CSC 2730

Group Project

Brandon Lara & Christian Facundus

12/6/2020

Initially, our goal for this project was to predict whether a fire inspection in Baton Rouge would yield a fire code violation based on a dataset. We used a dataset which had information on Baton Rouge fire inspections. The dataset included business name, information about the address, the city and state, and information about date and type of inspection, as well as if there was a violation. We removed many columns that were not relevant to the question, leaving business name, full address, zip code, date of inspection, and type of inspection. For our first attempt, we removed the violation column and replaced it with a column of ones and zeros, one representing there was a violation and zero representing there was not a violation. We then fed this dataset to a multinomial Bayes model, with the hope that the model would be able to predict whether there was a violation. The model was unable to meaningfully predict if a violation would happen because an overwhelming percentage of the dataset was “No Violation”. The model predicted that every outcome would be no violation. While this was 95% accurate, we still considered it a failure of the model to be meaningfully predictive. In order to address this fault, we decided it would be best to change our goal to instead focus on predicting the type of violation. We removed all of the non-violation rows and left the violation codes. Running the same models on this modified dataset produced inaccurate predictions, but the model was no longer overfitted.

For the multinomial Bayes model, we used alphas of .01, .1, .5, 1, and 10. An alpha value of .5 produced the most accurate predictions, while an alpha value of 10 produced the least accurate predictions.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, f1_score

for i in [.01,.1,.5,1,10]:
  model = MultinomialNB(alpha = i)
  model.fit(train, y_train)
  y_predicted = model.predict(test)
  print("Alpha: ", i)
  print(accuracy_score(y_test, y_predicted))
```

```
Alpha:  0.01
0.34316134316134317
Alpha:  0.1
0.346983346983347
Alpha:  0.5
0.34944034944034946
Alpha:  1
0.3333333333333333
Alpha:  10
0.2781872781872782
```

For the random forest classifier model, we used n_estimators of 1, 10, 20, 100, and 500. Using a n_estimator value of 500 produced the most accurate predictions, while a n_estimator value of 10 produced the least accurate predictions.

```
from sklearn.ensemble import RandomForestClassifier
for i in [1,10,20,100,500]:
  model = RandomForestClassifier(random_state=0, n_estimators=i)
  model.fit(train, y_train)
  y_predicted = model.predict(test)
  print("N_estimator: ", i)
  print(accuracy_score(y_test, y_predicted))
```

```
N_estimator:  1
0.2295932295932296
N_estimator:  10
0.27927927927927926
N_estimator:  20
0.2831012831012831
N_estimator:  100
0.28992628992628994
N_estimator:  500
0.2934752934752935
```

Multinomial Naive Bayes

Alpha Value	Accuracy (%)
.01	34.3
.1	34.6
.5	34.9
1	33.3
10	27.8

Random Forest Classifier

N_estimators Value	Accuracy (%)
1	22.9
10	27.9
20	28.3
100	28.9
500	29.3

We measured the performance of our models using accuracy_score on train_test_split data. The best performance we saw was from multinomial naive bayes with an accuracy of 34.9%. We believe that the data given in the dataset is not predictive of the type of violation, and this results in a poor performance of the model. If our group were to continue this project with the goal of producing a predictive model, we would attempt to collect new variables that might be more predictive, like the age of the building and the average income of the area.