final

December 7, 2018

```
In [2]: import pandas as pd
        import numpy as np
        data = pd.read_csv("a_affirmative.csv")
        data = data.drop(data.columns[0], axis=1)
```

```
In [3]: from sklearn.model_selection import train_test_split
        train, test = train_test_split(data, test_size=0.25)
        X = train.drop(train.columns[-1], axis=1)
        y = train[train.columns[-1]]
        y_true = test[test.columns[-1]]
        X_test = test.drop(train.columns[-1], axis=1)
```

```
In [4]: def print_stats(model):
            print("Statistics on Testing")
            y_pred = model.predict(X_test)
            cm = confusion_matrix(y_true, y_pred)
            tn, fp, fn, tp = cm.ravel()
            print("True Negative:", tn, "False Positives:", fp, "False Negatives:", fn, "True 
            accuracy = model.score(X_test, y_true)
            error = 1 - accuracy
            print("Accuracy:", accuracy, "Error:", error)
            precision = tp / (tp + fp)
            recall = tp / (tp + fn)
            f1 = 2 * (precision * recall) / (precision + recall)
            print("Precision:", precision, "Recall:", recall, "F1:", f1)
            print("Statistics on Training")
            y_pred = model.predict(X)
            cm = confusion_matrix(y, y_pred)
            tn, fp, fn, tp = cm.ravel()
            print("True Negative:", tn, "False Positives:", fp, "False Negatives:", fn, "True 
            accuracy = model.score(X, y)
            error = 1 - accuracy
            print("Accuracy:", accuracy, "Error:", error)
            precision = tp / (tp + fp)
            recall = tp / (tp + fn)
            f1 = 2 * (precision * recall) / (precision + recall)
            print("Precision:", precision, "Recall:", recall, "F1:", f1)
In [5]: from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import confusion_matrix
        model = RandomForestClassifier()
        model.fit(X,y)
        print_stats(model)

Statistics on Testing
True Negative: 165 False Positives: 4 False Negatives: 17 True Positives: 80
Accuracy: 0.9210526315789473 Error: 0.07894736842105265
Precision: 0.9523809523809523 Recall: 0.8247422680412371 F1: 0.883977900552486
Statistics on Training
True Negative: 478 False Positives: 1 False Negatives: 3 True Positives: 314
Accuracy: 0.9949748743718593 Error: 0.005025125628140725
Precision: 0.9968253968253968 Recall: 0.9905362776025236 F1: 0.9936708860759493


/usr/local/Cellar/python3/3.6.4_2/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)


In [6]: model2 = RandomForestClassifier(n_estimators=50)
        model2.fit(X,y)
        print_stats(model2)

Statistics on Testing
True Negative: 163 False Positives: 6 False Negatives: 13 True Positives: 84
Accuracy: 0.9285714285714286 Error: 0.0714285714285714
Precision: 0.9333333333333333 Recall: 0.865979381443299 F1: 0.8983957219251337
Statistics on Training
True Negative: 479 False Positives: 0 False Negatives: 0 True Positives: 317
Accuracy: 1.0 Error: 0.0
Precision: 1.0 Recall: 1.0 F1: 1.0


In [7]: importances = model.feature_importances_
        indices = np.argsort(importances)[::-1]
        col = list(X)
        for f in range(X.shape[1]):
            print("%d. feature %s (%f)" % (f + 1, col[indices[f]], importances[indices[f]]))

1. feature 56y (0.079645)
2. feature 51y (0.044687)
3. feature 64y (0.044446)
4. feature 65y (0.043404)
5. feature 46y (0.042312)
6. feature 37y (0.042229)
7. feature 5y (0.039268)
8. feature 44y (0.038877)
9. feature 45y (0.038641)
```

```
10. feature 48z (0.015025)
11. feature 43y (0.013527)
12. feature 73x (0.011465)
13. feature 60z (0.010808)
14. feature 98y (0.010480)
15. feature 35x (0.009939)
16. feature 8y (0.009609)
17. feature 76x (0.009079)
18. feature 30y (0.008910)
19. feature 0y (0.008857)
20. feature 22y (0.008350)
21. feature 50y (0.007831)
22. feature 3z (0.007378)
23. feature 42y (0.007099)
24. feature 75y (0.007084)
25. feature 59y (0.007067)
26. feature 82y (0.006976)
27. feature 68x (0.006865)
28. feature 76y (0.006756)
29. feature 57y (0.006749)
30. feature 74z (0.006724)
31. feature 55y (0.006434)
32. feature 72x (0.006159)
33. feature 19z (0.005791)
34. feature 79z (0.005738)
35. feature 91y (0.005467)
36. feature 81y (0.005357)
37. feature 34x (0.005133)
38. feature 60x (0.005077)
39. feature 54x (0.005066)
40. feature 78y (0.005049)
41. feature 92y (0.004982)
42. feature 70x (0.004961)
43. feature 4x (0.004857)
44. feature 16y (0.004837)
45. feature 25y (0.004768)
46. feature 15y (0.004731)
47. feature 44x (0.004567)
48. feature 39y (0.004520)
49. feature 99y (0.004470)
50. feature 13y (0.004430)
51. feature 41y (0.004326)
52. feature 40y (0.004156)
53. feature 19y (0.004007)
54. feature 20y (0.003925)
55. feature 99x (0.003908)
56. feature 60y (0.003855)
57. feature 48x (0.003703)
```

```
58. feature 51x (0.003686)
59. feature 35y (0.003627)
60. feature 31y (0.003602)
61. feature 29y (0.003598)
62. feature 81z (0.003562)
63. feature 63y (0.003510)
64. feature 75x (0.003503)
65. feature 97y (0.003487)
66. feature 43z (0.003478)
67. feature 25z (0.003450)
68. feature 49z (0.003418)
69. feature 87y (0.003316)
70. feature 95y (0.003313)
71. feature 33y (0.003275)
72. feature 6y (0.003257)
73. feature 84z (0.003195)
74. feature 88y (0.003118)
75. feature 98x (0.003032)
76. feature 91z (0.003004)
77. feature 89y (0.002973)
78. feature 73y (0.002964)
79. feature 90y (0.002900)
80. feature 62y (0.002886)
81. feature 64z (0.002858)
82. feature 70y (0.002848)
83. feature 79x (0.002805)
84. feature 4y (0.002738)
85. feature 48y (0.002699)
86. feature 8z (0.002642)
87. feature 15x (0.002546)
88. feature 74x (0.002507)
89. feature 36x (0.002506)
90. feature 85y (0.002493)
91. feature 10y (0.002489)
92. feature 54z (0.002483)
93. feature 90z (0.002468)
94. feature 14y (0.002442)
95. feature 66z (0.002437)
96. feature 32y (0.002414)
97. feature 67x (0.002413)
98. feature 2x (0.002388)
99. feature 1x (0.002378)
100. feature 58x (0.002369)
101. feature 52y (0.002318)
102. feature 69z (0.002260)
103. feature 16z (0.002238)
104. feature 78x (0.002220)
105. feature 41z (0.002151)
```

```
106. feature 3x (0.002141)
107. feature 39x (0.002129)
108. feature 52x (0.002126)
109. feature 84x (0.002114)
110. feature 23y (0.002104)
111. feature 32x (0.002077)
112. feature 77y (0.002069)
113. feature 69y (0.002041)
114. feature 97z (0.002024)
115. feature 52z (0.002024)
116. feature 36y (0.001952)
117. feature 98z (0.001946)
118. feature 26y (0.001937)
119. feature 47x (0.001933)
120. feature 96x (0.001932)
121. feature 67y (0.001927)
122. feature 26x (0.001881)
123. feature 15z (0.001842)
124. feature 38x (0.001810)
125. feature 27y (0.001763)
126. feature 7y (0.001709)
127. feature 92x (0.001691)
128. feature 94x (0.001689)
129. feature 90x (0.001679)
130. feature 1z (0.001664)
131. feature 80x (0.001662)
132. feature 17z (0.001610)
133. feature 93y (0.001608)
134. feature 85x (0.001596)
135. feature 96y (0.001577)
136. feature 50x (0.001543)
137. feature 86y (0.001531)
138. feature 12x (0.001488)
139. feature 9x (0.001450)
140. feature 73z (0.001449)
141. feature 28y (0.001405)
142. feature 24y (0.001401)
143. feature 20x (0.001398)
144. feature 71x (0.001398)
145. feature 11y (0.001379)
146. feature 7x (0.001376)
147. feature 12y (0.001359)
148. feature 86x (0.001358)
149. feature 49y (0.001356)
150. feature 37z (0.001318)
151. feature 28x (0.001271)
152. feature 72y (0.001254)
153. feature 92z (0.001254)
```

```
154. feature 80y (0.001243)
155. feature 19x (0.001236)
156. feature 18z (0.001218)
157. feature 68y (0.001212)
158. feature 47y (0.001187)
159. feature 84y (0.001175)
160. feature 89x (0.001129)
161. feature 81x (0.001120)
162. feature 71y (0.001110)
163. feature 62z (0.001090)
164. feature 56z (0.001078)
165. feature 34y (0.001058)
166. feature 39z (0.001044)
167. feature 14x (0.000978)
168. feature 55x (0.000959)
169. feature 32z (0.000937)
170. feature 30x (0.000930)
171. feature 38y (0.000923)
172. feature 54y (0.000921)
173. feature 65x (0.000912)
174. feature 8x (0.000906)
175. feature 22x (0.000880)
176. feature 53y (0.000863)
177. feature 67z (0.000831)
178. feature 74y (0.000829)
179. feature 55z (0.000800)
180. feature 64x (0.000787)
181. feature 77z (0.000755)
182. feature 78z (0.000751)
183. feature 21y (0.000725)
184. feature 94y (0.000704)
185. feature 41x (0.000703)
186. feature 27x (0.000701)
187. feature 99z (0.000692)
188. feature 75z (0.000691)
189. feature 0x (0.000670)
190. feature 24x (0.000660)
191. feature 33x (0.000643)
192. feature 30z (0.000638)
193. feature 97x (0.000593)
194. feature 83x (0.000522)
195. feature 40z (0.000522)
196. feature 89z (0.000518)
197. feature 3y (0.000512)
198. feature 42z (0.000510)
199. feature 31x (0.000499)
200. feature 69x (0.000493)
201. feature 25x (0.000489)
```

```
202. feature 56x (0.000482)
203. feature 27z (0.000482)
204. feature 86z (0.000480)
205. feature 10x (0.000472)
206. feature 23x (0.000471)
207. feature 66y (0.000447)
208. feature 13x (0.000439)
209. feature 58y (0.000420)
210. feature 37x (0.000416)
211. feature 18x (0.000405)
212. feature 49x (0.000402)
213. feature 2z (0.000402)
214. feature 26z (0.000400)
215. feature 62x (0.000394)
216. feature 57z (0.000392)
217. feature 53z (0.000387)
218. feature 77x (0.000353)
219. feature 57x (0.000351)
220. feature 38z (0.000329)
221. feature 21x (0.000264)
222. feature 36z (0.000236)
223. feature 5z (0.000212)
224. feature 82z (0.000161)
225. feature 43x (0.000161)
226. feature 0z (0.000135)
227. feature 6x (0.000122)
228. feature 93x (0.000113)
229. feature 35z (0.000109)
230. feature 24z (0.000097)
231. feature 7z (0.000089)
232. feature 79y (0.000060)
233. feature 59x (0.000056)
234. feature 63z (0.000037)
235. feature 70z (0.000019)
236. feature 9z (0.000000)
237. feature 87z (0.000000)
238. feature 9y (0.000000)
239. feature 83y (0.000000)
240. feature 10z (0.000000)
241. feature 83z (0.000000)
242. feature 85z (0.000000)
243. feature 34z (0.000000)
244. feature 11x (0.000000)
245. feature 44z (0.000000)
246. feature 87x (0.000000)
247. feature 53x (0.000000)
248. feature 42x (0.000000)
249. feature 88x (0.000000)
```

```
250. feature 6z (0.000000)
251. feature 88z (0.000000)
252. feature 5x (0.000000)
253. feature 4z (0.000000)
254. feature 91x (0.000000)
255. feature 93z (0.000000)
256. feature 94z (0.000000)
257. feature 40x (0.000000)
258. feature 2y (0.000000)
259. feature 95x (0.000000)
260. feature 95z (0.000000)
261. feature 1y (0.000000)
262. feature 96z (0.000000)
263. feature 11z (0.000000)
264. feature 33z (0.000000)
265. feature 82x (0.000000)
266. feature 12z (0.000000)
267. feature 65z (0.000000)
268. feature 63x (0.000000)
269. feature 20z (0.000000)
270. feature 50z (0.000000)
271. feature 21z (0.000000)
272. feature 61z (0.000000)
273. feature 61y (0.000000)
274. feature 22z (0.000000)
275. feature 29z (0.000000)
276. feature 61x (0.000000)
277. feature 23z (0.000000)
278. feature 51z (0.000000)
279. feature 59z (0.000000)
280. feature 29x (0.000000)
281. feature 58z (0.000000)
282. feature 66x (0.000000)
283. feature 47z (0.000000)
284. feature 31z (0.000000)
285. feature 45z (0.000000)
286. feature 28z (0.000000)
287. feature 80z (0.000000)
288. feature 13z (0.000000)
289. feature 45x (0.000000)
290. feature 14z (0.000000)
291. feature 76z (0.000000)
292. feature 72z (0.000000)
293. feature 18y (0.000000)
294. feature 16x (0.000000)
295. feature 71z (0.000000)
296. feature 17x (0.000000)
297. feature 17y (0.000000)
```

```
298. feature 46x (0.000000)
299. feature 46z (0.000000)
300. feature 68z (0.000000)
```

```
In [8]: from sklearn.model_selection import cross_val_score
        model2 = RandomForestClassifier(n_estimators=50)
        cvdata = data.sample(frac=1)
        x_data = cvdata.drop(cvdata.columns[-1], axis=1)
        y_data = cvdata[cvdata.columns[-1]]
        scores = cross_val_score(model2, x_data, y_data, cv=10)
        print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.92 (+/- 0.04)
```

1 SVM

```
In [6]: from sklearn.svm import SVC
        svcModel = SVC(kernel='linear')
        svcModel.fit(X, y)
        print_stats(svcModel)
```

```
Statistics on Testing
True Negative: 143 False Positives: 14 False Negatives: 25 True Positives: 84
Accuracy: 0.8533834586466166 Error: 0.14661654135338342
Precision: 0.8571428571428571 Recall: 0.7706422018348624 F1: 0.8115942028985508
Statistics on Training
True Negative: 456 False Positives: 35 False Negatives: 45 True Positives: 260
Accuracy: 0.8994974874371859 Error: 0.10050251256281406
Precision: 0.8813559322033898 Recall: 0.8524590163934426 F1: 0.8666666666666666
```

2 Regular Decision Tree

```
In [9]: from sklearn.tree import DecisionTreeClassifier
        model = DecisionTreeClassifier()
        model.fit(X,y)
        print_stats(model)
```

```
Statistics on Testing
True Negative: 150 False Positives: 19 False Negatives: 17 True Positives: 80
Accuracy: 0.8646616541353384 Error: 0.13533834586466165
Precision: 0.8080808080808081 Recall: 0.8247422680412371 F1: 0.8163265306122448
Statistics on Training
True Negative: 479 False Positives: 0 False Negatives: 0 True Positives: 317
Accuracy: 1.0 Error: 0.0
Precision: 1.0 Recall: 1.0 F1: 1.0
```

```
In [10]: import warnings;warnings.filterwarnings('ignore')
         def train(file):
             data = pd.read_csv(file)
             data = data.drop(data.columns[0], axis=1)
             train, test = train_test_split(data, test_size=0.2)
             X = train.drop(train.columns[-1], axis=1)
             y = train[train.columns[-1]]
             y_true = test[test.columns[-1]]
             X_test = test.drop(train.columns[-1], axis=1)
             model = RandomForestClassifier()
             cvdata = data.sample(frac=1)
             scores = cross_val_score(model, x_data, y_data, cv=10)
             model.fit(X,y)
             return (model, model.score(X_test, y_true), scores)
         files = ["affirmative", "conditional",
          "doubt_question", "emphasis", "negative","relative",
         "topics", "wh_question", "yn_question"]
         models = []
         for file in files:
             m, score, cvscores = train("a_" + file + ".csv")
             models.append(m)
             print(score)
             print("Accuracy: %0.2f (+/- %0.2f)" % (cvscores.mean(), cvscores.std() * 2))
```

```
0.8873239436619719
Accuracy: 0.91 (+/- 0.05)
0.9554973821989529
Accuracy: 0.90 (+/- 0.06)
0.9125475285171103
Accuracy: 0.91 (+/- 0.06)
0.9644128113879004
Accuracy: 0.90 (+/- 0.06)
0.9288888888888889
Accuracy: 0.91 (+/- 0.05)
0.9742489270386266
Accuracy: 0.91 (+/- 0.04)
0.9694444444444444
Accuracy: 0.90 (+/- 0.05)
0.9573643410852714
Accuracy: 0.91 (+/- 0.04)
0.960431654676259
Accuracy: 0.90 (+/- 0.04)
```

3 Binary Classifier built on Person A tested on Person B

```
In [11]: for i in range(len(files)):
             file = files[i]
```

```
            model = models[i]
            forFun = pd.read_csv("b_" + file + ".csv")
            forFun = forFun.drop(forFun.columns[0], axis=1)
            X = forFun.drop(forFun.columns[-1], axis=1)
            y_true = forFun[forFun.columns[-1]]
            print(model.score(X, y_true))
```

```
0.49162011173184356
0.3136676499508358
0.6820307281229125
0.3950892857142857
0.3419721871049305
0.29044117647058826
0.3293150684931507
0.5865963855421686
0.4108170310701956
```

```
In [13]: data = pd.read_csv("a_affirmative.csv")
         data = data.drop(data.columns[0], axis=1)
         y = data[data.columns[-1]].iloc[1:]
         funf = data.drop(data.columns[-1], axis=1).diff().iloc[1:]
         funf = pd.concat([funf, y], axis=1)
         train, test = train_test_split(funf, test_size=0.25)
         X = train.drop(train.columns[-1], axis=1)
         y = train[train.columns[-1]]
         y_true = test[test.columns[-1]]
         X_test = test.drop(train.columns[-1], axis=1)
```

```
In [14]: model = RandomForestClassifier(n_estimators=100)
         model.fit(X,y)
         print_stats(model)
```

```
Statistics on Testing
True Negative: 141 False Positives: 24 False Negatives: 26 True Positives: 75
Accuracy: 0.8120300751879699 Error: 0.18796992481203012
Precision: 0.7575757575757576 Recall: 0.7425742574257426 F1: 0.75
Statistics on Training
True Negative: 482 False Positives: 0 False Negatives: 0 True Positives: 313
Accuracy: 1.0 Error: 0.0
Precision: 1.0 Recall: 1.0 F1: 1.0
```

```
In [15]: datab = pd.read_csv("b_affirmative.csv")
         datab = datab.drop(datab.columns[0], axis=1)
         y = datab[datab.columns[-1]].iloc[1:]
         X = datab.drop(datab.columns[-1], axis=1).diff().iloc[1:]
         print(model.score(X, y))
         y_pred = model.predict(X)
```

```
        cm = confusion_matrix(y, y_pred)
        tn, fp, fn, tp = cm.ravel()
        print("True Negative:", tn, "False Positives:", fp, "False Negatives:", fn, "True Pos
        precision = tp / (tp + fp)
        recall = tp / (tp + fn)
        f1 = 2 * (precision * recall) / (precision + recall)
        print("Precision:", precision, "Recall:", recall, "F1:", f1)
        print("Statistics on Training")
        y_pred = model.predict(X)
```

```
0.706430568499534
True Negative: 479 False Positives: 66 False Negatives: 249 True Positives: 279
Precision: 0.808695652173913 Recall: 0.5284090909090909 F1: 0.6391752577319587
Statistics on Training
```

4 Neural Nets for Binary

```
In [16]: train, test = train_test_split(data, test_size=0.15)
         X = train.drop(train.columns[-1], axis=1)
         y = train[train.columns[-1]]
         y_true = test[test.columns[-1]]
         X_test = test.drop(train.columns[-1], axis=1)
```

```
In [17]: import keras
         model = keras.Sequential([
             keras.layers.Dense(300, activation="sigmoid"),
             keras.layers.Dense(200, activation="relu"),
             keras.layers.Dense(1, activation="sigmoid")
         ])
         model.compile(loss="binary_crossentropy",
                       optimizer="sgd", metrics=["accuracy"])
         model.fit(X.values, y.values, epochs=8)
```

```
Using TensorFlow backend.


Epoch 1/8
902/902 [==============================] - 1s 616us/step - loss: 0.6670 - acc: 0.6142
Epoch 2/8
902/902 [==============================] - 0s 135us/step - loss: 0.6654 - acc: 0.6142
Epoch 3/8
902/902 [==============================] - 0s 141us/step - loss: 0.6734 - acc: 0.5987
Epoch 4/8
902/902 [==============================] - 0s 145us/step - loss: 0.6675 - acc: 0.6131
Epoch 5/8
902/902 [==============================] - 0s 132us/step - loss: 0.6704 - acc: 0.6131
Epoch 6/8
```

```
902/902 [==============================] - 0s 153us/step - loss: 0.6703 - acc: 0.6153
Epoch 7/8
902/902 [==============================] - 0s 142us/step - loss: 0.6680 - acc: 0.5998
Epoch 8/8
902/902 [==============================] - 0s 148us/step - loss: 0.6692 - acc: 0.6109
```

```
Out[17]: <keras.callbacks.History at 0x119653e10>
```

5 Combining Data for Multiclass

```
In [31]: l= []
         def efg(file):
             data = pd.read_csv("a_" + file + ".csv")
             data = data.drop(data.columns[0], axis=1)
             t = data[data[data.columns[-1]] == 1]
             t[t.columns[-1]] = file
             return t
         for file in files:
             count = efg(file)
             l.append(count)
```

```
In [39]: data = pd.concat(l)
         train, test = train_test_split(data, test_size=0.2)
         X = train.drop(train.columns[-1], axis=1)
         y = train[train.columns[-1]]
         y_true = test[test.columns[-1]]
         X_test = test.drop(train.columns[-1], axis=1)
         model = RandomForestClassifier()
         scores = cross_val_score(model, x_data, y_data, cv=10)
         print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
         model.fit(X,y)
         print(model.score(X_test, y_true))
         print(model.score(X,y))
```

```
Accuracy: 0.90 (+/- 0.06)
0.9730941704035875
0.9980359147025814
```

```
In [40]: y_pred = model.predict(X_test)
         from sklearn.metrics import recall_score
         from sklearn.metrics import precision_score
         print(recall_score(y_true, y_pred, average="weighted"))
         print(precision_score(y_true, y_pred, average="weighted"))
```

```
0.9730941704035875
0.9734927669050176
```

```
In [43]: importances = model.feature_importances_
         indices = np.argsort(importances)[::-1]
         col = list(X)
         for f in range(X.shape[1]):
             print("%d. feature %s (%f)" % (f + 1, col[indices[f]], importances[indices[f]]))
```

```
1. feature 45y (0.138088)
2. feature 76y (0.118896)
3. feature 68x (0.108831)
4. feature 44z (0.090481)
5. feature 77z (0.078817)
6. feature 38x (0.074893)
7. feature 41y (0.045530)
8. feature 80z (0.045471)
9. feature 99y (0.025123)
10. feature 92x (0.022713)
11. feature 82x (0.021149)
12. feature 98x (0.020614)
13. feature 81x (0.017445)
14. feature 93x (0.014384)
15. feature 76z (0.009687)
16. feature 68y (0.009622)
17. feature 83x (0.009506)
18. feature 61z (0.008281)
19. feature 18x (0.008225)
20. feature 48z (0.006732)
21. feature 74z (0.004608)
22. feature 99x (0.004502)
23. feature 70z (0.004102)
24. feature 58y (0.003753)
25. feature 73z (0.003689)
26. feature 50x (0.003659)
27. feature 72y (0.003164)
28. feature 60z (0.003148)
29. feature 57z (0.003137)
30. feature 60x (0.003112)
31. feature 39x (0.003031)
32. feature 85y (0.003000)
33. feature 86y (0.002991)
34. feature 64x (0.002981)
35. feature 97x (0.002933)
36. feature 45z (0.002885)
37. feature 95x (0.002770)
38. feature 43z (0.002468)
39. feature 52z (0.002465)
40. feature 49z (0.002303)
41. feature 40y (0.002286)
42. feature 33z (0.002207)
```

```
43. feature 47y (0.002196)
44. feature 31z (0.002158)
45. feature 38y (0.002124)
46. feature 30y (0.002043)
47. feature 64z (0.001739)
48. feature 84x (0.001721)
49. feature 20z (0.001661)
50. feature 0z (0.001515)
51. feature 53x (0.001496)
52. feature 89y (0.001385)
53. feature 18y (0.001349)
54. feature 41z (0.001339)
55. feature 70y (0.001333)
56. feature 46z (0.001264)
57. feature 62z (0.001257)
58. feature 21z (0.001234)
59. feature 59y (0.001212)
60. feature 39z (0.001206)
61. feature 72z (0.001205)
62. feature 14y (0.001185)
63. feature 12z (0.001111)
64. feature 78z (0.001111)
65. feature 56y (0.001088)
66. feature 33y (0.001088)
67. feature 21y (0.001075)
68. feature 47x (0.001058)
69. feature 32x (0.001007)
70. feature 55z (0.001005)
71. feature 94x (0.000952)
72. feature 6y (0.000888)
73. feature 8z (0.000832)
74. feature 68z (0.000714)
75. feature 9y (0.000665)
76. feature 50z (0.000651)
77. feature 63x (0.000635)
78. feature 92y (0.000632)
79. feature 28x (0.000619)
80. feature 98z (0.000614)
81. feature 56z (0.000605)
82. feature 86x (0.000600)
83. feature 92z (0.000598)
84. feature 63y (0.000593)
85. feature 53z (0.000590)
86. feature 56x (0.000564)
87. feature 11x (0.000564)
88. feature 48x (0.000556)
89. feature 62x (0.000544)
90. feature 5x (0.000508)
```

```
91. feature 31y (0.000423)
92. feature 93z (0.000423)
93. feature 70x (0.000423)
94. feature 58z (0.000423)
95. feature 94z (0.000423)
96. feature 83z (0.000344)
97. feature 93y (0.000317)
98. feature 96x (0.000317)
99. feature 81y (0.000298)
100. feature 80x (0.000268)
101. feature 80y (0.000209)
102. feature 50y (0.000164)
103. feature 99z (0.000136)
104. feature 81z (0.000058)
105. feature 65y (0.000000)
106. feature 20y (0.000000)
107. feature 23y (0.000000)
108. feature 23x (0.000000)
109. feature 22z (0.000000)
110. feature 22y (0.000000)
111. feature 22x (0.000000)
112. feature 90y (0.000000)
113. feature 90z (0.000000)
114. feature 21x (0.000000)
115. feature 91x (0.000000)
116. feature 20x (0.000000)
117. feature 24x (0.000000)
118. feature 19z (0.000000)
119. feature 19y (0.000000)
120. feature 19x (0.000000)
121. feature 18z (0.000000)
122. feature 91y (0.000000)
123. feature 91z (0.000000)
124. feature 17z (0.000000)
125. feature 17y (0.000000)
126. feature 17x (0.000000)
127. feature 23z (0.000000)
128. feature 24y (0.000000)
129. feature 16y (0.000000)
130. feature 24z (0.000000)
131. feature 88z (0.000000)
132. feature 89x (0.000000)
133. feature 31x (0.000000)
134. feature 30z (0.000000)
135. feature 89z (0.000000)
136. feature 30x (0.000000)
137. feature 29z (0.000000)
138. feature 29y (0.000000)
```

```
139. feature 29x (0.000000)
140. feature 28z (0.000000)
141. feature 28y (0.000000)
142. feature 90x (0.000000)
143. feature 27z (0.000000)
144. feature 27y (0.000000)
145. feature 27x (0.000000)
146. feature 26z (0.000000)
147. feature 26y (0.000000)
148. feature 26x (0.000000)
149. feature 25z (0.000000)
150. feature 25y (0.000000)
151. feature 25x (0.000000)
152. feature 16z (0.000000)
153. feature 16x (0.000000)
154. feature 32y (0.000000)
155. feature 7z (0.000000)
156. feature 7x (0.000000)
157. feature 6z (0.000000)
158. feature 97y (0.000000)
159. feature 6x (0.000000)
160. feature 5z (0.000000)
161. feature 5y (0.000000)
162. feature 97z (0.000000)
163. feature 4z (0.000000)
164. feature 4y (0.000000)
165. feature 4x (0.000000)
166. feature 3z (0.000000)
167. feature 3y (0.000000)
168. feature 3x (0.000000)
169. feature 2z (0.000000)
170. feature 2y (0.000000)
171. feature 2x (0.000000)
172. feature 1z (0.000000)
173. feature 1y (0.000000)
174. feature 1x (0.000000)
175. feature 98y (0.000000)
176. feature 0y (0.000000)
177. feature 7y (0.000000)
178. feature 8x (0.000000)
179. feature 15z (0.000000)
180. feature 8y (0.000000)
181. feature 15y (0.000000)
182. feature 15x (0.000000)
183. feature 14z (0.000000)
184. feature 94y (0.000000)
185. feature 14x (0.000000)
186. feature 13z (0.000000)
```

```
187. feature 13y (0.000000)
188. feature 13x (0.000000)
189. feature 95y (0.000000)
190. feature 12y (0.000000)
191. feature 12x (0.000000)
192. feature 11z (0.000000)
193. feature 11y (0.000000)
194. feature 95z (0.000000)
195. feature 10z (0.000000)
196. feature 10y (0.000000)
197. feature 10x (0.000000)
198. feature 9z (0.000000)
199. feature 96y (0.000000)
200. feature 9x (0.000000)
201. feature 96z (0.000000)
202. feature 88y (0.000000)
203. feature 32z (0.000000)
204. feature 65x (0.000000)
205. feature 49y (0.000000)
206. feature 72x (0.000000)
207. feature 73x (0.000000)
208. feature 73y (0.000000)
209. feature 74x (0.000000)
210. feature 55y (0.000000)
211. feature 55x (0.000000)
212. feature 54z (0.000000)
213. feature 54y (0.000000)
214. feature 54x (0.000000)
215. feature 74y (0.000000)
216. feature 53y (0.000000)
217. feature 75x (0.000000)
218. feature 75y (0.000000)
219. feature 52y (0.000000)
220. feature 52x (0.000000)
221. feature 51z (0.000000)
222. feature 51y (0.000000)
223. feature 51x (0.000000)
224. feature 75z (0.000000)
225. feature 76x (0.000000)
226. feature 77x (0.000000)
227. feature 57x (0.000000)
228. feature 57y (0.000000)
229. feature 71z (0.000000)
230. feature 67z (0.000000)
231. feature 65z (0.000000)
232. feature 64y (0.000000)
233. feature 66x (0.000000)
234. feature 63z (0.000000)
```

```
235. feature 66y (0.000000)
236. feature 66z (0.000000)
237. feature 67x (0.000000)
238. feature 62y (0.000000)
239. feature 67y (0.000000)
240. feature 61y (0.000000)
241. feature 58x (0.000000)
242. feature 61x (0.000000)
243. feature 69x (0.000000)
244. feature 60y (0.000000)
245. feature 69y (0.000000)
246. feature 59z (0.000000)
247. feature 69z (0.000000)
248. feature 59x (0.000000)
249. feature 71x (0.000000)
250. feature 71y (0.000000)
251. feature 77y (0.000000)
252. feature 49x (0.000000)
253. feature 33x (0.000000)
254. feature 78x (0.000000)
255. feature 40x (0.000000)
256. feature 85z (0.000000)
257. feature 39y (0.000000)
258. feature 86z (0.000000)
259. feature 38z (0.000000)
260. feature 87x (0.000000)
261. feature 87y (0.000000)
262. feature 37z (0.000000)
263. feature 37y (0.000000)
264. feature 37x (0.000000)
265. feature 36z (0.000000)
266. feature 36y (0.000000)
267. feature 36x (0.000000)
268. feature 35z (0.000000)
269. feature 35y (0.000000)
270. feature 35x (0.000000)
271. feature 34z (0.000000)
272. feature 34y (0.000000)
273. feature 34x (0.000000)
274. feature 87z (0.000000)
275. feature 88x (0.000000)
276. feature 85x (0.000000)
277. feature 40z (0.000000)
278. feature 41x (0.000000)
279. feature 45x (0.000000)
280. feature 48y (0.000000)
281. feature 78y (0.000000)
282. feature 47z (0.000000)
```

```
283. feature 79x (0.000000)
284. feature 79y (0.000000)
285. feature 79z (0.000000)
286. feature 46y (0.000000)
287. feature 46x (0.000000)
288. feature 82y (0.000000)
289. feature 82z (0.000000)
290. feature 84z (0.000000)
291. feature 44y (0.000000)
292. feature 44x (0.000000)
293. feature 83y (0.000000)
294. feature 43y (0.000000)
295. feature 43x (0.000000)
296. feature 42z (0.000000)
297. feature 42y (0.000000)
298. feature 42x (0.000000)
299. feature 84y (0.000000)
300. feature 0x (0.000000)
```

```
In [41]: model = DecisionTreeClassifier()
         scores = cross_val_score(model2, x_data, y_data, cv=10)
         print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
         model.fit(X,y)
         print(model.score(X_test, y_true))
```

```
Accuracy: 0.92 (+/- 0.06)
0.9349775784753364
```

```
In [42]: y_pred = model.predict(X_test)
         from sklearn.metrics import recall_score
         from sklearn.metrics import precision_score
         print(recall_score(y_true, y_pred, average="weighted"))
         print(precision_score(y_true, y_pred, average="weighted"))
```

```
0.9349775784753364
0.9381452528537729
```

```
In [18]: def convert(y):
             files = ["affirmative", "conditional",
          "doubt_question", "emphasis", "negative","relative",
         "topics", "wh_question", "yn_question"]
             mapped = {}
             for i in range(len(files)):
                 mapped[files[i]] = i
             value = [0 for i in range(len(files))]
             ans = []
```

```
            for i in y:
                hot = value[:]
                hot[mapped[i]] = 1
                ans.append(hot)
            return np.asarray(ans)
        dummy_y = convert(y)
```

```
In [19]: import keras
         model = keras.Sequential([
             keras.layers.Dense(300, activation="sigmoid"),
             keras.layers.Dense(200, activation="tanh"),
             keras.layers.Dense(100, activation="relu"),
             keras.layers.Dense(9, activation="softmax")
         ])
         model.compile(loss="categorical_crossentropy",
                       optimizer="sgd", metrics=["accuracy"])
         model.fit(X.values, np.asarray(dummy_y), epochs=5)
```

```
Epoch 1/5
3564/3564 [==============================] - 1s 235us/step - loss: 2.1916 - acc: 0.1330
Epoch 2/5
3564/3564 [==============================] - 0s 123us/step - loss: 2.1887 - acc: 0.1299
Epoch 3/5
3564/3564 [==============================] - 1s 160us/step - loss: 2.1842 - acc: 0.1375
Epoch 4/5
3564/3564 [==============================] - 1s 179us/step - loss: 2.1816 - acc: 0.1386
Epoch 5/5
3564/3564 [==============================] - 1s 159us/step - loss: 2.1834 - acc: 0.1254
```

```
Out[19]: <keras.callbacks.History at 0x1193ed198>
```

6 Testing model against other person

```
In [20]: def efg(person, file):
             data = pd.read_csv(person + "_" + file + ".csv")
             data = data.drop(data.columns[0], axis=1)
             y = data[data.columns[-1]].iloc[1:]
             funf = data.drop(data.columns[-1], axis=1).diff().iloc[1:]
             funf = pd.concat([funf, y], axis=1)
             t = funf[funf[funf.columns[-1]] == 1]
             t[t.columns[-1]] = file
             return t
```

```
In [21]: l= []
         for file in files:
             count = efg("a", file)
             l.append(count)
         data_a = pd.concat(l)
         l= []
         for file in files:
             count = efg("b", file)
             l.append(count)
         data_b = pd.concat(l)

In [22]: train, test = train_test_split(data_a, test_size=0.25)
         X = train.drop(train.columns[-1], axis=1)
         y = train[train.columns[-1]]
         y_true = test[test.columns[-1]]
         X_test = test.drop(train.columns[-1], axis=1)

In [23]: model = RandomForestClassifier(n_estimators=100)
         model.fit(X,y)
         print(model.score(X_test, y_true))

0.533213644524237


In [24]: y_b = data_b[data_b.columns[-1]].iloc[1:]
         X_b = data_b.drop(data_b.columns[-1], axis=1).diff().iloc[1:]
         model.score(X_b, y_b)

Out[24]: 0.16955719557195573

In [ ]:
```