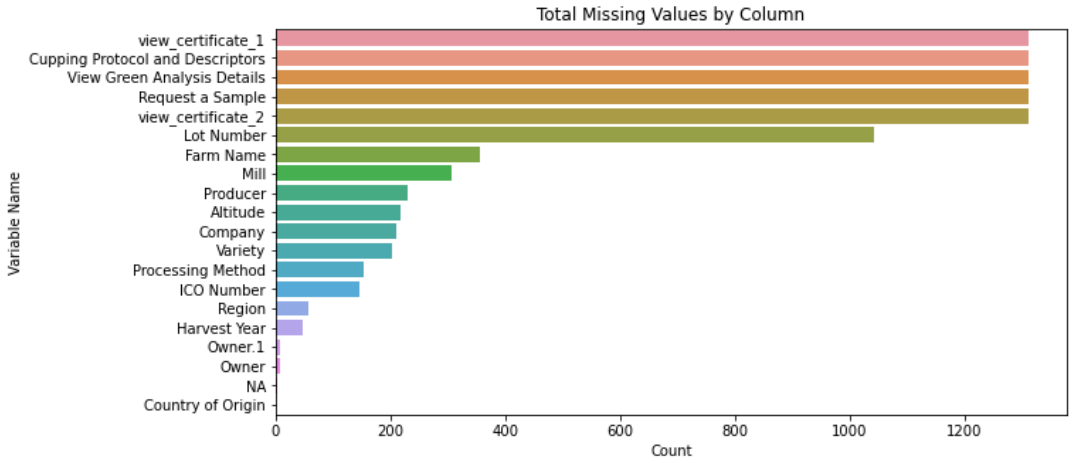
Dataset Description and Analysis

The tabular dataset comes from the Coffee Quality Institute (CQI) database (which contains coffee review data, including sensory / chemical / metadata). It is posted publicly on GitHub by James LeDoux who obtained the data by scraping the CQI database in 2018. We chose this dataset because it contained a scoring variable for coffee beans which we needed to create prediction models to answer the research question. The Github page states the data is open source under an MIT License. The dataset contains 1310 rows and 51 columns. Out of the 51 columns, most of them are numeric variables while a few are categorical. Some examples of the numerical variables are quality ratings for aroma, acidity, and sweetness. Examples of categorical variables include color, processing method, and the origin of the coffee beans. The target variable we are looking to predict is the quality score of the coffee beans, which is a numeric score that the CQI rates between 0 and 100. A summary statistic of the numeric variables after data cleaning is shown below.

Summary Statistics

| | quality_score | Aroma | Flavor | Aftertaste | Acidity | Body | Balance | Uniformity | Clean Cup | Sweetness | Cupper Points |
|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 | 1312.000000 |
| mean | 82.086212 | 7.562614 | 7.516913 | 7.396822 | 7.532614 | 7.516570 | 7.516349 | 9.825899 | 9.825625 | 9.895724 | 7.496723 |
| std | 3.675542 | 0.380976 | 0.402017 | 0.406202 | 0.381879 | 0.361512 | 0.408316 | 0.621552 | 0.817449 | 0.596925 | 0.476228 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 81.170000 | 7.420000 | 7.330000 | 7.250000 | 7.330000 | 7.330000 | 7.330000 | 10.000000 | 10.000000 | 10.000000 | 7.250000 |
| 50% | 82.500000 | 7.580000 | 7.580000 | 7.420000 | 7.500000 | 7.500000 | 7.500000 | 10.000000 | 10.000000 | 10.000000 | 7.500000 |
| 75% | 83.670000 | 7.750000 | 7.750000 | 7.580000 | 7.750000 | 7.670000 | 7.750000 | 10.000000 | 10.000000 | 10.000000 | 7.750000 |
| max | 90.580000 | 8.750000 | 8.830000 | 8.670000 | 8.750000 | 8.580000 | 8.750000 | 10.000000 | 10.000000 | 10.000000 | 10.000000 |

The raw dataset had many missing values and undescriptive or irrelevant columns. For these reasons, we had to clean the dataset before doing any analysis on the data. We were easily able to remove over 20 columns that were clearly not useful for the prediction of the quality score. We programmatically removed columns that had over 1000 missing values, and also removed columns with only a singular unique value.



The dataset also contained numeric variables that had to be extracted from string and converted to floats. To extract the numbers from the strings, we used regular expressions along with the str.extract() function from the pandas library.

Before Manipulating Strings

| | Total Cup Points | Moisture | Category One Defects | Category Two Defects |
|---|---|---|---|---|
| 0 | Sample 90.58 | 12 % | 0 full defects | 0 full defects |
| 1 | Sample 89.92 | 12 % | 0 full defects | 1 full defects |
| 2 | Sample 89.75 | 0 % | 0 full defects | 0 full defects |
| 3 | Sample 89.00 | 11 % | 0 full defects | 2 full defects |
| 4 | Sample 88.83 | 12 % | 0 full defects | 2 full defects |

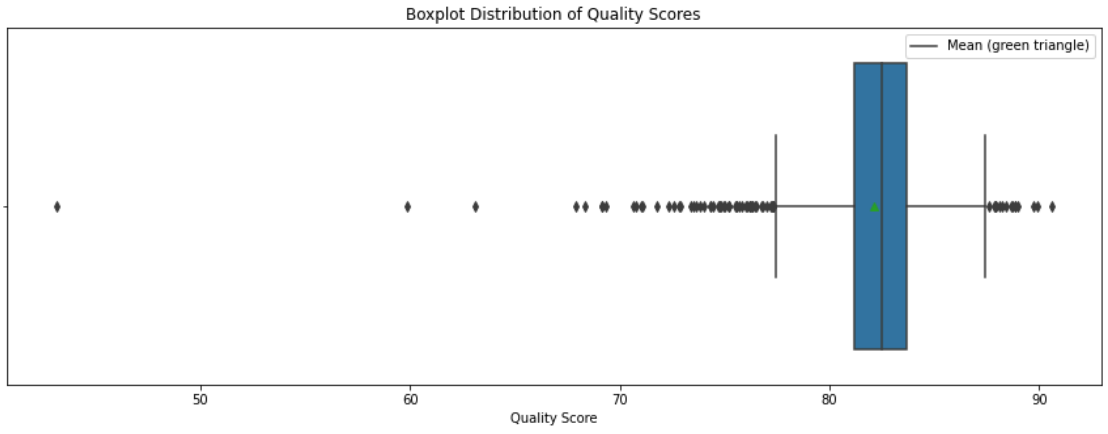
After Manipulating Strings

| | Total Cup Points | Moisture | Category One Defects | Category Two Defects |
|---|---|---|---|---|
| 0 | 90.58 | 12.0 | 0.0 | 0.0 |
| 1 | 89.92 | 12.0 | 0.0 | 1.0 |
| 2 | 89.75 | 0.0 | 0.0 | 0.0 |
| 3 | 89.00 | 11.0 | 0.0 | 2.0 |
| 4 | 88.83 | 12.0 | 0.0 | 2.0 |

After finishing cleaning the data column by column, we checked row data to ensure good data quality. We manually checked rows that took our attention. For example, there was only one row with a coffee quality score of 0. There seems to be something wrong with this row as most of the values are defaulted to 0. For this reason, we have manually dropped this row. We manually checked a few other examples that stood out, but decided not to take any action as there didn't seem to be anything wrong with them.

| quality_score | Country of Origin | Processing Method | Aroma | Flavor | Aftertaste | Acidity | Body | Balance | Uniformity | Clean Cup | Sweetness | Cupper Points | Total Cup Points | Moisture | Category One Defects |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0 | Honduras | NaN | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 12.0 | 0.0 |

Although the quality score is a value defined between 0 and 100, the distribution of quality scores follows a normal bell curve shape with a mean and median relatively close together around 85, meaning it isn't very skewed. There aren't many low quality scores, so we can expect data points with low quality scores to be very influential when we create prediction models.




Lastly, we calculated the pairwise correlation matrix for all the variables. By doing so, we found that quality score and total cup points had a correlation coefficient of 1. This means they are

linearly calculated from one another. Using total cup points would give perfect model predictions, therefore, we removed total cup points. We also found that flavor, aftertaste, and balance were very positively correlated to the quality score. We can expect them to be impactful features when we conduct our experiments.

Sources and References

Seaborn plotting syntax and plot choices for EDA
https://www.geeksforgeeks.org/data-visualization/types-of-seaborn-plots/
Used code df.isnull.sum() from this source in the dealing with missing values portion
https://stackoverflow.com/questions/26266362/how-do-i-count-the-nan-values-in-a-column-in-pandas-dataframe
Took inspiration for regular expressions for string extraction from pandas documentation to transform strings to floats
https://pandas.pydata.org/docs/reference/api/pandas.Series.str.extract.html
Read and used code from this article to create the correlation matrix using the Seaborn library
https://medium.com/@szabo.bibor/how-to-create-a-seaborn-correlation-heatmap-in-python-834c0686b88e
Used code for normalizing data from pandas's documentation
https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html