

Final Project

| | |
|-------------------------|---|
| finalproj/ | 1 |
| └─ finalp3.py | 1 |
| └─ p3class.py | 3 |
| └─ prob2.py | 6 |
| └─ realp3.py | 9 |

final3.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, (30)), ('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
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.shape, quant_data_a.shape)
16    #print(type(cata_data_a), type(quant_data_a))
17
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(1)
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 seperate_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 def main():
123     #data_a = hstack([data_loader(fname) for fname in sys.argv[1:]])
124
125     data_a, cata_data_a, quant_data_a = data_loader("Study_E.csv")
126     patientid_v = data_a['patientid']
127     upatientid_v = unique(patientid_v)
128     print('Patient id:', upatientid_v.shape[0])
129
130     print("just viewing", patientid_v)
131     #print(da)
```

```

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(f'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 = separate_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()

```

```

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

```

```

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

```

```

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 = seperate_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()

```

prob2.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 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'),('assessmentid', 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     data_a = empty(quant_data_a.shape[0], dtype=patient_visit_dt)
33     #print(data_a.shape)
34
35
36     data_a['study'] = cata_data_a[:, 0]
37     data_a['country'] = cata_data_a[:, 1]
38     data_a['txgroup'] = cata_data_a[:, 2]
39     data_a['assessmentid'] = quant_data_a[:, 3]
40     data_a['patientid'] = quant_data_a[:, 0]
41     data_a['visitday'] = quant_data_a[:, 4]
42
43
44     data_a['xvalues'] = quant_data_a[:, 5:36]
45     data_a['panss'] = quant_data_a[:, 35]
46     if 'Study_E.csv' in fname:
47         data_a['leadstatus'] = 0
48     else:
49         data_a['leadstatus'] = cata_data_a[:, 3]
50     return data_a
51
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 seperate_control_treatment(patient_d):
78     control_d = {}
79     treatment_d = {}
80
81     for patient_a in patient_d.values():
82         if patient_a[0]['txgroup'] == "Control":
83             control_d[patient_a[0]['patientid']] = patient_a
84         else:
85             #print('test')
86             treatment_d[patient_a[0]['patientid']] = patient_a
87
88     return control_d, treatment_d
89
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     plt.show()
104
105 def filter_patients(patient_d, day_limit=126):
106     print(type(patient_d))
107     print(patient_d)
108     #delete patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] < day_limit]
109     delete_patient_l = [patientid for patientid, patient_a in patient_d.items() if patient_a[-1]['visitday'] - patient_a[0]['visitday'] < day_limit]
110     for patientid in delete_patient_l:
111         del patient_d[patientid]
112
113 def difference_in_fields(patient_d, field_name):
114     difference_values_l = []
115     for patient_a in patient_d.values():
116         dif1 = patient_a[-1][field_name]
117         dif2 = patient_a[0][field_name]
118         dif = dif1-dif2
119         #dif = patient_a[-1]['panss'] - patient_a[1]['panss']
120         difference_values_l.append(dif)
121     difference_values_a = array(difference_values_l)
122     return difference_values_a
123
124 def difference_in_scores(patient_d):
125     assert 0
126     difference_values_l = []
127     for patient_a in patient_d.values():
128         dif1 = patient_a[-1]['panss']
129         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
159     # m_v = ypred_v == ytest_v
160     # print('match samples:', ypred_v[m_v])
161     # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
162
163     print(f'The training accuracy is: {accuracy_train} ')
164     print(f'The test accuracy is {accuracy_test}')
165     return ypred_proba_a
166
167 def z_score_conver(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['panns']
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['assessmentid'].shape)
208     class_proba_l = list(zip(test_data_a['assessmentid'], 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 n1 values
227     plt.scatter(kmeans.cluster_centers[:, 0], kmeans.cluster_centers[:, 1], marker='X')
228     plt.show()
229     return cluster_labels, cluster_centers
230 def dendro_construct(x_a):
231     linkage_matrix = hierarchy.linkage(x_a, method='ward')
232
233     plt.figure(figsize=(10, 6))
234     dendrogram = hierarchy.dendrogram(linkage_matrix)
235     plt.show()
236 def save_file(patient_data_a, fname):
237     savetxt(fname, patient_data_a, delimiter=',')
238
239 def main():
240     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]]])
241     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]])
242     test_data_a = data_loader(sys.argv[-1])
243
244     m_v = logical_not(train_data_a['leadstatus'] == "Passed")
245     train_data_a = train_data_a[m_v]
246     m_v = logical_not(test_data_a['leadstatus'] == "Passed")
247     test_data_a = test_data_a[m_v]
248
249
250     # print("seeing train after load", train_data_a.shape)
251     #print("seeing test after load", test_data_a.shape)
252
253     patientid_v = data_a['patientid']
254     upatientid_v = unique(patientid_v)
255     #print('Patient id:', upatientid_v.shape[0])
256
257     #print("just viewing", patientid_v)
258
259     #print(da)
260     patient_d = find_patients(data_a)
261
262     #print('-----')
263

```

```

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'), ('assessmentid', 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['assessmentid'] = 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     return data_a
45
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         if patient_a[0]['txgroup'] == "Control":
77             control_d[patient_a[0]['patientid']] = patient_a
78         else:
79             #print('test')
80             treatment_d[patient_a[0]['patientid']] = patient_a
81
82     return control_d, treatment_d
83
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)
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
153 # m_v = ypred_v == ytest_v
154 # print('match samples:', ypred_v[m_v])
155 # print('mismatch samples:', list(zip(ypred_v[logical_not(m_v)], ytest_v[logical_not(m_v)])))
156
157 print(f'The training accuracy is: {accuracy_train} ')
158 print(f'The test accuracy is {accuracy_test}')
159 return ypred_proba_a
160
161 def z_score_conver(data_a):
162     train_data_a = data_a['xvalues']
163     #print(train_data_a)
164     mean_a = train_data_a.mean(axis=0)
165     standev_a = train_data_a.std(axis=0)
166     zscore = (train_data_a - mean_a)/standev_a
167     #print(zscore)
168     return zscore
169
170 def z_score_convert2(data_a, input_test_data_a):
171     train_data_a = data_a['xvalues']
172     test_data_a = input_test_data_a['xvalues']
173     #print(train_data_a)
174     train_mean_v = train_data_a.mean(axis=0)
175     train_standev_v = train_data_a.std(axis=0)
176     train_zscore_a = (train_data_a - train_mean_v)/train_standev_v
177     test_zscore_a = (test_data_a - train_mean_v)/train_standev_v
178     #print(zscore)
179     return train_zscore_a, test_zscore_a
180
181 def time_shifter(patient_d):
182     #adjusted d = {}
183     for patient_id, visitday in patient_d.items():
184         time_zero = visitday[0]['visitday']
185         visitday[0] = 0
186         for other_days in visitday[1:]:
187             other_days['visitday'] -= time_zero
188
189
190
191 def desired_data(patient_d):
192     predicted_data_l = []
193     for patient_id, array_values_a in patient_d.items():
194         last_array = array_values_a[-1]
195         output_data = last_array['panss']
196         predicted_data_l.append((patient_id, output_data))
197     submitted_data_a = array(predicted_data_l, output_data)
198     return submitted_data_a
199
200 def class_data_merge(test_data_a, maxproba_v):
201     #print('see assesment shapes', test_data_a['assessmentid'].shape)
202     class_proba_l = list(zip(test_data_a['assessmentid'], maxproba_v))
203     class_proba_a = array(class_proba_l)
204
205     return class_proba_a
206
207 def save_file(patient_data_a, fname):
208     savetxt(fname, patient_data_a, delimiter=',')
209
210 def main():
211     data_a = hstack([data_loader(fname) for fname in sys.argv[1:]]
212     train_data_a = hstack([data_loader(fname) for fname in sys.argv[1:-1]]
213     test_data_a = data_loader(sys.argv[-1])
214
215     m_v = logical_not(train_data_a['leadstatus'] == "Passed")
216     train_data_a = train_data_a[m_v]
217     m_v = logical_not(test_data_a['leadstatus'] == "Passed")
218     test_data_a = test_data_a[m_v]
219
220
221 # print("seeing train after load", train_data_a.shape)
222 # print("seeing test after load", test_data_a.shape)
223
224 patientid_v = data_a['patientid']
225 upatientid_v = unique(patientid_v)
226 #print('Patient id:', upatientid_v.shape[0])
227
228 #print("just viewing", patientid_v)
229
230 #print(da)
231 patient_d = find_patients(data_a)
232
233 #print('-----')
234
235 control_d, treatment_d = separte_control_treatment(patient_d)
236 #print("view control people", control_d)
237 #print(type(control_d))
238 #print("view treatment people", treatment_d)
239 #print(type(control_d))
240 #print('-----')
241
242 control_patientdiff = difference_in_fields(control_d, 'panss')
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, 'panss')
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
279 #ypredproba_a = knn_pred(zscore_train_a, train_data_a['leadstatus'], zscore_test_a, test_data_a['leadstatus'])
280 #ypredproba_a = knn_pred(train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
281
282 #clf = KNeighborsClassifier(n_neighbors = 5)
283 #clf = CategoricalNB(force_alpha=True)
284 #clf = GaussianNB()
285 #clf = tree.DecisionTreeClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, criterion='entropy')
286 clf = RandomForestClassifier(max_depth=8, min_samples_leaf=10, ccp_alpha=0.005, random_state=0)
287 ypredproba_a = class_pred(clf, train_data_a['xvalues'], train_data_a['leadstatus'], test_data_a['xvalues'], test_data_a['leadstatus'])
288
289
290 print(ypredproba_a)
291 print('-----')
292
293 max_proba_v = ypredproba_a.max(axis=1)
294 class_data_a = class_data_merge(test_data_a, max_proba_v)
295
296 fname = "classupload1.csv"
297 save_file(class_data_a, fname)
298 raise SystemExit
299 plot_patient_data(control_d)
300 plot_patient_data(treatment_d)
301 raise SystemExit
302
303 # day_v = data_a['visitday']
304 # # uday_v = unique(day_v)
305 # week_v = uday_v//7
306 # print(uday_v)
307 # print('-----')
308 # print('-----')
309 # print(week_v)
310 # print('-----')
311
312 # print(f'data a:10 {data_a[10:]}\n')
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()

```