Willy Esquivel-Lopez

(wesquive - G01127937)

(miner2 user-name): Gopher123 – Best Public Score: .67

HW2 Report

My Approach:

For This assignment I approached by looking at the data and seeing what I can do to decide the best model to use for classification. I saw a combination of categorical and continuous data, thus between SVMs and Decision trees, decision trees made more sense as it would be faster to compute, it would be less complex, and would give me more control over the classification.

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	credit
0	0	13	40	1	1	2174	0	4	7	9	White	Male	0
1	1	13	13	0	4	0	0	2	6	9	White	Male	0
2	2	9	40	1	6	0	0	0	4	11	White	Male	0
3	3	7	40	0	6	0	0	2	4	1	Black	Male	0
4	4	13	40	5	10	0	0	2	4	9	Black	Female	0
5	5	14	40	5	4	0	0	2	4	12	White	Female	0
6	6	5	16	1	8	0	0	3	4	6	Black	Female	0
7	7	9	45	0	4	0	0	2	6	11	White	Male	1
8	8	14	50	1	10	14084	0	4	4	12	White	Female	1
9	9	13	40	0	4	5178	0	2	4	9	White	Male	1
10	10	10	80	0	4	0	0	2	4	15	Black	Male	1
11	11	13	40	0	10	0	0	2	7	9	Asian-Pac-Islander	Male	1
12	12	13	30	3	1	0	0	4	4	9	White	Female	0
13	13	12	50	1	12	0	0	4	4	7	Black	Male	0
14	14	11	40	0	3	0	0	2	4	8	Asian-Pac-Islander	Male	1

Figure 1 - Train data unchanged

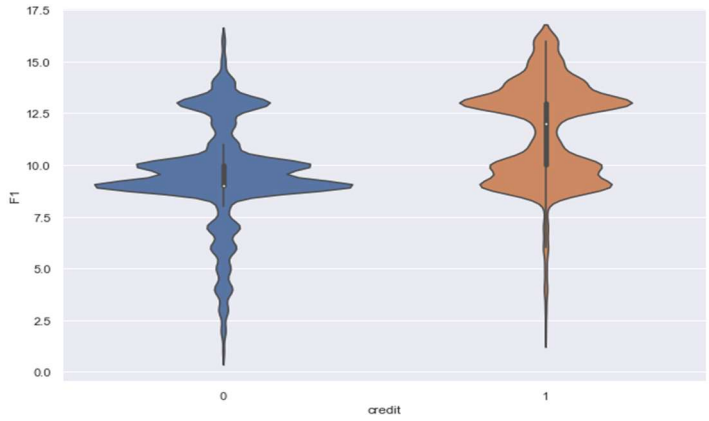
I had to modify two columns to continuous variable to simplify the tree

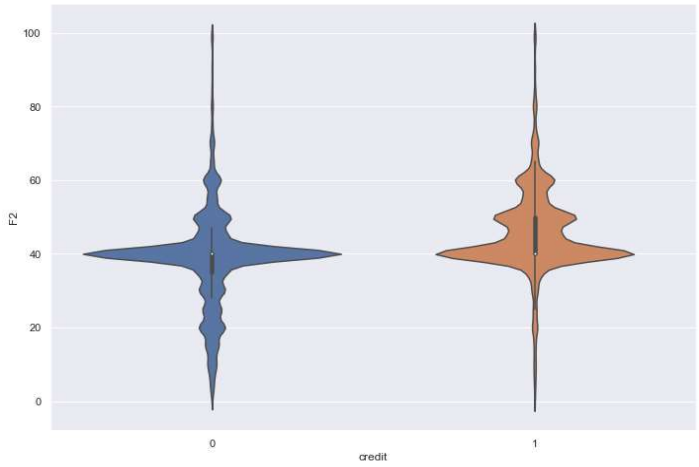
```
# after looking at some of the categorical values we will need to type convert them to work with continuis values
df['F10'],_ = pd.factorize(df['F10'])
df['F11'],_ = pd.factorize(df['F11'])
df.head(15)
```

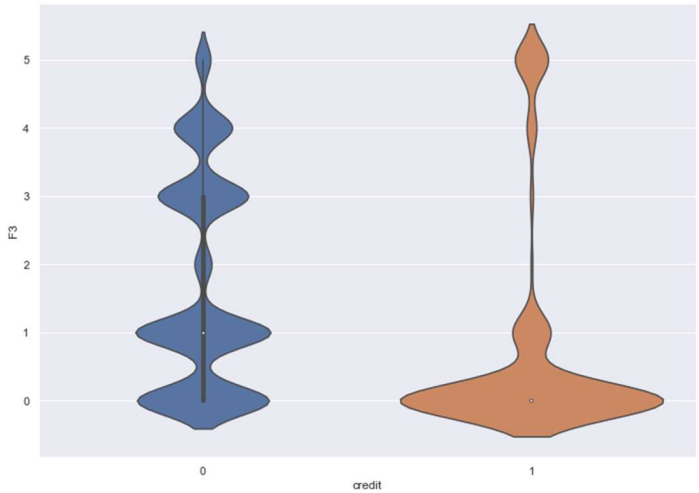
	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	credit
0	0	13	40	1	1	2174	0	4	7	9	0	0	0
1	1	13	13	0	4	0	0	2	6	9	0	0	0
2	2	9	40	1	6	0	0	0	4	11	0	0	0
3	3	7	40	0	6	0	0	2	4	1	1	0	0
4	4	13	40	5	10	0	0	2	4	9	1	1	0
5	5	14	40	5	4	0	0	2	4	12	0	1	0
6	6	5	16	1	8	0	0	3	4	6	1	1	0
7	7	9	45	0	4	0	0	2	6	11	0	0	1
8	8	14	50	1	10	14084	0	4	4	12	0	1	1
9	9	13	40	0	4	5178	0	2	4	9	0	0	1
10	10	10	80	0	4	0	0	2	4	15	1	0	1
11	11	13	40	0	10	0	0	2	7	9	2	0	1
12	12	13	30	3	1	0	0	4	4	9	0	1	0
13	13	12	50	1	12	0	0	4	4	7	1	0	0
14	14	11	40	0	3	0	0	2	4	8	2	0	1

Analysis:

To Gather a clearer and broader picture of the distribution in the data I plotted violin plots for each feature against my target value (credit). There appeared to be some imbalances, I will explain how I solved that issue.







etc. (The rest are on the notebook)

F3 had no label but appeared to be crucial to the outcome of the decision tree.

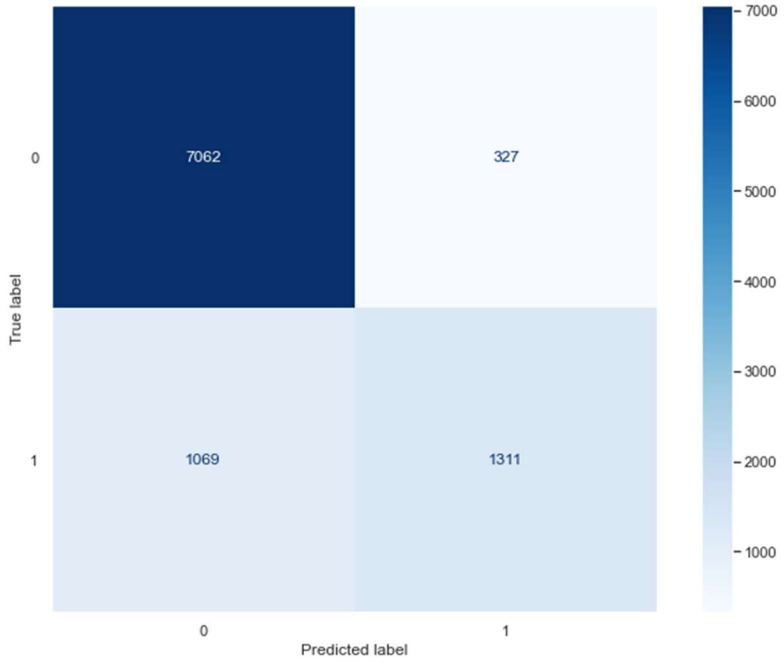
The Training and Testing:

I Played it safe and used a 70-30 split for my models. Labeled the features and my target variables like so:

```
# Here I will perform the test train split
feature_cols = ['F1','F2','F3','F4','F5','F6','F7','F8','F9','F10','F11']
X = df[feature_cols] # Features
y = df.credit # Target variable
# train 70% test 30% split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

I decided to test the Gini Index on using a decision tree with max depth of 3, 5, and 7. All of which appeared to do fine but would not break on 85% accuracy. But the F1 score suffers regarding false negatives upon investigating the confusion matrix generated:

	precision	recall	f1-score	support
0	0.87	0.96	0.91	7389
1	0.80	0.55	0.65	2380
accuracy			0.86	9769
macro avg	0.83	0.75	0.78	9769
weighted avg	0.85	0.86	0.85	9769



True Negatives (Predicted = 0, Actual = 0)	**False Positives** (Predicted = 1, Actual = 0)
False Negatives (Predicted = 0, Actual = 1)	**True Positives** (Predicted = 1, Actual = 1)

$$Precision(0) = \frac{TN}{TN + FN} \qquad Precision(1) = \frac{TP}{TP + FP}$$

$$Recall(0) = \frac{TN}{TN + FP} \qquad Recall(1) = \frac{TP}{TP + FN}$$

Bagging and Predictive Threshold:

I had to improve the decision tree model somehow, so I decided to use the Bagging technique to get a better performing model. I discovered with bagging the best performance results when using a decision tree of max_depth=9 to create subsets of random replacement.

After doing some research it appears the best way to solve the misclassification errors resulting in false negatives from my model prediction was to limit the threshold on which the predictions are made. The major reason this was happening is due to the unbalance of the target classification. So, I tuned the threshold to .31 to compensate for the imbalances, like so:

```
# first, i will adjust the threshhold
# I will keep the split at 70-30 as it has the best outcomes
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=17)
# Here I will be trining and fitting the decision tree model using entropy information gain node splitting and maxdepth 7
d_tree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=9, random_state=17)
d_tree.fit(X_train, y_train)
# I will initiate a bagging classifier in effort to improve the performance of my model
bc = BaggingClassifier(base_estimator=d_tree, n_estimators=300, oob_score=True, random_state=17)
# Fit the bagging classifier to the training set
bc.fit(X_train, y_train.ravel())

# Predict test set labels using a threshold of .31
y_pred = (bc.predict_proba(X_test)[:,1] >= 0.31).astype(bool)

# I will now check thoroughly the performance of my most accurate model predictor most importantly the F1 Score
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.87      0.89      7389
           1       0.65      0.76      0.70      2380

    accuracy                           0.84      9769
   macro avg       0.78      0.81      0.79      9769
weighted avg       0.85      0.84      0.84      9769
```

Resulting in a best score of:

Rank	User	Submission Time	Public Score
54	Gopher123	Oct. 14, 2020, 11:41 a.m.	0.67

This approach increased my score form 62% to 67% by using bagging and a threshold delimiter to handle data imbalances and boosting the performance of my decision tree.