Ryan Christopher CS677

Assignment 3
-Revenge of the Banknote-

Question 1:

Part 1 - assign_class.py loads in the dataset of banknotes, and assigns the proper color to each entry. When uncommented, line 99 displays the dataframe after the csv file is loaded in and the setColor method is applied to each row.

Part 2 - Within assign_class.py, the function analyzeFrame takes the dataframe created in part one, and calculates the mean and standard deviation for each class and each feature. To see the information displayed, uncomment lines 85 through 95.

*note - all amounts are shown rounded to the nearest 2 decimal places. Unrounded values are shown in the outputs when the python files are run.

Class	μ(f1)	σ(f1)	μ(f2)	σ(f2)	μ(f3)	σ(f3)	μ(f4)	σ(f4)
0	2.28	2.02	4.26	5.14	0.80	3.24	-1.15	2.12
1	-1.87	1.88	-0.99	5.40	2.15	5.26	-1.25	2.07
All	0.43	2.84	1.92	5.87	1.40	4.31	-1.91	2.10

Part 3 - The mean of the variance and skewness show a positive trend for the real notes and a negative trend for the fake notes. One other trend that appears noteworthy is the curtosis (f3) feature of the real notes has a lower value mean than the fake notes and also has a much more noticeable difference in the value of the standard deviation, whereas all the other features have similar standard deviation values.

Question 2:

Part 1 - classifier.py contains the function generatePairwise which takes the dataset X and splits it into training and testing parts with a 50/50 split. It also saves the train and test sets as csv files for consistency in the rest of the question.

Part 2 - When comparing my two plots, a majority of the real notes have a majority between variance, skewness, and curtosis greater than 0. Therefore, my classifier will look like:

```
if (f_1 > 0 & f_2 > 0) or (f_1 > 0 & f_3 > 0) or (f_1 > 0 & f_3):
        x = 'good'
else:
        x = 'bad'
```

Part 3 - Within classifier.py, the function simpleClassifier takes the test dataset and applies the simple classifier mentioned in part 2.

Part 4 and 5 -

*note - all percentages are shown rounded to the nearest 2 decimal places. Unrounded values are shown in the outputs when the python files are run.

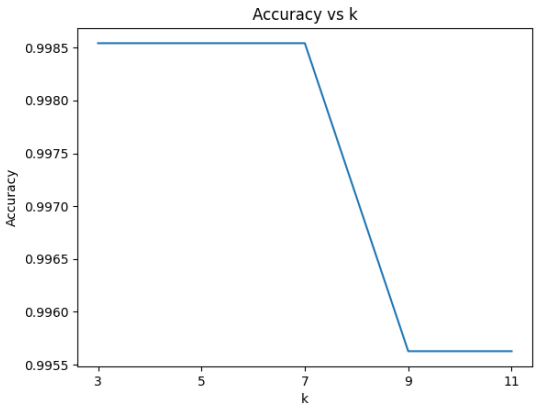
TP	FP	TN	FN	Accuracy	TPR	TNR
358	20	217	91	83.82%	79.73%	91.56%

Part 6 - My simple classifier gives me a higher accuracy when identifying fake bills, as my TNR was around 12% better than my TPR. My accuracy calculated to 83.82% which is noticeably more accurate than a coin flip.

Question 3:

Part 1 - The function knn in knn.py takes k = 3, 5, 7, 9, 11, trains the k-NN classifier on the training set, then computes the accuracy for the testing set.

Part 2 -



After computing the accuracy for the 5 values of k, the optimal values is 7, where k = 3, 5, and 7 produce the same accuracy.

Part 3 - Since k = 3, 5, 7 are all equal in accuracy, the optimal k value is k = 7 was used to compute the performance measures for this table:

*note - all percentages are shown rounded to the nearest 2 decimal places. Unrounded values are shown in the outputs when the python files are run.

TP	FP	TN	FN	Accuracy	TPR	TNR
377	1	308	0	99.85%	100%	99.68%

Part 4 - The k-NN classifier is much better than my simple classifier for all of the measures from my previous table. Of the 686 entries, there was only 1 incorrect prediction compared to the 101 incorrect prediction from my simple classifier.

Part 5 - If a bill x contained the last 4 digits of my BUID, the values would be f_1 = 3, f_2 = 3, f_3 = 6, and f_4 = 0. Because my simple classifier checks if at least 2 values in f_1, f_2, and f_3 are positive, bill x will be labeled as real. When using k = 7, the k-NN prediction is also real.

Question 4:

Part 1 - knnFeatureSelect in knn.py takes the best value of k, and computes the accuracy of k-NN when missing each of the 4 features.

Part 2 - The accuracy only decreased when removing any of the four features. No single feature removed resulted in an increase in accuracy compared to when all features were used.

Part 3 - Variance contributed the most to loss of accuracy where k-NN was 93.88% accurate.

Part 4 - Entropy contributed the least to loss of accuracy where k-NN was 98.69% accurate.

Question 5:

Part 1 - logReg in logistic.py trains the logistic regression classifier on the training dataset, and computes the accuracy for the testing dataset.

Part 2 -

*note - all percentages are shown rounded to the nearest 2 decimal places. Unrounded values are shown in the outputs when the python files are run.

TP	FP	TN	FN	Accuracy	TPR	TNR
376	2	307	1	99.56%	99.73%	99.35%

Part 3 - The logistic regression is more accurate than my simple classifier, where all measures are more accurate.

Part 4 - The logistic regression is slightly less accurate than the k-NN in the previous table with 2 false positives and 1 false negative compared to the 1 false positive and 0 false negatives that k-NN predicted.

Part 5 - If bill x is predicted by logistic regression, it is predicted to be real in addition to the k-NN prediction.

Question 6:

Part 1 - logRegFeatureSelect in logistic.py computes the accuracy for each feature to be missing.

Part 2 - No, in the 3 cases of missing variance, skewness, and curtosis, the accuracy was lower than the accuracy computed with all features. However, with the entropy feature missing, the accuracy calculated was the same as the accuracy for all 4 features.

Part 3 - Variance contributed the most to loss of accuracy where logistic regression was 81.20% accurate.

Part 4 - Entropy contributed the least to loss of accuracy where logistic regression stayed the same at 99.56% accurate.

Part 5 - Yes, in both k-NN and logistic regression, variance was the most significant feature and entropy was the least significant feature whereas skewness and curtosis were similar in value in between variance and entropy.