient_a[1]['panss']
123         dif2 = patient_a[0]['panss']
124         dif = dif1-dif2
125         difference_values_l.append(dif)
126     return difference_values_l
127
128 def difference_in_scores_stats(difference_values_l):
129     mean_difference = mean(difference_values_l)
```

```

130     standev_difference = std(difference_values_1)
131     print(f'The mean difference is {mean_difference} and has a standard deviation of {standev_difference}')
132     return mean_difference, standev_difference
133
134 def knn_pred(xtrain_a, ytrain_v, xtest_a, ytest_v):
135     knn_model = KNeighborsClassifier(n_neighbors = 5)
136     knn_model.fit(xtrain_a, ytrain_v)
137     ytrainpred_v = knn_model.predict(xtrain_a)
138     ypred_v = knn_model.predict(xtest_a)
139     ypred_proba_a = knn_model.predict_proba(xtest_a)
140     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
141     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
142
143 def class_pred(clf, xtrain_a, ytrain_v, xtest_a, ytest_v):
144     clf.fit(xtrain_a, ytrain_v)
145     ytrainpred_v = clf.predict(xtrain_a)
146     ypred_v = clf.predict(xtest_a)
147     ypred_proba_a = clf.predict_proba(xtest_a)
148     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
149     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
150
151
152     # m_v = ypred_v == ytest_v
153     # print('match samples:', ypred_v[m_v])
154     # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
155
156     print(f'The training accuracy is: {accuracy_train} ')
157     print(f'The test accuracy is {accuracy_test}')
158     return ypred_proba_a
159
160 def z_score_converter(data_a):
161     train_data_a = data_a['xvalues']
162     #print(train_data_a)
163     mean_a = train_data_a.mean(axis=0)
164     standev_a = train_data_a.std(axis=0)
165     zscore = (train_data_a - mean_a)/standev_a
166     #print(zscore)
167     return zscore
168
169 def z_score_convert2(data_a, input_test_data_a):
170     train_data_a = data_a['xvalues']
171     test_data_a = input_test_data_a['xvalues']
172     #print(train_data_a)
173     train_mean_v = train_data_a.mean(axis=0)
174     train_standev_v = train_data_a.std(axis=0)
175     train_zscore_a = (train_data_a - train_mean_v)/train_standev_v
176     test_zscore_a = (test_data_a - train_mean_v)/train_standev_v
177     #print(zscore)
178     return train_zscore_a, test_zscore_a
179
180 def time_shifter(patient_d):
181     #adjusted_d = {}
182     for patient_id, visitday in patient_d.items():
183         time_zero = visitday[0]['visitday']
184         visitday[0] = 0
185         for other_days in visitday[1:]:
186             other_days['visitday']-=time_zero
187
188
189
190 def desired_data0(patient_d):
191     predicted_data_l = []
192     for patient_id, array_values_a in patient_d.items():
193         last_array = array_values_a[-1]
194         output_data = last_array['panss']
195         predicted_data_l.append((patient_id, output_data))
196     submitted_data_a = array(predicted_data_l, output_data)
197     return submitted_data_a
198
199
200 def desired_data1(patient_d, target_day = 129.5):
201     predicted_data_l = []
202     for patient_id, patient_a in patient_d.items():
203
204         week = (patient_a[-1]['visitday']-patient_a[0]['visitday'])/7
205         if week > 18:
206             print('more days', patient_id, week)
207             day_v = patient_a['visitday']
208             day_v = day_v - day_v[0]
209             day_v = day_v - day_v[-1]
210             panss_v = patient_a['panss']
211
212             if target_day-day_v[-1] and patient_a.shape[0] > 1 and day_v[-1] != day_v[-2]:
213                 m = (panss_v[-1]-panss_v[-2])/(day_v[-1]-day_v[-2])
214                 panss = m*(target_day-day_v[-1]) + panss_v[-1]
215             else:
216                 panss = interp(target_day, day_v, panss_v)
217
218             predicted_data_l.append((patient_id, panss))
219     return array(predicted_data_l)
220
221 def desired_data(patient_d, target_day = 129.5):
222     predicted_data_l = []
223     for patient_id, patient_a in patient_d.items():
224
225         week = (patient_a[1]['visitday']-patient_a[0]['visitday'])/7
226         if week > 18:
227             print('more days', patient_id, week)
228             day_v = patient_a['visitday']
229             day_v = day_v - day_v[0]
230             day_v = day_v - day_v[-1]
231             panss_v = patient_a['panss']
232
233             if target_day-day_v[-1]>50:
234                 panss = 62.5
235
236             else:
237                 panss = interp(target_day, day_v, panss_v)
238
239             predicted_data_l.append((patient_id, panss))
240     return array(predicted_data_l)
241
242
243 def class_data_merge(test_data_a, maxproba_v):
244     #print("see assesment shapes", test_data_a['assesmentid'].shape)
245     class_proba_l = list(zip(test_data_a['assesmentid'],maxproba_v))
246     class_proba_a = array(class_proba_l)
247
248     return class_proba_a
249
250 def save_file(patient_data_a, fname):
251     savetxt(fname, patient_data_a, delimiter=',', fmt='%d')
252
253 def main():
254     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]])
255     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]])
256     test_data_a = data_loader(sys.argv[-1])
257
258     m_v = logical_not(train_data_a['leadstatus'] == "Passed")
259     train_data_a = train_data_a[m_v]
260     m_v = logical_not(test_data_a['leadstatus'] == "Passed")
261     test_data_a = test_data_a[m_v]
262
263

```

```

264
265 # print("seeing train after load", train_data_a.shape)
266 #print("seeing test after load", test_data_a.shape)
267
268 patientid_v = data_a['patientid']
269 upatientid_v = unique(patientid_v)
270 #print('Patient id:', upatientid_v.shape[0])
271
272 #print("just viewing", patientid_v)
273
274 #print(da)
275 patient_d = find_patients(test_data_a)
276
277 #print('-----')
278
279 control_d, treatment_d = seperate_control_treatment(patient_d)
280 #print("view control people", control_d)
281 #print(type(control_d))
282 #print("view treatment people", treatment_d)
283 #print(type(treatment_d))
284 #print('-----')
285
286 control_patientdiff = difference_in_fields(control_d, 'panss')
287 # plt.hist(control_patientdiff, bins = 10)
288 # plt.xlabel('Newest Measurement - Oldest Measurement')
289 # plt.ylabel('Frequency')
290 # plt.title('Histogram of Differences in Control Group')
291 # plt.show()
292 control_patientdiffstats = difference_in_scores_stats(control_patientdiff)
293 #print(type(control_patientdiff))
294
295 treatment_patientdiff = difference_in_fields(treatment_d, 'panss')
296 # plt.hist(treatment_patientdiff, bins = 10)
297 # plt.xlabel('Newest Measurement - Oldest Measurement')
298 # plt.ylabel('Frequency')
299 # plt.title('Histogram of Differences in Treatment Group')
300 # plt.show()
301 treatment_patientdiffstats = difference_in_scores_stats(treatment_patientdiff)
302
303 #print('-----')
304
305
306 print(f'Control Patients: {control_patientdiffstats}')
307 print(len(control_d))
308 print(f'Treatment Patients: {treatment_patientdiffstats}')
309 print(len(treatment_d))
310
311 #print(patient_d)
312 fname = "upload1.csv"
313 upload_data_a = desired_data(patient_d)
314 print(upload_data_a)
315 save_file(upload_data_a, fname)
316 print('-----')
317 zscore_train_a, zscore_test_a = z_score_convert2(train_data_a, test_data_a)
318
319
320
321
322
323 #ypredproba_a = knn_pred(zscore_train_a, train_data_a['leadstatus'], zscore_test_a, test_data_a['leadstatus'])
324 #ypredproba_a = knn_pred(train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
325
326 #clf = KNeighborsClassifier(n_neighbors = 5)
327 #clf = CategoricalNB(force_alpha=True)
328 #clf = GaussianNB()
329 #clf = tree.DecisionTreeClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, criterion='entropy')
330 #clf = RandomForestClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, random_state=0)
331 ypredproba_a = class_pred(clf, train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
332
333
334 print(ypredproba_a)
335 print('-----')
336
337 max_proba_v = ypredproba_a.max(axis=1)
338 class_data_a = class_data_a.merge(test_data_a, max_proba_v)
339
340 fname = "classupload1.csv"
341 save_file(class_data_a, fname)
342 raise SystemExit
343 plot_patient_data(control_d)
344 plot_patient_data(treatment_d)
345 raise SystemExit
346
347 # day_v = data_a['visitday']
348 # # uday_v = unique(day_v)
349 # week_v = uday_v/7
350 # print(uday_v)
351 # print('-----')
352 # print('-----')
353 # print(week_v)
354 # print('-----')
355
356 # print(f'data a:10 {data_a[10:]}')
357 m_v = data_a['txgroup'] == 'Treatment'
358 day_v = data_a[m_v]['visitday']
359 print('treatment day', sorted(day_v))
360 uday_v = unique(day_v)
361 # week_v = uday_v//7
362 # print(f'uday_v treatments: {uday_v}')
363 # print(f'week_v treatments: {week_v}')
364 # #week_v = int32(week_v)
365 # print('week count:', list(zip(arange(week_v.shape[0]), bincount(week_v))))
366 patient_d = find_patients(data_a)
367 print('Before delete:', len(patient_d))
368 filter_patients(patient_d, day_limit=105)
369 print('After delete:', len(patient_d))
370
371 #print(len(a))
372 print('-----')
373 #print(d)
374 print(patient_d)
375
376 control_d, treatment_d = seperate_control_treatment(patient_d)
377 #print("control", control_l)
378 print('-----')
379 print('-----')
380 print('-----')
381 print('-----')
382 print('-----')
383 #print("treatment", treatment_l)
384 plot_patient_data(control_l)
385 plot_patient_data(treatment_l)
386 if __name__ == '__main__':
387     main()

```

```

1 from cgi import test
2 from numpy import *
3 import sys
4 import matplotlib.pyplot as plt
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.naive_bayes import CategoricalNB
7 from sklearn.naive_bayes import GaussianNB
8 from sklearn import tree
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn import svm
11 patient_visit_dt = dtype([('study','U10'),('country','U10'),('txgroup','U10'),('assesmentid', float64),('patientid', float64),('visitday', int32),('xvalues',float64,(31)),('panss',float64),
12
13 def data_loader(fname):
14     #Study   Country PatientID      SiteID RaterID AssessmentID   TxGroup VisitDay      P1   P2   P3   P4   P5   P6   P7   N1   N2   N3   N4   N5
15     # 0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15     16     17     18     19     20     21     22     23     24     25     26     27
16     # G7     G8     G9     G10    G11    G12    G13    G14    G15    G16    PANSS_Total
17     # 28    29    30    31    32    33    34    35    36    37    38
18
19 if 'Study_E.csv' in fname:
20     cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6), delimiter=',', dtype=str )
21 else:
22     cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6,39), delimiter=',', dtype=str )
23 quant_data_a = loadtxt(fname,skiprows=1, usecols=(2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',', dtype=float64)
24
25
26
27 data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
28 #print(data_a.shape)
29
30
31 data_a['study'] = cata_data_a[:, 0]
32 data_a['country'] = cata_data_a[:, 1]
33 data_a['txgroup'] = cata_data_a[:, 2]
34 data_a['assesmentid'] = quant_data_a[:,3]
35 data_a['patientid'] = quant_data_a[:,0]
36 data_a['visitday'] = quant_data_a[:,4]
37
38
39 data_a['xvalues'] = quant_data_a[:,5:36]
40 data_a['panss'] = quant_data_a[:,35]
41 if 'Study_E.csv' in fname:
42     data_a['leadstatus'] = 0
43 else:
44     data_a['leadstatus'] = cata_data_a[:,3]
45 return data_a
46
47 def trial():
48     num_l = []
49     for i in range(100):
50         num_l.append(i)
51     return num_l
52
53 def find_patients(data_a):
54     d = {}
55     for i in range(data_a.shape[0]):
56         key = data_a[i]['patientid']
57         if key in d:
58             d[key].append(i)
59         else:
60             d[key] = [i]
61
62 patient_d = {}
63 for patientid, index_l in d.items():
64     patient_a = data_a[index_l]
65     index_v = patient_a['visitday'].argsort()
66     patient_d[patientid] = patient_a[index_v]
67
68 print(type(patient_d))
69 return patient_d
70
71 def seperate_control_treatment(patient_d):
72     control_d = {}
73     treatment_d = {}
74
75     for patient_a in patient_d.values():
76
77         if patient_a[0]['txgroup'] == 'Control':
78             control_d[patient_a[0]['patientid']] = patient_a
79         else:
80             #print('test')
81             treatment_d[patient_a[0]['patientid']] = patient_a
82
83     return control_d, treatment_d
84
85
86 def plot_patient_data(data_l):
87     print(type(data_l))
88     print(len(data_l))
89
90     fig, ax = plt.subplots()
91
92     for patient_a in data_l:
93
94         x_l = patient_a['visitday']
95         y_l = patient_a['panss']
96         ax.plot(x_l, y_l)
97     plt.show()
98
99 def filter_patients(patient_d, day_limit=126):
100    print(type(patient_d))
101    print(patient_d)
102    #delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
103    delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
104    for patientid in delete_patient_l:
105        del patient_d[patientid]
106
107 def difference_in_fields(patient_d, field_name):
108     difference_values_l = []
109     for patient_a in patient_d.values():
110         dif1 = patient_a[1][field_name]
111         dif2 = patient_a[0][field_name]
112         dif = dif1-dif2
113         #dif = patient_a[-1]['panss'] - patient_a[1]['panss']
114         difference_values_l.append(dif)
115     difference_values_a = array(difference_values_l)
116     return difference_values_a
117
118 def difference_in_scores(patient_d):
119     assert 0
120     difference_values_l = []
121     for patient_a in patient_d.values():
122         dif1 = patient_a[1]['panss']
123         dif2 = patient_a[0]['panss']
124         dif = dif1-dif2
125         difference_values_l.append(dif)
126     return difference_values_l
127
128 def difference_in_scores_stats(difference_values_l):
129     mean_difference = mean(difference_values_l)

```

```

130     standev_difference = std(difference_values_1)
131     print(f'The mean difference is {mean_difference} and has a standard deviation of {standev_difference}')
132     return mean_difference, standev_difference
133
134 def knn_pred(xtrain_a, ytrain_v, xtest_a, ytest_v):
135     knn_model = KNeighborsClassifier(n_neighbors = 5)
136     knn_model.fit(xtrain_a, ytrain_v)
137     ytrainpred_v = knn_model.predict(xtrain_a)
138     ypred_v = knn_model.predict(xtest_a)
139     ypred_proba_a = knn_model.predict_proba(xtest_a)
140     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
141     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
142
143 def class_pred(clf, xtrain_a, ytrain_v, xtest_a, ytest_v):
144     clf.fit(xtrain_a, ytrain_v)
145     ytrainpred_v = clf.predict(xtrain_a)
146     ypred_v = clf.predict(xtest_a)
147     ypred_proba_a = clf.predict_proba(xtest_a)
148     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
149     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
150
151
152     # m_v = ypred_v == ytest_v
153     # print('match samples:', ypred_v[m_v])
154     # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
155
156     print(f'The training accuracy is: {accuracy_train} ')
157     print(f'The test accuracy is {accuracy_test}')
158     return ypred_proba_a
159
160 def z_score_converter(data_a):
161     train_data_a = data_a['xvalues']
162     #print(train_data_a)
163     mean_a = train_data_a.mean(axis=0)
164     standev_a = train_data_a.std(axis=0)
165     zscore = (train_data_a - mean_a)/standev_a
166     #print(zscore)
167     return zscore
168
169 def z_score_convert2(data_a, input_test_data_a):
170     train_data_a = data_a['xvalues']
171     test_data_a = input_test_data_a['xvalues']
172     #print(train_data_a)
173     train_mean_v = train_data_a.mean(axis=0)
174     train_standev_v = train_data_a.std(axis=0)
175     train_zscore_a = (train_data_a - train_mean_v)/train_standev_v
176     test_zscore_a = (test_data_a - train_mean_v)/train_standev_v
177     #print(zscore)
178     return train_zscore_a, test_zscore_a
179
180 def time_shifter(patient_d):
181     #adjusted_d = {}
182     for patient_id, visitday in patient_d.items():
183         time_zero = visitday[0]['visitday']
184         visitday[0] = 0
185         for other_days in visitday[1:]:
186             other_days['visitday']-=time_zero
187
188
189
190 def desired_data(patient_d):
191     predicted_data_l = []
192     for patient_id, array_values_a in patient_d.items():
193         last_array = array_values_a[-1]
194         output_data = last_array['pans']
195         predicted_data_l.append((patient_id, output_data))
196     submitted_data_a = array(predicted_data_l, output_data)
197     return submitted_data_a
198
199 def class_data_merge(test_data_a, maxproba_v):
200     #print("see assessment shapes", test_data_a['assessmentid'].shape)
201     class_proba_l = list(zip(test_data_a['assessmentid'],maxproba_v))
202     class_proba_a = array(class_proba_l)
203
204     return class_proba_a
205
206 def save_file(patient_data_a, fname):
207     savetxt(fname, patient_data_a, delimiter=',')
208
209 def main():
210     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]])
211     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]])
212     test_data_a = data_loader(sys.argv[-1])
213
214     m_v = logical_not(train_data_a['leadstatus']) == "Passed"
215     train_data_a = train_data_a[m_v]
216     m_v = logical_not(test_data_a['leadstatus']) == "Passed"
217     test_data_a = test_data_a[m_v]
218
219
220     # print("seeing train after load", train_data_a.shape)
221     #print("seeing test after load", test_data_a.shape)
222
223     patientid_v = data_a['patientid']
224     upatientid_v = unique(patientid_v)
225     #print('Patient id:', upatientid_v.shape[0])
226
227     #print("just viewing", patientid_v)
228
229     #print(da)
230     patient_d = find_patients(data_a)
231
232     #print('-----')
233
234     control_d, treatment_d = separate_control_treatment(patient_d)
235     #print("view control people", control_d)
236     #print(type(control_d))
237     #print("view treatment people", treatment_d)
238     #print(type(treatment_d))
239     #print(type(control_d))
240     #print('-----')
241
242     control_patientdiff = difference_in_fields(control_d, 'pans')
243     control_patientdiffstats = difference_in_scores_stats(control_patientdiff)
244     #print(type(control_patientdiff))
245
246     treatment_patientdiff = difference_in_fields(treatment_d, 'pans')
247     treatment_patientdiffstats = difference_in_scores_stats(treatment_patientdiff)
248
249     #print('-----')
250
251     print(f'Control Patients: {control_patientdiffstats}')
252     print(f'Treatment Patients: {treatment_patientdiffstats}')
253
254     #print('-----')
255     #print(patient_d)
256     fname = "upload1.csv"
257     upload_data_a = desired_data(patient_d)
258     print(upload_data_a)
259     save_file(upload_data_a, fname)
260
261     #print('-----')
262     zscore_train_a, zscore_test_a = z_score_convert2(train_data_a, test_data_a)
263

```

```

264
265
266
267
268 #ypredproba_a = knn_pred(zscore_train_a, train_data_a['leadstatus'], zscore_test_a, test_data_a['leadstatus'])
269 #ypredproba_a = knn_pred(train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
270
271 #clf = KNeighborsClassifier(n_neighbors = 5)
272 #clf = CategoricalNB(force_alpha=True)
273 #clf = GaussianNB()
274 #clf = tree.DecisionTreeClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, criterion='entropy')
275 #clf = RandomForestClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, random_state=0)
276 clf = svm.SVC(probability=True)
277 ypredproba_a = class_pred(clf, train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
278
279
280 print(ypredproba_a)
281 print('-----')
282
283 max_proba_v = ypredproba_a.max(axis=1)
284 class_data_a = class_data_merge(test_data_a, max_proba_v)
285
286 fname = "classupload1.csv"
287 save_file(class_data_a, fname)
288
289 raise SystemExit
290
291 # day_v = data_a['visitday']
292 # # uday_v = unique(day_v)
293 # week_v = uday_v/7
294 # print(uday_v)
295 # print('-----')
296 # print('-----')
297 # print(week_v)
298 # print('-----')
299
300 # print(f'data_a[10: {data_a[10:]}]')
301 m_v = data_a['txgroup'] == 'Treatment'
302 day_v = data_a[m_v]['visitday']
303 print('treatment day', sorted(day_v))
304 uday_v = unique(day_v)
305 # week_v = uday_v//7
306 # print(f'uday_v treatments: {uday_v}')
307 # print(f'week_v treatments: {week_v}')
308 # #week_v = int32(week_v)
309 # print('week count:', list(zip(arange(week_v.shape[0]), bincount(week_v))))
310 patient_d = find_patients(data_a)
311 print('Before delete:', len(patient_d))
312 filter_patients(patient_d, day_limit=105)
313 print('After delete:', len(patient_d))
314
315 #print(len(a))
316 print('-----')
317 #print(d)
318 print(patient_d)
319
320 control_d, treatment_d = separate_control_treatment(patient_d)
321 #print("control", control_l)
322 print('-----')
323 print('-----')
324 print('-----')
325 print('-----')
326 print('-----')
327 #print("treatment", treatment_l)
328 #plot_patient_data(control_l)
329 #plot_patient_data(treatment_l)
330 if __name__ == '__main__':
331     main()

```

```

1 from cgi import test
2 from numpy import *
3 import sys
4 import matplotlib.pyplot as plt
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.naive_bayes import CategoricalNB
7 from sklearn.naive_bayes import GaussianNB
8 from sklearn import tree
9 from sklearn.ensemble import RandomForestClassifier
10 from sklearn import svm
11 from sklearn.cluster import KMeans
12 from scipy.cluster.hierarchy import complete, fcluster
13 from scipy.cluster import hierarchy
14 from scipy.spatial.distance import pdist
15 from sklearn.model_selection import KFold
16
17 patient_visit_dt = dtype([('study','U10'),('country','U10'),('txgroup','U10'),('assesmentid', float64),('patientid', float64),('visitday', int32),('xvalues',float64,(31)),('panss',float64),
18
19 def data_loader(fname):
20     #Study   Country PatientID      SiteID RaterID AssessmentID    TxGroup VisitDay      P1    P2    P3    P4    P5    P6    P7    N1    N2    N3    N4    N5
21     # 0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15     16     17     18     19     20     21     22     23     24     25     26     27
22     # G7     G8     G9     G10    G11    G12    G13    G14    G15    G16    PANSS_Total
23     # 28     29     30     31     32     33     34     35     36     37     38
24
25     if 'Study_E.csv' in fname:
26         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6), delimiter=',', dtype=str )
27     else:
28         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6,39), delimiter=',', dtype=str )
29         quant_data_a = loadtxt(fname,skiprows=1, usecols=(2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',', dtype=float64)
30
31
32
33 data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
34 #print(data_a.shape)
35
36
37 data_a['study'] = cata_data_a[:, 0]
38 data_a['country'] = cata_data_a[:, 1]
39 data_a['txgroup'] = cata_data_a[:, 2]
40 data_a['assesmentid'] = quant_data_a[:,3]
41 data_a['patientid'] = quant_data_a[:,0]
42 data_a['visitday'] = quant_data_a[:,4]
43
44
45 data_a['xvalues'] = quant_data_a[:, 5:36]
46 data_a['panss'] = quant_data_a[:,35]
47 if 'Study_E.csv' in fname:
48     data_a['leadstatus'] = 0
49 else:
50     data_a['leadstatus'] = cata_data_a[:,3]
51 return data_a
52
53 def trial():
54     num_l = []
55     for i in range(100):
56         num_l.append(i)
57     return num_l
58
59 def find_patients(data_a):
60     d = {}
61     for i in range(data_a.shape[0]):
62         key = data_a[i]['patientid']
63         if key in d:
64             d[key].append(i)
65         else:
66             d[key] = [i]
67
68 patient_d = {}
69 for patientid, index_l in d.items():
70     patient_a = data_a[index_l]
71     index_v = patient_a['visitday'].argsort()
72     patient_d[patientid] = patient_a[index_v]
73
74 print(type(patient_d))
75 return patient_d
76
77 def separate_control_treatment(patient_d):
78     control_d = {}
79     treatment_d = {}
80
81     for patient_a in patient_d.values():
82
83         if patient_a[0]['txgroup'] == '"Control"':
84             control_d[patient_a[0]['patientid']] = patient_a
85         else:
86             #print('test')
87             treatment_d[patient_a[0]['patientid']] = patient_a
88
89     return control_d, treatment_d
90
91
92 def plot_patient_data(data_l):
93     print(type(data_l))
94     print(len(data_l))
95
96     fig, ax = plt.subplots()
97
98     for patient_a in data_l:
99
100         x_l = patient_a['visitday']
101         y_l = patient_a['panss']
102         ax.plot(x_l, y_l)
103
104 plt.show()
105
106 def filter_patients(patient_d, day_limit=126):
107     print(type(patient_d))
108     print(patient_d)
109     #delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
110     delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
111     for patientid in delete_patient_l:
112         del patient_d[patientid]
113
114 def difference_in_fields(patient_d, field_name):
115     difference_values_l = []
116     for patient_a in patient_d.values():
117         dif1 = patient_a[-1][field_name]
118         dif2 = patient_a[0][field_name]
119         dif = dif1-dif2
120         #dif = patient_a[-1]['panss'] - patient_a[1]['panss']
121         difference_values_l.append(dif)
122     difference_values_a = array(difference_values_l)
123     return difference_values_a
124
125 def difference_in_scores(patient_d):
126     assert 0
127     difference_values_l = []
128     for patient_a in patient_d.values():
129         dif1 = patient_a[-1]['panss']
130         dif2 = patient_a[0]['panss']

```

```

130     dif = dif1-dif2
131     difference_values_l.append(dif)
132 return difference_values_l
133
134 def difference_in_scores_stats(difference_values_l):
135     mean_difference = mean(difference_values_l)
136     standev_difference = std(difference_values_l)
137     print(f'The mean difference is {mean_difference} and has a standard deviation of {standev_difference}')
138     return mean_difference, standev_difference
139
140 def knn_pred(xtrain_a, ytrain_v, xtest_a, ytest_v):
141     knn_model = KNeighborsClassifier(n_neighbors = 5)
142     knn_model.fit(xtrain_a, ytrain_v)
143     ytrainpred_v = knn_model.predict(xtrain_a)
144     ypred_v = knn_model.predict(xtest_a)
145     ypred_proba_a = knn_model.predict_proba(xtest_a)
146     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/(ytrainpred_v.shape[0])
147     accuracy_test = ((ytest_v==ypred_v).sum())/(ytest_v.shape[0])
148
149 def class_pred(clf, xtrain_a, ytrain_v, xtest_a, ytest_v):
150     clf.fit(xtrain_a, ytrain_v)
151     ytrainpred_v = clf.predict(xtrain_a)
152     ypred_v = clf.predict(xtest_a)
153     ypred_proba_a = clf.predict_proba(xtest_a)
154     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/(ytrainpred_v.shape[0])
155     accuracy_test = ((ytest_v==ypred_v).sum())/(ytest_v.shape[0])
156
157
158     # m_v = ypred_v == ytest_v
159     # print('match samples:', ypred_v[m_v])
160     # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
161
162     print(f'The training accuracy is: {accuracy_train} ')
163     print(f'The test accuracy is {accuracy_test}')
164
165     return ypred_proba_a
166
167 def z_score_converter(data_a):
168     train_data_a = data_a['xvalues']
169     #print(train_data_a)
170     mean_a = train_data_a.mean(axis=0)
171     standev_a = train_data_a.std(axis=0)
172     zscore = (train_data_a - mean_a)/standev_a
173     #print(zscore)
174     return zscore
175
176 def z_score_convert2(data_a, input_test_data_a):
177     train_data_a = data_a['xvalues']
178     test_data_a = input_test_data_a['xvalues']
179     #print(train_data_a)
180     train_mean_v = train_data_a.mean(axis=0)
181     train_standev_v = train_data_a.std(axis=0)
182     train_zscore_a = (train_data_a - train_mean_v)/train_standev_v
183     test_zscore_a = (test_data_a - train_mean_v)/train_standev_v
184     #print(zscore)
185     return train_zscore_a, test_zscore_a
186
187 def time_shifter(patient_d):
188     #adjusted_d = {}
189     for patient_id, visitday in patient_d.items():
190         time_zero = visitday[0]['visitday']
191         visitday[0] = 0
192         for other_days in visitday[1:]:
193             other_days['visitday']-=time_zero
194
195
196
197 def desired_data(patient_d):
198     predicted_data_l = []
199     for patient_id, array_values_a in patient_d.items():
200         last_array = array_values_a[-1]
201         output_data = last_array['panss']
202         predicted_data_l.append((patient_id, output_data))
203     submitted_data_a = array(predicted_data_l, output_data)
204     return submitted_data_a
205
206 def class_data_merge(test_data_a, maxproba_v):
207     #print("see assesment shapes", test_data_a['assesmentid'].shape)
208     class_proba_l = list(zip(test_data_a['assesmentid'],maxproba_v))
209     class_proba_a = array(class_proba_l)
210
211     return class_proba_a
212
213 def get_first_day(patient_d):
214     return array([patient_a[0]['xvalues'] for patient_a in patient_d.values()])
215
216
217 def k_means_clustering(x, n_clusters):
218     #best_k = max(range(2, 11), key=lambda k: np.mean(cross_val_score(KMeans(n_clusters=k), cv=10)))
219     kf = KFold(n_splits=10, shuffle=True, random_state=42)
220     kmeans = KMeans(n_clusters=n_clusters)
221     kmeans.fit(x)
222     cluster_labels = kmeans.labels_
223     cluster_centers = kmeans.cluster_centers_
224     plt.scatter(x[:, 0], x[:, 7], marker='o')
225     print("x", x)
226     #In this case x[:,0] and x[:,7] are the p1 and p1 values
227     plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], marker='X')
228     plt.show()
229     return cluster_labels, cluster_centers
230
231 def dendo_construct(x_a):
232     linkage_matrix = hierarchy.linkage(x_a, method='ward')
233
234     plt.figure(figsize=(10, 6))
235     dendrogram = hierarchy.dendrogram(linkage_matrix)
236     plt.show()
237
238 def save_file(patient_data_a, fname):
239     savetxt(fname, patient_data_a, delimiter=',')
240
241 def main():
242     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]])
243     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]])
244     test_data_a = data_loader(sys.argv[-1])
245
246     m_v = logical_not(train_data_a['leadstatus'] == "Passed")
247     train_data_a = train_data_a[m_v]
248     m_v = logical_not(test_data_a['leadstatus'] == "Passed")
249     test_data_a = test_data_a[m_v]
250
251     # print("seeing train after load", train_data_a.shape)
252     #print("seeing test after load", test_data_a.shape)
253
254     patientid_v = data_a['patientid']
255     upatientid_v = unique(patientid_v)
256     #print('Patient id:', upatientid_v.shape[0])
257
258     #print("just viewing", patientid_v)
259
260     #print(da)
261     patient_d = find_patients(data_a)
262
263     #print('-----')

```

```

264 control_d, treatment_d = seperate_control_treatment(patient_d)
265 #print("view control people", control_d)
266 #print(type(control_d))
267 #print("view treatment people", treatment_d)
268 #print(type(treatment_d))
269 #print('-----')
270
271 control_patientdiff = difference_in_fields(control_d, 'panss')
272 control_patientdiffstats = difference_in_scores_stats(control_patientdiff)
273 #print(type(control_patientdiff))
274
275 treatment_patientdiff = difference_in_fields(treatment_d, 'panss')
276 treatment_patientdiffstats = difference_in_scores_stats(treatment_patientdiff)
277
278 #print('-----')
279
280
281 print(f'Control Patients: {control_patientdiffstats}')
282 print(f'Treatment Patients: {treatment_patientdiffstats}')
283
284 print('-----')
285 #print(patient_d)
286 fname = "upload1.csv"
287 upload_data_a = desired_data(patient_d)
288 print(upload_data_a)
289 save_file(upload_data_a, fname)
290
291 print('-----')
292 zscore_train_a, zscore_test_a = z_score_convert2(train_data_a, test_data_a)
293 first_x_a = get_first_day(patient_d)
294 cluster_labels, cluster_centers = k_means_clustering(first_x_a,3)
295 print("cluster labels:", cluster_labels)
296 print("cluster centers:", cluster_centers)
297
298 #ypredproba_a = knn_pred(zscore_train_a, train_data_a['leadstatus'], zscore_test_a, test_data_a['leadstatus'])
299 #ypredproba_a = knn_pred(train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
300
301 #clf = KNeighborsClassifier(n_neighbors = 5)
302 #clf = CategoricalNB(force_alpha=True)
303 #clf = GaussianNB()
304 #clf = tree.DecisionTreeClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, criterion='entropy')
305 #clf = RandomForestClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, random_state=0)
306 clf = svm.SVC(probability=True)
307 ypredproba_a = class_pred(clf, train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
308
309
310 print(ypredproba_a)
311 print('-----')
312
313 max_proba_v = ypredproba_a.max(axis=1)
314 class_data_a = class_data_merge(test_data_a, max_proba_v)
315
316 fname = "classupload1.csv"
317 save_file(class_data_a, fname)
318
319 dendo_construct(get_first_day(patient_d))
320
321 raise SystemExit
322
323 # day_v = data_a['visitday']
324 # # uday_v = unique(day_v)
325 # week_v = uday_v/7
326 # print(uday_v)
327 # print('-----')
328 # print('-----')
329 # print(week_v)
330 # print('-----')
331
332 # print(f'data_a[10: {data_a[10:]}]')
333 m_v = data_a['txgroup'] == "Treatment"
334 day_v = data_a[m_v]['visitday']
335 print('treatment day', sorted(day_v))
336 uday_v = unique(day_v)
337 # week_v = uday_v//7
338 # print(f'uday_v treatments: {uday_v}')
339 # print(f'week_v treatments: {week_v}')
340 # #week_v = int32(week_v)
341 # print('week count:', list(zip(arange(week_v.shape[0]), bincount(week_v))))
342 patient_d = find_patients(data_a)
343 print('Before delete:', len(patient_d))
344 filter_patients(patient_d, day_limit=105)
345 print('After delete:', len(patient_d))
346
347 #print(len(a))
348 print('-----')
349 #print(d)
350 print(patient_d)
351
352 control_d, treatment_d = seperate_control_treatment(patient_d)
353 #print("control", control_l)
354 print('-----')
355 print('-----')
356 print('-----')
357 print('-----')
358 print('-----')
359 #print("treatment", treatment_l)
360 #plot_patient_data(control_l)
361 #plot_patient_data(treatment_l)
362 if __name__ == '__main__':
363     main()

```

```
# realp3.py
```

```
1 from cgi import test
2 from numpy import *
3 import sys
4 import matplotlib.pyplot as plt
5 from sklearn.neighbors import KNeighborsClassifier
6 from sklearn.naive_bayes import CategoricalNB
7 from sklearn.naive_bayes import GaussianNB
8 from sklearn import tree
9 from sklearn.ensemble import RandomForestClassifier
10
11 patient_visit_dt = dtype([('study','U10'),('country','U10'),('txgroup','U10'),('assesmentid', float64),('patientid', float64),('visitday', int32),('xvalues',float64,(30)),('panss',float64),
12
13 def data_loader(fname):
14     #Study   Country PatientID      SiteID RaterID AssessmentID   TxGroup VisitDay      P1   P2   P3   P4   P5   P6   P7   N1   N2   N3   N4   N5
15     # 0      1      2      3      4      5      6      7      8      9      10     11     12     13     14     15     16     17     18     19     20     21     22     23     24     25     26     27
16     # G7    G8    G9    G10   G11   G12   G13   G14   G15   G16   PANSS_Total
17     # 28   29   30   31   32   33   34   35   36   37   38
18
19     if 'Study_E.csv' in fname:
20         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6), delimiter=',', dtype=str )
21     else:
22         cata_data_a = loadtxt(fname,skiprows=1, usecols=(0,1,6,39), delimiter=',', dtype=str )
23     quant_data_a = loadtxt(fname,skiprows=1, usecols=(2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38), delimiter=',', dtype=float64)
24
25
26     data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
27     #print(data_a.shape)
28
29
30     data_a[['study']] = cata_data_a[:, 0]
31     data_a[['country']] = cata_data_a[:, 1]
32     data_a[['txgroup']] = cata_data_a[:, 2]
33     data_a[['assesmentid']] = quant_data_a[:,3]
34     data_a[['patientid']] = quant_data_a[:,0]
35     data_a[['visitday']] = quant_data_a[:,4]
36
37
38     data_a[['xvalues']] = quant_data_a[:,5:35]
39     data_a[['panss']] = quant_data_a[:,35]
40     if 'Study_E.csv' in fname:
41         data_a[['leadstatus']] = 0
42     else:
43         data_a[['leadstatus']] = cata_data_a[:,3]
44
45     return data_a
46
47 def trial():
48     num_l = []
49     for i in range(100):
50         num_l.append(i)
51     return num_l
52
53 def find_patients(data_a):
54     d = {}
55     for i in range(data_a.shape[0]):
56         key = data_a[i]['patientid']
57         if key in d:
58             d[key].append(i)
59         else:
60             d[key] = [i]
61
62     patient_d = {}
63     for patientid, index_l in d.items():
64         patient_a = data_a[index_l]
65         index_v = patient_a['visitday'].argsort()
66         patient_d[patientid] = patient_a[index_v]
67
68     print(type(patient_d))
69     return patient_d
70
71 def separate_control_treatment(patient_d):
72     control_d = {}
73     treatment_d = {}
74
75     for patient_a in patient_d.values():
76
77         if patient_a[0]['txgroup'] == 'Control':
78             control_d[patient_a[0]['patientid']] = patient_a
79         else:
80             #print('test')
81             treatment_d[patient_a[0]['patientid']] = patient_a
82
83     return control_d, treatment_d
84
85
86 def plot_patient_data(data_l):
87     print(type(data_l))
88     print(len(data_l))
89
90     fig, ax = plt.subplots()
91
92     for patient_a in data_l:
93
94         x_l = patient_a['visitday']
95         y_l = patient_a['panss']
96         ax.plot(x_l, y_l)
97     plt.show()
98
99 def filter_patients(patient_d, day_limit=126):
100    print(type(patient_d))
101    print(patient_d)
102    #delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
103    delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
104    for patientid in delete_patient_l:
105        del patient_d[patientid]
106
107 def difference_in_fields(patient_d, field_name):
108     difference_values_l = []
109     for patient_a in patient_d.values():
110         dif1 = patient_a[1][field_name]
111         dif2 = patient_a[0][field_name]
112         dif = dif1-dif2
113         #dif = patient_a[-1]['panss'] - patient_a[1]['panss']
114         difference_values_l.append(dif)
115     difference_values_a = array(difference_values_l)
116     return difference_values_a
117
118 def difference_in_scores(patient_d):
119     assert 0
120     difference_values_l = []
121     for patient_a in patient_d.values():
122         dif1 = patient_a[1]['panss']
123         dif2 = patient_a[0]['panss']
124         dif = dif1-dif2
125         difference_values_l.append(dif)
126     return difference_values_l
127
128 def difference_in_scores_stats(difference_values_l):
129     mean_difference = mean(difference_values_l)
```

```

130     standev_difference = std(difference_values_1)
131     print(f'The mean difference is {mean_difference} and has a standard deviation of {standev_difference}')
132     return mean_difference, standev_difference
133
134 def knn_pred(xtrain_a, ytrain_v, xtest_a, ytest_v):
135     knn_model = KNeighborsClassifier(n_neighbors = 5)
136     knn_model.fit(xtrain_a, ytrain_v)
137     ytrainpred_v = knn_model.predict(xtrain_a)
138     ypred_v = knn_model.predict(xtest_a)
139     ypred_proba_a = knn_model.predict_proba(xtest_a)
140     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
141     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
142
143 def class_pred(clf, xtrain_a, ytrain_v, xtest_a, ytest_v):
144     clf.fit(xtrain_a, ytrain_v)
145     ytrainpred_v = clf.predict(xtrain_a)
146     ypred_v = clf.predict(xtest_a)
147     ypred_proba_a = clf.predict_proba(xtest_a)
148     accuracy_train = ((ytrain_v==ytrainpred_v).sum())/ytrainpred_v.shape[0]
149     accuracy_test = ((ytest_v==ypred_v).sum())/ytest_v.shape[0]
150
151
152     # m_v = ypred_v == ytest_v
153     # print('match samples:', ypred_v[m_v])
154     # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
155
156     print(f'The training accuracy is: {accuracy_train} ')
157     print(f'The test accuracy is {accuracy_test}')
158     return ypred_proba_a
159
160 def z_score_converter(data_a):
161     train_data_a = data_a['xvalues']
162     #print(train_data_a)
163     mean_a = train_data_a.mean(axis=0)
164     standev_a = train_data_a.std(axis=0)
165     zscore = (train_data_a - mean_a)/standev_a
166     #print(zscore)
167     return zscore
168
169 def z_score_convert2(data_a, input_test_data_a):
170     train_data_a = data_a['xvalues']
171     test_data_a = input_test_data_a['xvalues']
172     #print(train_data_a)
173     train_mean_v = train_data_a.mean(axis=0)
174     train_standev_v = train_data_a.std(axis=0)
175     train_zscore_a = (train_data_a - train_mean_v)/train_standev_v
176     test_zscore_a = (test_data_a - train_mean_v)/train_standev_v
177     #print(zscore)
178     return train_zscore_a, test_zscore_a
179
180 def time_shifter(patient_d):
181     #adjusted_d = {}
182     for patient_id, visitday in patient_d.items():
183         time_zero = visitday[0]['visitday']
184         visitday[0] = 0
185         for other_days in visitday[1:]:
186             other_days['visitday']-=time_zero
187
188
189
190 def desired_data(patient_d):
191     predicted_data_l = []
192     for patient_id, array_values_a in patient_d.items():
193         last_array = array_values_a[-1]
194         output_data = last_array['pans']
195         predicted_data_l.append((patient_id, output_data))
196     submitted_data_a = array(predicted_data_l, output_data)
197     return submitted_data_a
198
199 def class_data_merge(test_data_a, maxproba_v):
200     #print("see assessment shapes", test_data_a['assessmentid'].shape)
201     class_proba_l = list(zip(test_data_a['assessmentid'],maxproba_v))
202     class_proba_a = array(class_proba_l)
203
204     return class_proba_a
205
206 def save_file(patient_data_a, fname):
207     savetxt(fname, patient_data_a, delimiter=',')
208
209 def main():
210     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]])
211     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]])
212     test_data_a = data_loader(sys.argv[-1])
213
214     m_v = logical_not(train_data_a['leadstatus']) == "Passed"
215     train_data_a = train_data_a[m_v]
216     m_v = logical_not(test_data_a['leadstatus']) == "Passed"
217     test_data_a = test_data_a[m_v]
218
219
220     # print("seeing train after load", train_data_a.shape)
221     #print("seeing test after load", test_data_a.shape)
222
223     patientid_v = data_a['patientid']
224     upatientid_v = unique(patientid_v)
225     #print('Patient id:', upatientid_v.shape[0])
226
227     #print("just viewing", patientid_v)
228
229     #print(da)
230     patient_d = find_patients(data_a)
231
232     #print('-----')
233
234     control_d, treatment_d = separate_control_treatment(patient_d)
235     #print("view control people", control_d)
236     #print(type(control_d))
237     #print("view treatment people", treatment_d)
238     #print(type(treatment_d))
239     #print(type(control_d))
240     #print('-----')
241
242     control_patientdiff = difference_in_fields(control_d, 'pans')
243     plt.hist(control_patientdiff, bins = 10)
244     plt.xlabel('Newest Measurement - Oldest Measurement')
245     plt.ylabel('Frequency')
246     plt.title('Histogram of Differences in Control Group')
247     plt.show()
248     control_patientdiffstats = difference_in_scores_stats(control_patientdiff)
249     #print(type(control_patientdiff))
250
251     treatment_patientdiff = difference_in_fields(treatment_d, 'pans')
252     plt.hist(treatment_patientdiff, bins = 10)
253     plt.xlabel('Newest Measurement - Oldest Measurement')
254     plt.ylabel('Frequency')
255     plt.title('Histogram of Differences in Treatment Group')
256     plt.show()
257     treatment_patientdiffstats = difference_in_scores_stats(treatment_patientdiff)
258
259     #print('-----')
260
261     print(f'Control Patients: {control_patientdiffstats}')
262     print(len(control_d))
263

```

```

264 print(f'Treatment Patients: {treatment_patientdiffstats}')
265 print(len(treatment_d))
266
267 print('-----')
268 #print(patient_d)
269 fname = "upload1.csv"
270 upload_data_a = desired_data(patient_d)
271 print(upload_data_a)
272 save_file(upload_data_a, fname)
273 print('-----')
274 zscore_train_a, zscore_test_a = z_score_convert2(train_data_a, test_data_a)
275
276
277
278 #ypredproba_a = knn_pred(zscore_train_a, train_data_a['leadstatus'], zscore_test_a, test_data_a['leadstatus'])
279 #ypredproba_a = knn_pred(train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
280
281 #clf = KNeighborsClassifier(n_neighbors = 5)
282 #clf = CategoricalNB(force_alpha=True)
283 #clf = GaussianNB()
284 #clf = tree.DecisionTreeClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, criterion='entropy')
285 #clf = RandomForestClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, random_state=0)
286 ypredproba_a = class_pred(clf, train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
287
288
289 print(ypredproba_a)
290 print('-----')
291
292 max_proba_v = ypredproba_a.max(axis=1)
293 class_data_a = class_data_merge(test_data_a, max_proba_v)
294
295 fname = "classupload1.csv"
296 save_file(class_data_a, fname)
297 raise SystemExit
298 plot_patient_data(control_d)
299 plot_patient_data(treatment_d)
300 raise SystemExit
301
302 # day_v = data_a['visitday']
303 # # uday_v = unique(day_v)
304 # week_v = uday_v/7
305 # print(uday_v)
306 # print('-----')
307 # print('-----')
308 # print(week_v)
309 # print('-----')
310 # print('-----')
311
312 # print(f'data_a[10: {data_a[10:]}]')
313 m_v = data_a['txgroup'] == 'Treatment'
314 day_v = data_a[m_v]['visitday']
315 print('treatment day', sorted(day_v))
316 uday_v = unique(day_v)
317 # week_v = uday_v//7
318 # print(f'uday_v treatments: {uday_v}')
319 # print(f'week_v treatments: {week_v}')
320 # #week_v = int32(week_v)
321 # print('week count:', list(zip(arange(week_v.shape[0]), bincount(week_v))))
322 patient_d = find_patients(data_a)
323 print('Before delete:', len(patient_d))
324 filter_patients(patient_d, day_limit=105)
325 print('After delete:', len(patient_d))
326
327 #print(len(a))
328 print('-----')
329 #print(d)
330 print(patient_d)
331
332 control_d, treatment_d = seperate_control_treatment(patient_d)
333 #print("control", control_l)
334 print('-----')
335 print('-----')
336 print('-----')
337 print('-----')
338 print('-----')
339 #print("treatment", treatment_l)
340 plot_patient_data(control_l)
341 plot_patient_data(treatment_l)
342 if __name__ == '__main__':
343     main()

```