Scott Breitbach
30-June-2022

Rapid Microbe Identification

White Paper

Business Problem:

Rapid microbial identification.

The goal of this project is to support rapid clinical diagnoses through automatic machine identification of microbes using measurements of cellular features generated from microscopic imagery.

Background / History:

When dealing with an infection, time is an important factor because doubling times for many bacteria are often measured in hours and sometimes even minutes. Culturing bacteria for identification via genetic analysis and other methods is often time consuming and having technicians manually identify microbes can vary from person to person and is often subjective.

Rapid and accurate identification of bacteria can help to support appropriate treatment options, saving lives and potentially limiting further development of antibiotic-resistant forms of bacteria.

Some of my inspiration for selecting this project comes from a product my employer sells to veterinarians that uses machine learning recognition of high-resolution microscope images to assist in diagnosing parasites that show up in fecal scans.

Data Explanation:

About the dataset:

To establish an initial proof of concept for microbe identification, I will utilize a dataset available on Kaggle, appropriately named “Microbes Dataset.” The dataset contains a little over 30,000 observations consisting of unique serial numbers, 24 numeric features and a single, categorical microbe name target.

The target variable is made up of 10 different microbes and is imbalanced, with the most common, Ulothrix, accounting for almost ¼ of the observations, while the least common, Spirogyra, accounts for only around 2% of the observations. Most of the rest make up around 6-13% each.

The numeric features are measurements automatically generated from images taken via microscopy. These consist of some basic measurements like Area or Perimeter of the microbe, length of Major (length) and Minor (width) axes, or the Convex Hull area, which is like if you imagine a rubber band wrapped around an object that has indentations so that the rubber band jumps the gaps and you take the area inside the rubber band. There are also engineered features that are calculated from other features (mostly ratios) such as Eccentricity, which is the ratio of the major and minor axes and Solidity, which is the ratio of the Area to the Convex Hull. Some features are just different measurements of size, like Bounding Box, which has to do with the size of the smallest rectangle that bounds the object.

Other features I have more difficulty seeing how they will be useful in identification, such as Centroid which tells where the microbe's center lies in the image or Orientation, which tells which direction the microbe is angled. Still others I'm not sure I fully understand but will keep in the data in case they prove to be useful.

It turns out that ConvexHull1 and ConvexHull2 are identical, so ConvexHull2 was removed before moving on to anything else. While I'm not sure why this duplicate information exists, it might be worth looking into further, in case another potentially useful measurement was inadvertently written over in preparing the data for Kaggle.

Distributions:

All the measured values in the data have already been scaled to a range of 0 – 23. Distribution of two of the features (Eccentricity and EulerNumber) show a negative skew and eight (EquivDiameter, FilledArea, BoundingBox 3-4, Major & MinorAxisLength, Perimeter, and ConvexArea) show a relatively strong positive skew. The remaining features show relatively Gaussian to uniform distributions.

See Appendix, Table 1: Histograms

Looking at the distributions of the raw data in the histograms and box plots I noticed that some had higher counts at zero than I would expect, and all the features had values out to 23 even if these values were apparent outliers. Giving some thought to the features themselves, I realized that several of the measurements, like area or length, should not include any zero values. I removed the rows with zero values from those columns. Similarly, I removed rows containing values of 23 from columns that looked like the 23s were probable outliers. This removed 212 observations, or 0.7% of the data, which I feel is a relatively minimal loss to the dataset.

See Appendix, Table 2: Box and Whisker Plots

Splitting:

At this point, I split the data into training and testing sets. This was done prior to scaling of the data, to prevent leaking of information that might occur in the transformation methods used.

Transformations:

Given that around half of the features showed moderate to high skew, I considered addressing each feature separately and doing log, square, or other transformations as seemed appropriate for each variable. In the interest of time and simplicity and because all the features are already all positive values, I opted instead to perform Box-Cox transformations on all the features. This may not be the optimal solution for every feature, but I feel it does lead to an overall net gain for usability of the dataset. Additionally, it helps to make both the positively and negatively skewed data more Gaussian as well as the uniformly distributed data.

See Appendix, Table 3: Box-Cox Transformation

Feature Selection:

Using a Pearson Ranking of the feature variables shows some strong correlations between several groups of variables.

See Appendix, Table 4: Pearson Ranking Chart

Using the ANOVA F-test showed some low scores for 9 of the 23 feature variables, but it was determined that the optimal number of features to keep is 23, or all of them. This was not using the transformed data however, and the target classes remain imbalanced.

See Appendix, Table 5: ANOVA F-test Results

Ultimately, all 23 feature variables were left in place for modeling.

Methods:

Several things that I considered in generating a useful predictive model from this dataset are: imbalanced target variable, which algorithms are appropriate, and how to compare the algorithms.

Balancing:

It is known that imbalanced data can lead to a poor model. With one of the ten categories in the target data being significantly overrepresented and one significantly underrepresented, the imbalance needs to be addressed. To deal with this, one common option is to sample from the overrepresented class do reduce its size, and/or resample from underrepresented classes to increase their size. Instead, I opted to utilize the `class_weight` parameter, available in many scikit-learn models, setting it to 'balanced'. This applies weighting to different classes inversely according to their size. If time allows, I would like to revisit the resampling option.

Algorithm selection:

In selecting the appropriate classification model, my strategy is a scatter-shot approach, trying a bunch of models to see which perform well with the dataset. Accuracy was used as an initial metric to weed out poorly performing models. However, since the dataset is imbalanced, accuracy can be a misleading metric for model suitability. Other performance metrics that can be used to compare models include precision, recall, and a confusion matrix.

Analysis:

Compared to a default baseline of 10% accuracy (based on randomly guessing one of the 10 classes) or even 25% accuracy (based on the imbalanced classes, if every prediction was Ulothrix), all 9 of the models I tried performed better than chance, although some only modestly better. The worst models for this proved to be the Ada Boost Classifier at 30% accuracy and the Gaussian Naïve Bayes Classifier at 33% accuracy. The best performing models were the Random Forest Classifier and the Decision Tree Classifier, both at or above 98% accuracy.

To compensate for the imbalanced classes, I initially included "*class_weight='balanced'*" in most of the model's parameters, but it did not appear to have much impact on accuracy. Of note, the two poorest performing models, did not have a balanced class weight option. I later balanced my training dataset by sampling from the overrepresented classes and resampling the underrepresented classes, keeping the overall number of samples in the training set the same. Unfortunately, this resulted in slightly poorer results across all models in all metrics. I suspect this is largely because I eliminated approximately ¼ of the information that was previously available. Given enough time, it would be interesting to try

comparing the models again with resampling of all the less represented classes up to the count of the most common class, possibly using something like SMOTE for resampling.

I did get an opportunity to do some hyperparameter tuning of my best performing models. While the improvements were relatively minor, I was able to increase accuracy of the Random Forest and Decision Tree models from 98.2% and 98.0%, respectively, both to 98.7%

Conclusion:

While the Random Forest model performed slightly better than the Decision Tree model, I am leaning toward using the Decision Tree for implementation of the application because it is much easier to interpret if we need to dig in and do troubleshooting or it may provide clues for model improvement.

Also, the Random Forest model was quite fast to train, at around 5 seconds, but the Decision Tree, clocking in around half a second, was about 10 times faster.

Assumptions:

I think the biggest assumption here, which is the case with many datasets, is that the recorded data is representative of the larger environment. There are ways in which bias could be inadvertently introduced into the data. For example, it is unclear how the microorganisms were propagated and isolated prior to measurement. Organisms that are growing inside an animal may display subtle variations in characteristics as compared to organisms that are grown in a flask with media. The media is likely to provide optimal nutrients and environmental conditions with no competition from other organisms or from the host itself, which may impact how quickly and how large a cell might grow, as well as other features.

Limitations:

The imbalance of the target features is one limitation. At 10 categories, with the smallest consisting of ~600 entries, the entire dataset would only be ~6,000 if we were to sample the larger categories down to that size (about 1/5 of the current size). Of course, the smaller categories can be resampled to increase their size, but this comes with its own limitations. Ideally more real-world data could be generated to balance the dataset.

The measurements used are another potential limitation. It is unclear why each of the measurements was selected and what some of them represent. Several of the measurements appear to be at least partially redundant and it is unknown whether additional features exist that might prove useful. The lack of clarity in the descriptions of some of these features also adds a layer of difficulty to the interpretability of the model.

Perhaps the greatest limitation is that some of the microorganisms listed represent large categories of organisms, like Yeast, Protozoa, and Diatoms. Each of these may display a wide variety of morphologies depending on the species or subspecies, which may have an outsized effect on the measured features found within an individual grouping.

Challenges:

One challenge I ran into was that, while some algorithms will automatically handle my target labels as strings, others require the targets be converted to a numerical value. Because the labels are distinct from one another, I needed to ensure that they were encoded in a "one-hot" fashion rather than integer/ordinal encoding, since I don't want the algorithm to interpret the label assigned a value of 2 as being twice the microorganism than the label assigned a value of 1 is (which wouldn't make any sense).

I think I finally have it figured out that scikit-learn's LabelEncoder and LabelBinarizer are for encoding target labels, while OrdinalEncoder and OneHotEncoder are for encoding other categorical features. That said, I still sometimes have trouble getting my target variables encoded correctly for use in neural networks.

Future Uses / Additional Applications:

This algorithm could be expanded to include several other organisms, potentially providing more specific categories down to the species level, though that may not be necessary for the purpose of diagnostics in animal care.

Additionally, I could see this being used with environmental or even process monitoring in cleanroom environments, to quickly identify contaminants to help determine the appropriate remediation methods.

Recommendations:

Use of ensemble models may prove useful in improving accuracy of the model when predicting organisms, especially as more precision is needed to differentiate between similar and closely related species.

Instead of building a model based on measurements selected by humans and what they think are the best characteristics, perhaps training a neural network on the images themselves and allowing the algorithm to extract the important features directly would result in better classification.

Implementation Plan:

Upon establishing feasibility, additional data will need to be generated to fine tune the algorithm and to expand it to include additional organisms that are immediately relevant to animal care.

Implementation of this algorithm will involve standardizing a methodology for isolating and imaging microorganisms so the appropriate measurements can be made. These measurements will be automatically input into the algorithm and a prediction made. It may prove useful to indicate the percent likelihood of the prediction and possibly include all predictions above a certain threshold to communicate more effectively that algorithm predictions aren't infallible, and to provide backup options in case the predicted class is incorrect.

Ethical Assessment:

It is important to get the predictions right. The health of animals is at stake and providing an incorrect prediction could impact the treatment chosen. This could mean the difference between life and death for the animal.

Securing patient data is another concern. Maybe not so much for animals, but if at some point this is used with human pathogens, the data stored would need to be protected, possibly anonymized. This could be done by using serial numbers for the tests and linking the results to the patient in their personal records, so that no patient information is introduced to the software.

References:

Brownlee, Jason. “8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset.” Machine Learning Mastery, 15 Aug. 2020, https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/.

Brownlee, Jason. “How to Perform Feature Selection with Numerical Input Data.” Machine Learning Mastery, 18 Aug. 2020, https://machinelearningmastery.com/feature-selection-with-numerical-input-data/.

Brownlee, Jason. “Random Oversampling and Undersampling for Imbalanced Classification.” Machine Learning Mastery, 4 Jan. 2021, https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/.

Dhindsa, Anaahat ; Bhatia , Sanjay; Agrawal , Sunil ; Sohi , B.S. (2020), “Dataset for Efficient Microbes Classification System ”, Mendeley Data, V3, doi: 10.17632/f9m85ptmvc.3

Durgance, Gaur. “A Guide to Any Classification Problem.” Kaggle, Kaggle, 15 Feb. 2022, https://www.kaggle.com/code/durgancegaur/a-guide-to-any-classification-problem.

Lab, WIRED Brand. “Wired Brand Lab | Cloud to Clinic: Zoetis' Vision for Veterinary Practices.” Wired, Conde Nast, 27 Apr. 2022, https://www.wired.com/sponsored/story/cloud-to-clinic-zoetis-vision-for-veterinary-practices/.

Saha, Sayan. “Microbes Dataset.” Kaggle, 12 Apr. 2022, https://www.kaggle.com/datasets/sayansh001/microbes-dataset.

Statista Research Department. “Dog and Cat Spending per Year U.S. 2020.” Statista.com, 13 Dec. 2021, https://www.statista.com/statistics/250851/basic-annual-expenses-for-dog-and-cat-owners-in-the-us/.

sangy987. “Imbalanced-Learn Module in Python.” GeeksforGeeks, 11 Dec. 2020, https://www.geeksforgeeks.org/imbalanced-learn-module-in-python/.

Appendix:

Table 1: Histograms

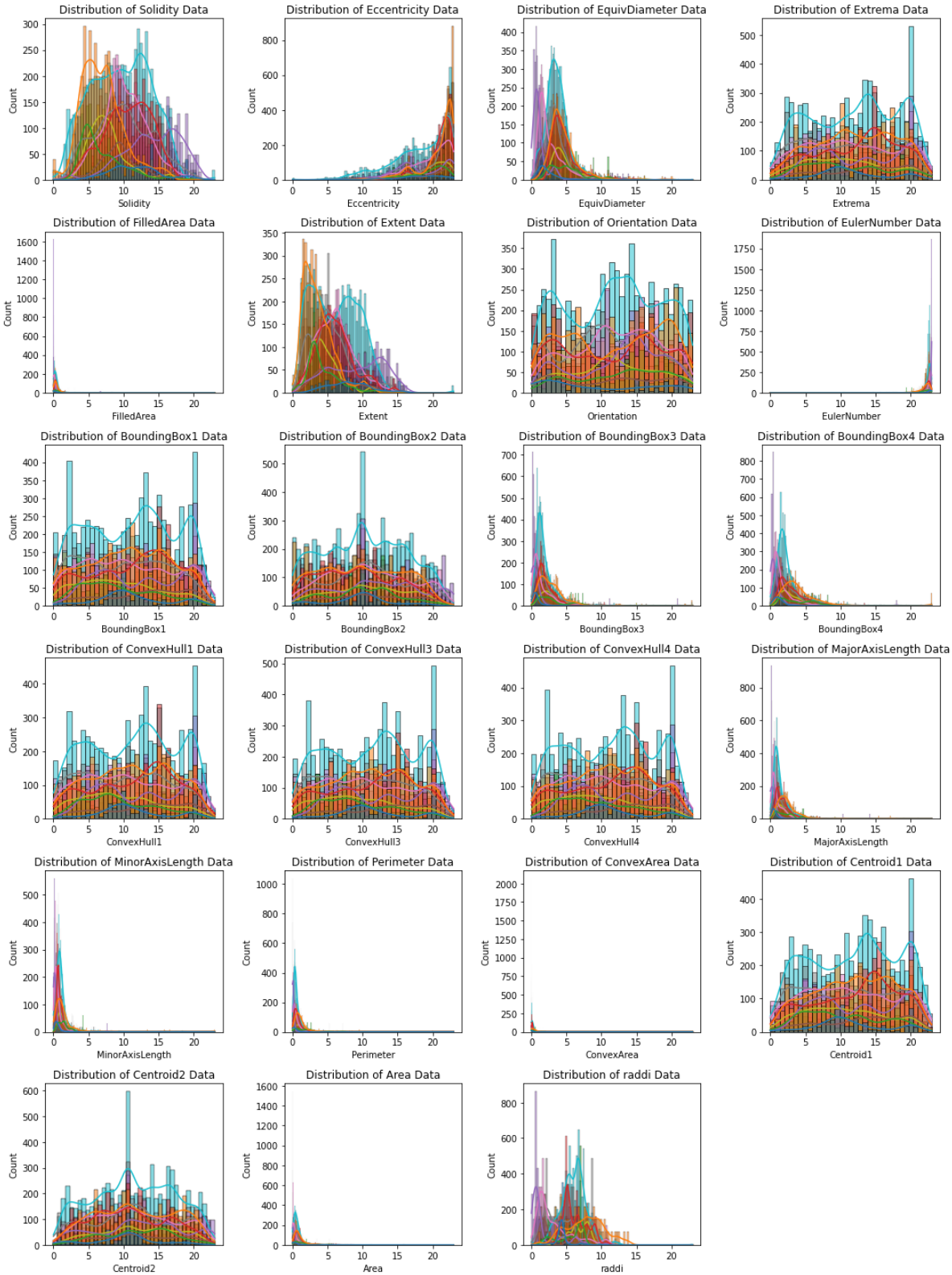


Table 2: Box and Whisker Plots

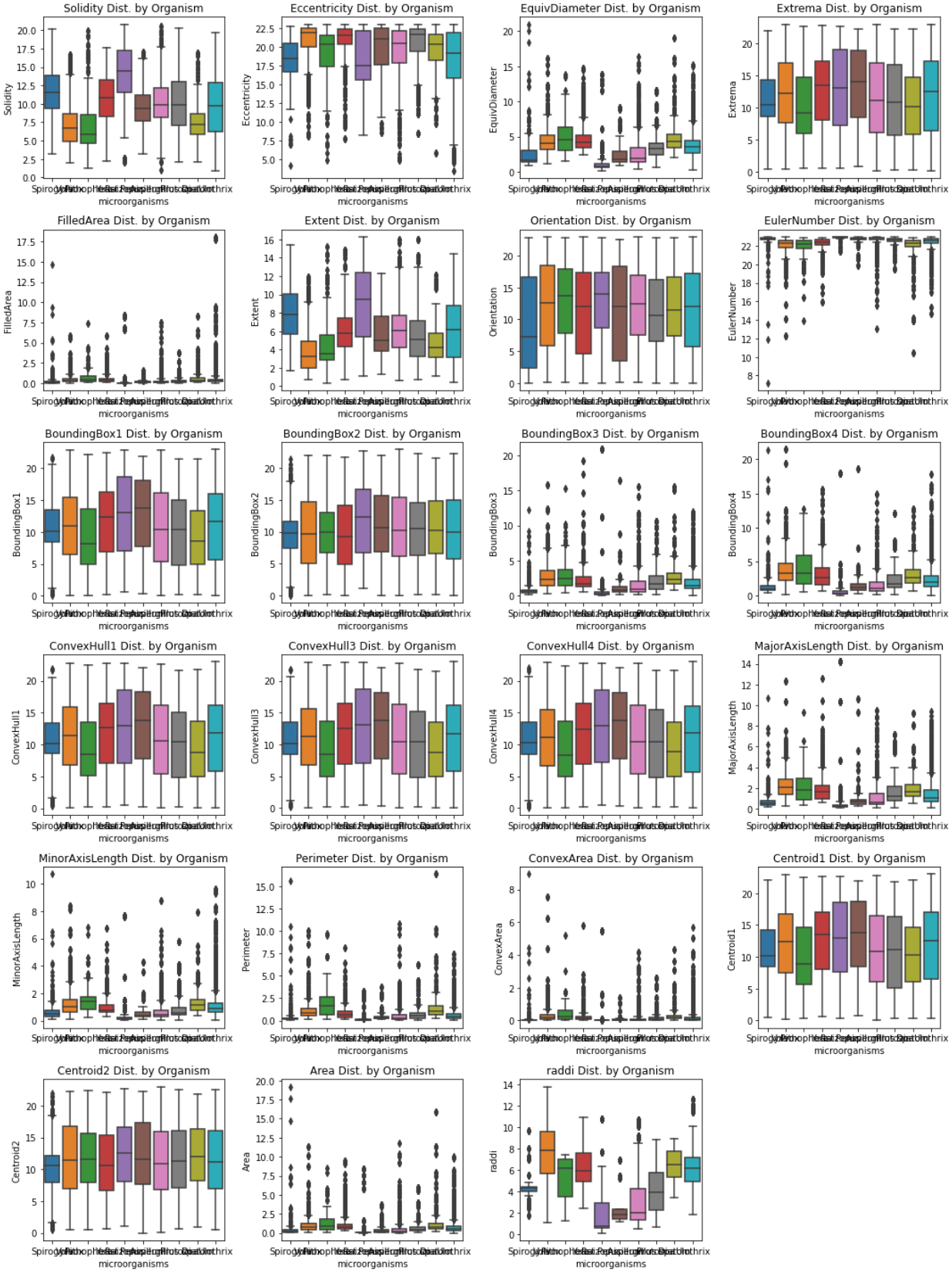
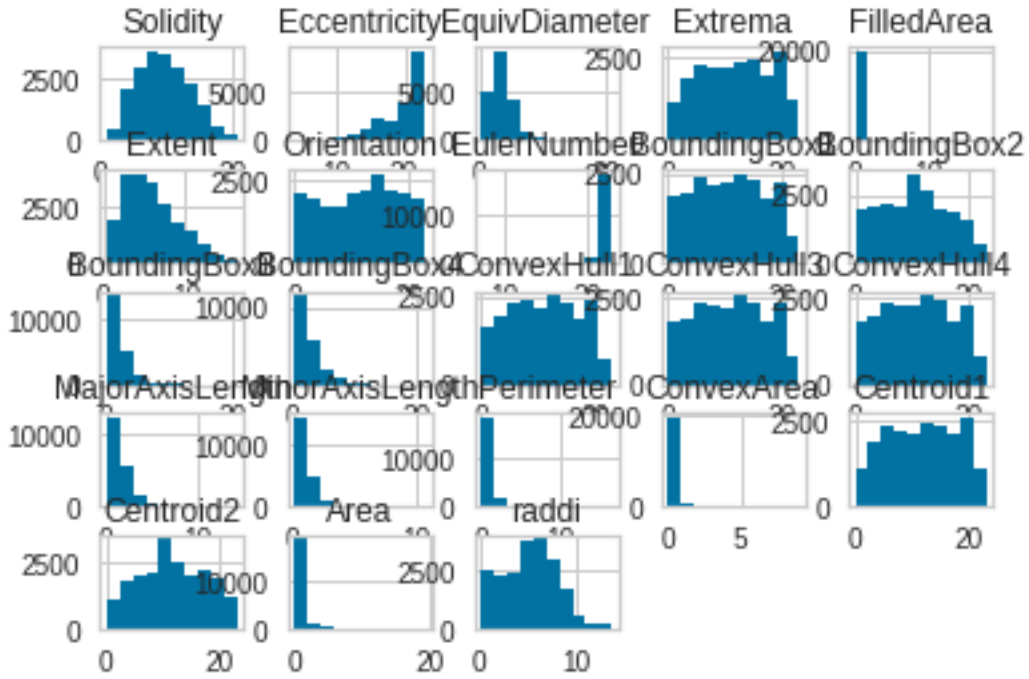


Table 3: Box-Cox Transformation

3a: Feature distributions before Box-Cox Transformation



3b: Feature distributions after Box-Cox Transformation

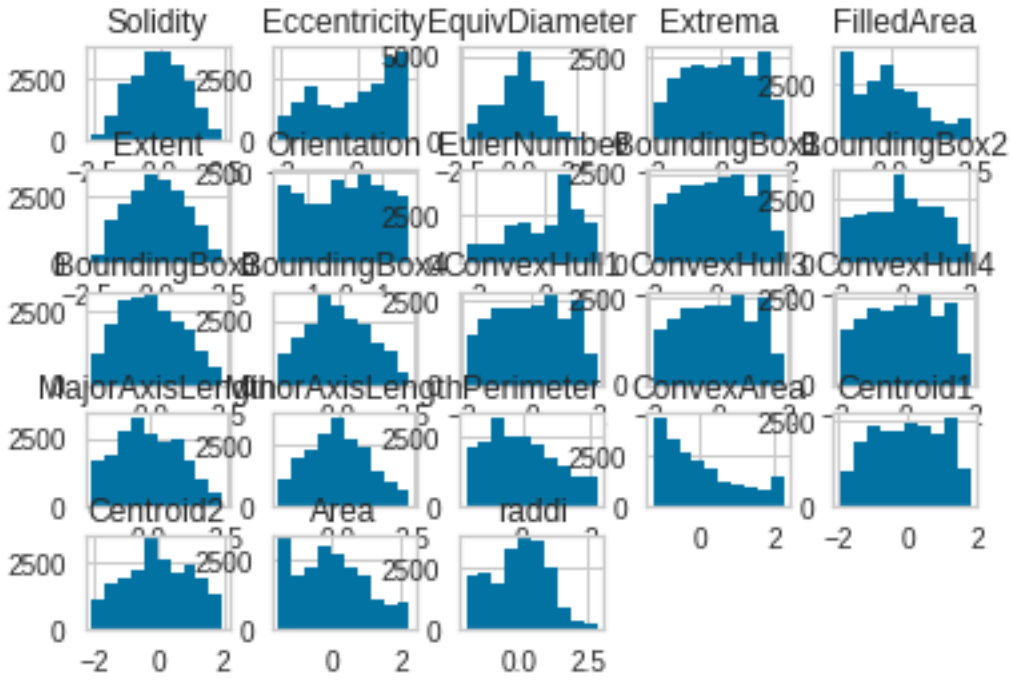


Table 4: Pearson Ranking Chart

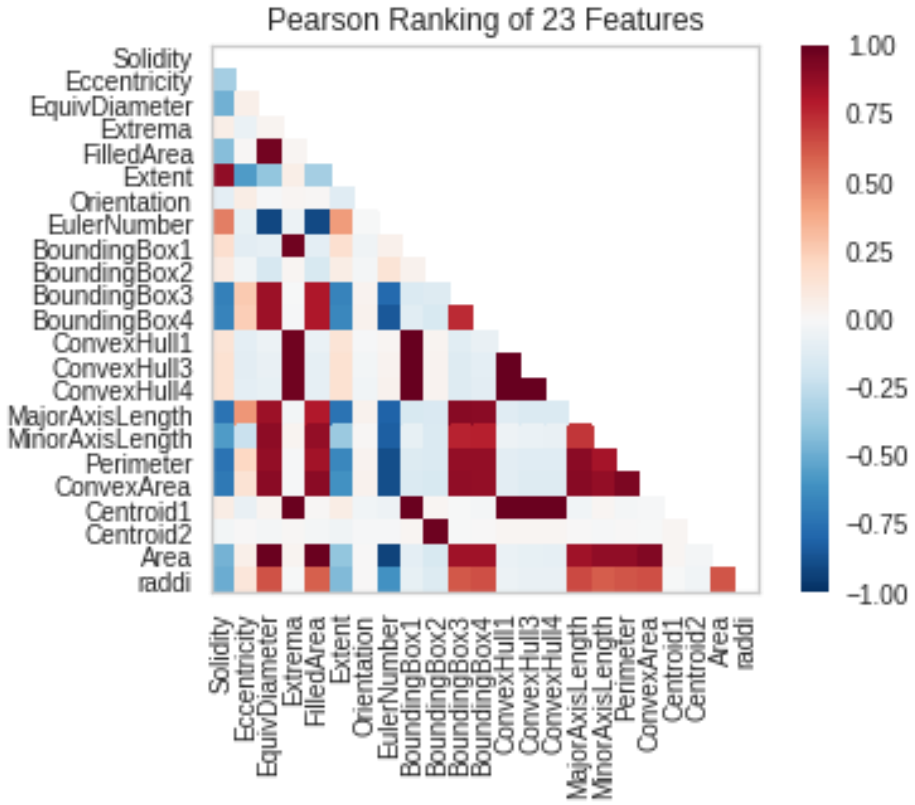


Table 5: ANOVA F-test Results