

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.svm import SVC
4 import matplotlib.pyplot as plt
5 from sklearn.preprocessing import StandardScaler
6
7
8 #Importing the Data from the give files
9 train_set = pd.read_csv("tweets-train-data.csv", sep= ",", header = None, names =
10 ["Tweet", "Date", "Retweets", "Likes", "Location"])
11 test_set = pd.read_csv("tweets-test-data.csv", sep= ",", header = None, names =
12 ["Tweet", "Date", "Retweets", "Likes", "Location"])
13 train_targets = pd.read_csv("tweets-train-targets.csv", sep= ",", header = None,
14 names = ["Author"])
15 test_targets = pd.read_csv("tweets-test-targets.csv", sep= ",", header = None, names
16 = ["Author"])
17
18 #Formating the data
19 training = pd.concat([train_set, train_targets], axis = 1)
20 training = training[training["Date"].astype(str).str.startswith("2016")]
21 training["Likes"] = training["Likes"].astype(float)
22 training["Date"], training["Time"] = training["Date"].str.split("T").str
23 training["Date"].astype("datetime64")
24 print(training)
25
26 #Likes information
27 Trump = training[training["Author"] == "DT"]
28 Clinton = training[training["Author"] == "HC"]
29 Clinton_likes = Clinton["Likes"]
30 Trump_likes = Trump["Likes"]
31 print(Trump_likes)
32 print(Clinton_likes)
33
34 #Ploting the retweets
35 Clinton_rt = Clinton["Retweets"]
36 Trump_rt = Trump["Retweets"]
37
38 plt.plot(Clinton_rt, "c")
39 plt.plot(Trump_rt, "r")
40 plt.xlabel("Tweet id")
41 plt.ylabel("Number of rts")
42 plt.legend(("HC", "DT"))
43 plt.grid(True)
44
45 plt.show()
46 plt.close()
47
48 #Clinton most popular tweet (rts)
49 print(max(Clinton_rt))
50 top_Clinton_rt = training.loc[training["Retweets"]==max(Clinton_rt)]
51 print(top_Clinton_rt)
52
53 #Trump most popular tweet (rts)
54 print(max(Trump_rt))
55 top_Trump_rt = training.loc[training["Retweets"]==max(Trump_rt)]
56 print(top_Trump_rt)
57
58 #Clinton most liked tweet
59 print(max(Clinton_likes))
60 top_Clinton_likes = training.loc[training["Likes"]==max(Clinton_likes)]
61 print(top_Clinton_likes)
62
63 #Trump most liked tweet
64 print(max(Trump_likes))
65 top_Trump_likes = training.loc[training["Likes"]==max(Trump_likes)]
66 print(top_Trump_likes)
67
68 #Formating the testing data
69 testing = pd.concat([test_set, test_targets], axis = 1)
70 testing = testing[testing["Date"].astype(str).str.startswith("2016")]
71 testing["Likes"] = testing["Likes"].astype(float)
72 testing["Date"], testing["Time"] = testing["Date"].str.split("T").str

```

```

69 testing["Date"].astype("datetime64")
70 print(testing)
71
72 #Extracting the features
73 train_features = training["Author"]
74 train_rts = training["Retweets"].values.reshape(-1,1)
75 test_features = testing["Author"]
76 test_rts = testing["Retweets"].values.reshape(-1,1)
77
78
79 #Normalize the tweets length
80 a = np.mean(training["Tweet"].apply(len))
81 b = np.mean(testing["Tweet"].apply(len))
82
83 training["word_length"] = training["Tweet"].apply(len)-a
84 testing["word_length"] = testing["Tweet"].apply(len)-b
85
86 #Re-assign
87 Trump = training[training["Author"] == "DT"]
88 Clinton = training[training["Author"] == "HC"]
89
90 Trump_average_length = Trump["word_length"]
91 Clinton_average_length = Clinton["word_length"]
92
93 #Spaces count
94 training["space_count"] = training["Tweet"].str.count(" ")
95 training["a_count"] = training["Tweet"].str.count("a")
96 testing["space_count"] = testing["Tweet"].str.count(" ")
97 testing["a_count"] = testing["Tweet"].str.count("a")
98
99 #Taking logarithms (monotonic function)
100 training["log_likes"] = np.log(training["Likes"])
101 training["log_rts"] = np.log(training["Retweets"])
102 testing["log_likes"] = np.log(testing["Likes"])
103 testing["log_rts"] = np.log(testing["Retweets"])
104
105 #Dot count
106 training["dot_count"] = training["Tweet"].str.count(".")
107 testing["dot_count"] = testing["Tweet"].str.count(".")
108
109 #Hours
110 training["hour"] = training["Time"].str[0:2]
111 pd.to_numeric(training["hour"])
112 testing["hour"] = testing["Time"].str[0:2]
113 pd.to_numeric(testing["hour"])
114
115 #Months
116 training["month"] = training["Date"].str[5:7]
117 pd.to_numeric(training["month"])
118 testing["month"] = testing["Date"].str[5:7]
119 pd.to_numeric(testing["month"])
120
121 scale = StandardScaler()
122
123 log_training_matrix = training[["log_likes", "log_rts", "word_length", "a_count",
124 "space_count", "dot_count", "month", "hour", "Retweets"]]
125 log_testing_matrix = testing[["log_likes", "log_rts", "word_length", "a_count",
126 "space_count", "dot_count", "month", "hour", "Retweets"]]
127
128 transformed_x = scale.fit_transform(log_training_matrix, train_features)
129 transformed_y = scale.fit_transform(log_testing_matrix, test_features)
130
131 #Training using SVC
132
133 clf = SVC(C=10, kernel = "rbf", gamma = 0.02)
134 clf.fit(transformed_x, train_features)
135
136 y_predict = clf.predict(transformed_y)
137
138 from sklearn import metrics
139
140 report = metrics.classification_report(test_features, y_predict)

```

```

139 metrics.accuracy_score(test_features, y_predict)
140
141 #K-fold cross Validation
142 try:
143     from sklearn.model_selection import KFold, cross_val_score
144     legacy = False
145 except ImportError:
146     from sklearn.model_selection import KFold, cross_val_score
147     legacy = True
148
149 #K=3
150 if legacy:
151     kf = KFold(len(train_features), n_folds = 3, shuffle = True, random_state = 42)
152 else:
153     kf = KFold(n_splits = 3, shuffle = True, random_state = 42)
154
155 gamma_values = [0.01, 0.001, 0.0001, 0.00001]
156 accuracy_scores = []
157
158 #K-Fold cross validation algorithm:
159 #-Train predictor
160 #-Compute score
161 for gamma in gamma_values:
162
163     #Train classifier
164     clf = SVC(C=10, kernel = "rbf", gamma = gamma)
165
166     #Score
167     if legacy:
168         scores = cross_val_score(clf, transformed_x, train_features, cv = kf,
169                                 scoring="accuracy")
170     else:
171         scores = cross_val_score(clf, transformed_x, train_features, cv =
172                                 kf.split(transformed_x), scoring = "accuracy")
173
174     #Compute Average score
175     accuracy_score = scores.mean()
176     accuracy_scores.append(accuracy_score)
177
178 #Best value of gamma
179 best_index = np.array(accuracy_scores).argmax()
180 best_gamma = gamma_values[best_index]
181
182 #Train with the selected gamma
183 clf = SVC(C=10, kernel = "rbf", gamma = best_gamma)
184 clf.fit(transformed_x, train_features)
185
186 #Evaluate on the test set
187 y_predict = clf.predict(transformed_y)
188 accuracy = metrics.accuracy_score(test_features, y_predict)
189
190 print(accuracy)
191
192 print(best_gamma)
193
194 from sklearn.model_selection import learning_curve
195
196 plt.figure(2)
197 plt.title("Learning curve")
198 plt.xlabel("Training examples")
199 plt.ylabel("Score")
200 plt.grid(True)
201
202 clf = SVC(C=10, kernel = "rbf", gamma = best_gamma)
203
204 train_sizes, train_scores, val_scores = learning_curve(clf, transformed_x,
205                                                         train_features, scoring = "accuracy")
206
207 train_scores_mean = np.mean(train_scores, axis = 1)
208 train_scores_std = np.std(train_scores, axis = 1)
209 val_scores_mean = np.mean(val_scores, axis = 1)

```

```

208 val_scores_std = np.std(val_scores, axis = 1)
209
210 #Training scores: mean
211 plt.plot(train_sizes, train_scores_mean, '*', color = "r", label = "Training Score")
212 #Training scores: std
213 plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
214 train_scores_mean + train_scores_std, alpha = 0.1, color = "r")
215 #Validation scores: mean
216 plt.plot(train_sizes, val_scores_mean, '*', color = "c", label = "Cross Validation
217 Scores")
218 #Validation scores: std
219 plt.fill_between(train_sizes, val_scores_mean - val_scores_std, val_scores_mean +
220 val_scores_std, alpha = 0.1, color = "c")
221
222 plt.ylim(0.05, 1.3)
223 plt.legend()
224 plt.show()
225 plt.close()
226
227 try:
228     from sklearn.model_selection import GridSearchCV
229 except ImportError:
230     from sklearn.grid_search import GridSearchCV
231 possible_parameters = {
232     'C' : [10,4,3,2],
233     'gamma' : [0.01,0.001,0.0001,0.00001]
234 }
235
236 svc = SVC(kernel = "rbf")
237
238 #The GridSearch is a classifier. Hence, we try to fit it with the training data,
239 #and then use it for prediction
240 clf = GridSearchCV(svc, possible_parameters, n_jobs = 4, cv = 3)
241 clf.fit(transformed_x, train_features)
242
243 y_predidct = clf.predict(transformed_y)
244 accuracy = metrics.accuracy_score(test_features, y_predict)
245
246 #Training phase
247 clf = SVC(C=10, kernel = "rbf", gamma = 0.01)
248 clf.fit(transformed_x, train_features)
249 #Prediction phase
250 y_predict = clf.predict(transformed_y)
251
252 report = metrics.classification_report(test_features, y_predict)
253 metrics.accuracy_score(test_features, y_predict)
254
255 print(report)
256
257 #Performance parameters
258 from sklearn.model_selection import cross_val_score
259
260 accuracy = cross_val_score(clf, transformed_x, train_features, cv = 10, scoring =
261 "accuracy")
262
263 #Labels
264 labels = train_features.map(lambda x: 1 if x == "HC" else 0).values
265
266 precision = cross_val_score(clf, transformed_x, labels, cv = 10, scoring =
267 "precision")
268
269 recall = cross_val_score(clf, transformed_x, labels, cv = 10, scoring = "recall")
270
271 f1 = cross_val_score(clf, transformed_x, labels, cv=10, scoring = "f1")
272
273 print("Avg Precision: {}".format(round(precision.mean(), 3)))
274 print("Avg Recall: {}".format(round(recall.mean(), 3)))
275 print("Avg Accuracy: {}".format(round(accuracy.mean(), 3)))
276 print("Avg f1: {}".format(round(f1.mean(), 3)))

```

```
275 #Retweets plot
276 plt.hist(training["Retweets"][training["Author"]=="HC"], color = "c")
277 plt.hist(training["Retweets"][training["Author"]=="DT"], color = "r")
278 plt.title("Retweets")
279 plt.legend()
280 plt.grid(True)
281 plt.show()
282 plt.close()
283
284
285 #Log Retweets plot
286 plt.hist(training["log_rts"][training["Author"]=="HC"], color = "c")
287 plt.hist(training["log_rts"][training["Author"]=="DT"], color = "r")
288 plt.title("Log Retweets")
289 plt.legend()
290 plt.grid(True)
291 plt.show()
292 plt.close()
293
294 #Likes plot
295 plt.hist(training["Likes"][training["Author"]=="HC"], color = "c")
296 plt.hist(training["Likes"][training["Author"]=="DT"], color = "r")
297 plt.title("Likes")
298 plt.legend()
299 plt.grid(True)
300 plt.show()
301 plt.close()
302
303 #Log likes plot
304 plt.hist(training["log_likes"][training["Author"]=="HC"], color = "c")
305 plt.hist(training["log_likes"][training["Author"]=="DT"], color = "r")
306 plt.title("Log Likes")
307 plt.legend()
308 plt.grid(True)
309 plt.show()
310 plt.close()
311
```