Assignment 3: Unhealthy Conversations

Joshua W. Brook

November 17, 2021

1 What Makes an Comment Unhealthy?

In order to understand and identify what constitutes unhealthy online interactions, a team of researchers compiled The Unhealthy Comments Corpus (UCC). The UCC consists of 44355 comments which have been taken from online news and media platforms, and have been meticulously classified by humans. These are the labels which subjects were asked to assign to each comment in the corpus:

- antagonistic
- condescending
- dismissive
- generalisation
- generalisation_unfair
- hostile
- sarcastic
- healthy

These labels are encoded with values of either 1 or 0, representing booleans. A confidence value is also given for each label, in the range [0,1]. A machine learning algorithm can be implemented onto the data and used to predict whether the comments are *healthy* or *unhealthy* based on the values of the labels.

2 Predicting Unhealthy Comments

2.1 Exploring the Data

The original comments are taken from the SFU Opinion and Comments Corpus (SOCC), which is compiled of comments taken from opinion articles available online. The data is already split into three .csv files - train, test, and validation. Each row of data has a comment, taken at random from the main dataset, as well as all associated labels and their confidence values. The data is well structured and doesn't require any additional cleaning or formatting.

2.2 Classifying the Data

I tried implementing multiple different classifiers from the scikit-learn library to see which one's could best fit the data. In general, ensemble methods performed much better than standard classifiers. Following the **Scikit-Learn cheat-sheet**, I started by implementing a **KNeighbors** classifier. This was reached as we are trying to predict a category from labelled numerical data with less than 100K samples. Expanding on this, I implemented some ensemble methods to really try and stretch the accuracy of predictions. Implenting **RandomForest** and **GradientBoosting** classifiers increased the accuracy, but only slightly. I fiddled with some *for loops* to check the accuracy scores for different hyperparameters of each classifier. I finally settled on using a voting classifier based on these three classifiers (now with optimised hyperparameters) and utilising hard voting to get the best results that I could. Strangely enough, classifying just based on the *hostile* and *antagonize* labels returned the best accuracy score; proving that sometimes less is indeed more. Ideally, this data could be visualised with a nice scatter plot or some histograms, but unfortunately binary data is not very exciting to look at. In the next section, I will implement the optimal version of my code and analyse the results.

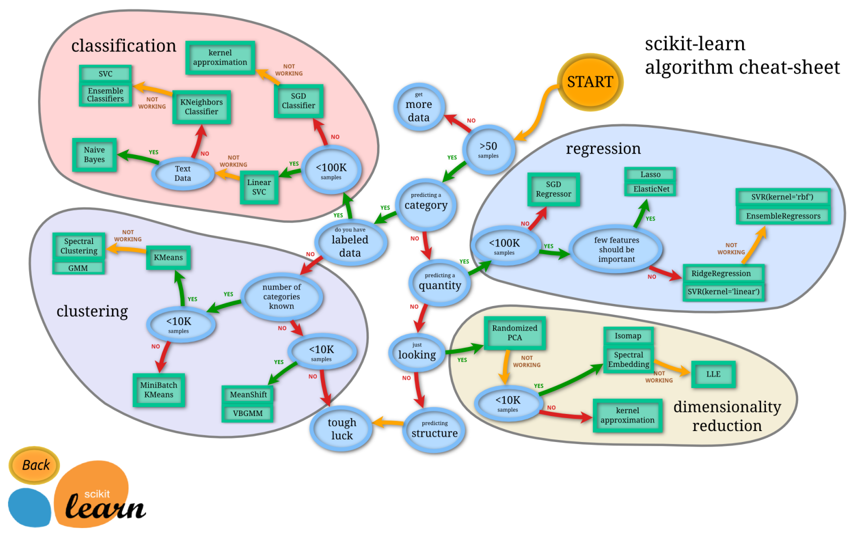

Figure 1: Sci-Kit Learn Cheat Sheet

3 Implementation of Machine Learning

3.1 Importing

Let's try running some code on the UCC data to try and predict whether a comment is healthy or not. I'll begin by doing some general imports from pandas, scikit-learn and the UCC datasets.

```
import pandas as pd

from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import ( GradientBoostingClassifier,
                               VotingClassifier,
                               RandomForestClassifier )

data = pd.read_csv("corpus/train.csv")
testdata = pd.read_csv("corpus/test.csv")
```

3.2 Classification

With these imports we can beign to classify some comments. Let's start by grouping together the labels for ease of use, as well as defining the train and test data sets. I'll also set up all the classifers that I imported by defining their hyperparameters.

```
all_attributes = ["antagonize" , "condescending", "dismissive",
              "generalisation", "generalisation_unfair",
              "hostile", "sarcastic"]

best_attributes = ["antagonize", "hostile"]

X = data[best_attributes]
y = data["healthy"]

Xtest = testdata[best_attributes]
ytest = testdata["healthy"]

knn = KNeighborsClassifier(n_neighbors=10)
rfc = RandomForestClassifier(n_estimators=200, random_state=42)
gbc = GradientBoostingClassifier(n_estimators=100,
                learning_rate=1.0, max_depth=1, random_state=0)

vote = VotingClassifier(estimators=[("gbc", gbc),
                                    ("rfc", rfc),
                                    ("knn", knn) ],
                        voting='hard')
```

3.3 Testing

With the classifiers set up, we can fit the best one to the training data and finally predict which values from the test set are healthy. I'll also add some code to print the results and an accuracy score based on the real values for *healthy* vs the predictions.

```
clf = vote

clf.fit(X, y)
ypred = clf.predict(Xtest)
acc = accuracy_score(ytest, ypred)

p = print("Accuracy Score:", acc,
          "\nytest: True Values:\n", ytest.value_counts(),
          "\n\nypred: Predictions:\n", pd.Series(ypred).value_counts())
```

3.4 Results

```
Accuracy Score: 0.944858757062147

ytest: True Values:
1 4105
0 320
Name: healthy, dtype: int64

ypred: Predictions:
1 4197
0 228
dtype: int64
```

3.5 Analysis

The values that have been assigned to the testing data by humans are shown above, under *True Values*. As we can see, there are 4105 comments assigned as *healthy* and another 320 as *unhealthy*. The classification algorithms that I implemented assign 4197 as *healthy* and 228 as *unhealthy*. This gives an accuracy of 94.49%, and if we run a bit more code on our predictions we can see which comments are incorrectly assigned. We get 255 comments which have been incorrectly assigned in one way or another.

Index	**True**	**Predicted**
3	1.0	0.0
10	0.0	1.0
19	1.0	0.0
35	1.0	0.0
60	0.0	1.0
...	...	...
4250	0.0	1.0
4269	0.0	1.0
4331	0.0	1.0
4336	0.0	1.0
4420	0.0	1.0

Table 1: Incorrectly Assigned Comments

4 Plots

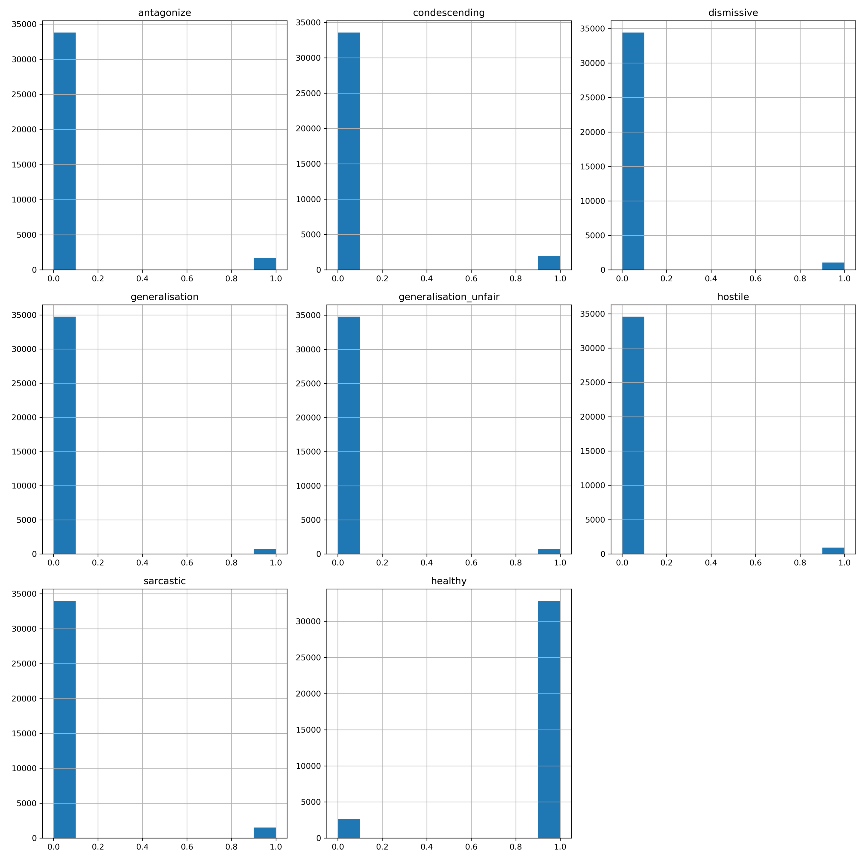

Figure 2: Histogram Plots of Label Values