Martin Lahoumh
CSC 59866
Proffesor Yu
March 9 2024

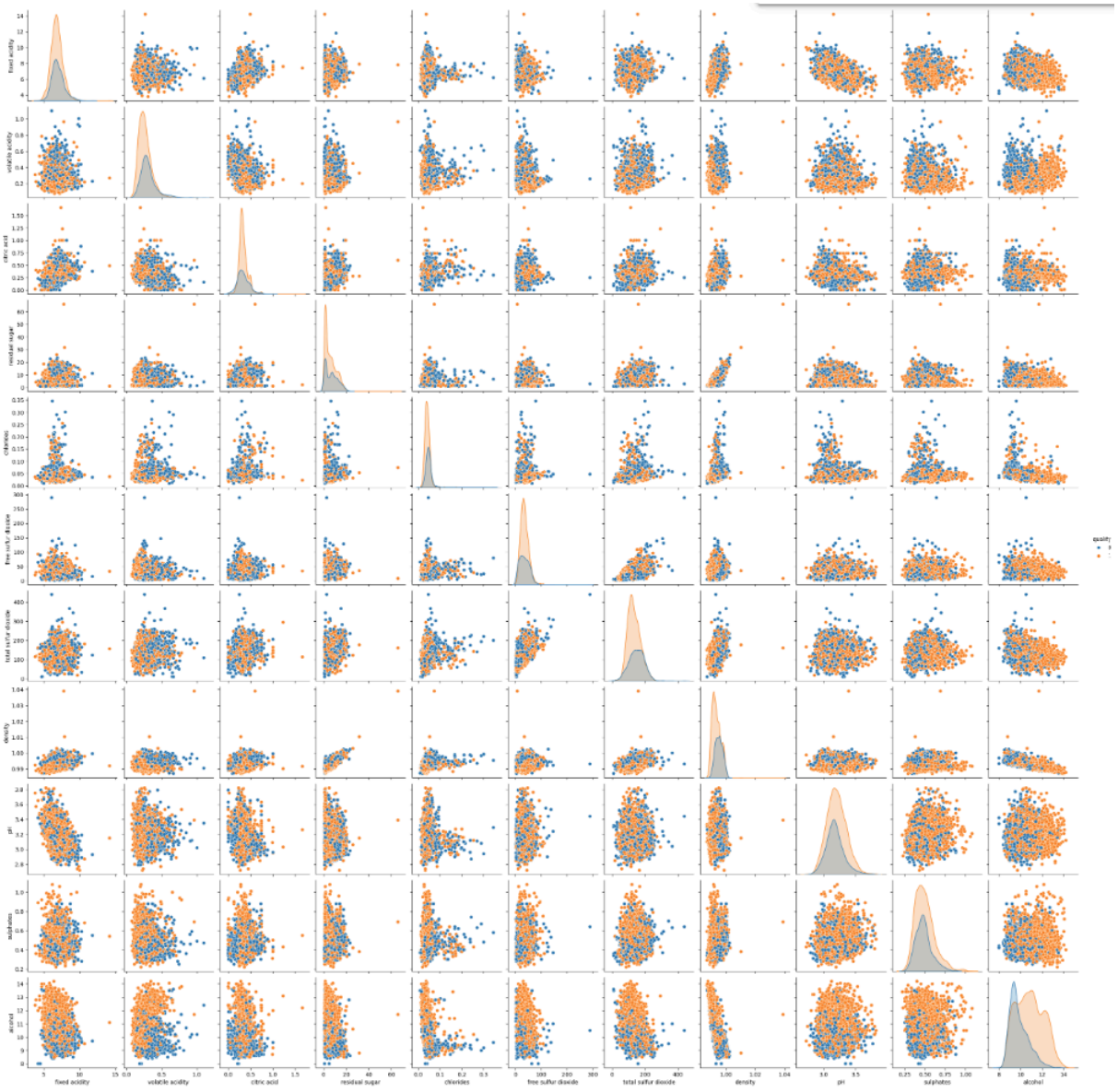
The csv dataset given by Professor Yu contained different qualities of wine, which included: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. Given all of these attributes, it was out job to create a model that would be able to predict the quality of the wine. The csv file given was not formatted regularly, so all columns had to be separated based on the ‘;’

```
# Reads the CSV file
df = pd.read_csv(io.BytesIO(uploaded['winequality-white.csv']), sep=";")
#seperate the quality column based on when values are greater than 5 and when they are not
df['quality'] = df['quality'].apply(lambda x: 1 if x > 5 else 0)

# Shuffles the row in the data
dfShuffled = df.sample(frac=1)

df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415	0.045772	35.308085	138.360657	0.994027	3.188267	0.489847	10.514267	0.665169
std	0.843868	0.100795	0.121020	5.072058	0.021848	17.007137	42.498065	0.002991	0.151001	0.114126	1.230621	0.471979
min	3.800000	0.080000	0.000000	0.600000	0.009000	2.000000	9.000000	0.987110	2.720000	0.220000	8.000000	0.000000
25%	6.300000	0.210000	0.270000	1.700000	0.036000	23.000000	108.000000	0.991723	3.090000	0.410000	9.500000	0.000000
50%	6.800000	0.260000	0.320000	5.200000	0.043000	34.000000	134.000000	0.993740	3.180000	0.470000	10.400000	1.000000
75%	7.300000	0.320000	0.390000	9.900000	0.050000	46.000000	167.000000	0.996100	3.280000	0.550000	11.400000	1.000000
max	14.200000	1.100000	1.660000	65.800000	0.346000	289.000000	440.000000	1.038980	3.820000	1.080000	14.200000	1.000000

With the data, different pairplots were printed out to see the correlation between the different variables. To make the data set stronger, redundant information was removed (information that contained a higher correlation than the threshold 0.6).

Before Dropped:



After Dropped:

To see how accurate the predictions were, the accuracy was calculated using a function that was made for this project in partA.

```
#Calculate the accuracy and error of the model predictions compared to the test dataset and the f1
unscaledAccuracy, unscaledError = accuracyGeneralization(yTest, unscaledPredictions)


#Get the average and standard deviation for futhur analysis
meanValues = np.mean(XTrain, axis=0)
stdValues = np.std(XTrain, axis=0)
XTrainStandardized = (XTrain - meanValues) / stdValues
XTestStandardized = (XTest - meanValues) / stdValues

# results
print("Unscaled Data Results:")
print(f"Accuracy: {unscaledAccuracy:.2%}")
```

```
Unscaled Data Results:
Accuracy: 67.65%
```

The accuracy was about 67%. The error for the model was also low. Next was to standardize the results

```
#Now that we trained our model, we can train the model on the standardized data that we created
# Re runs the KNN Classifier method on the new standarized data
model.fit(XTrainStandardized, yTrain, n_neighbors=5, weight='uniform')
predictionsStandardized = model.predict(XTestStandardized)

#accuracy and f1 score for furthur analysis of the model
accuracy_standardized, error_standardized = accuracyGeneralization(yTest, predictionsStandardized)


print("\nResults for Standardized Data:")
print(f"Accuracy: {accuracy_standardized:.2%}")
```

```
Results for Standardized Data:
Accuracy: 74.08%
```

We see a much higher accuracy to the standardized compared to the unscaled accuracy. The standardized came in at about 74%, whereas the unscaled accuracy was 67%. Using the new standardized test and train data, it was rerun on the model.