COMP4702 Assignment

Jack Wallace
48613167

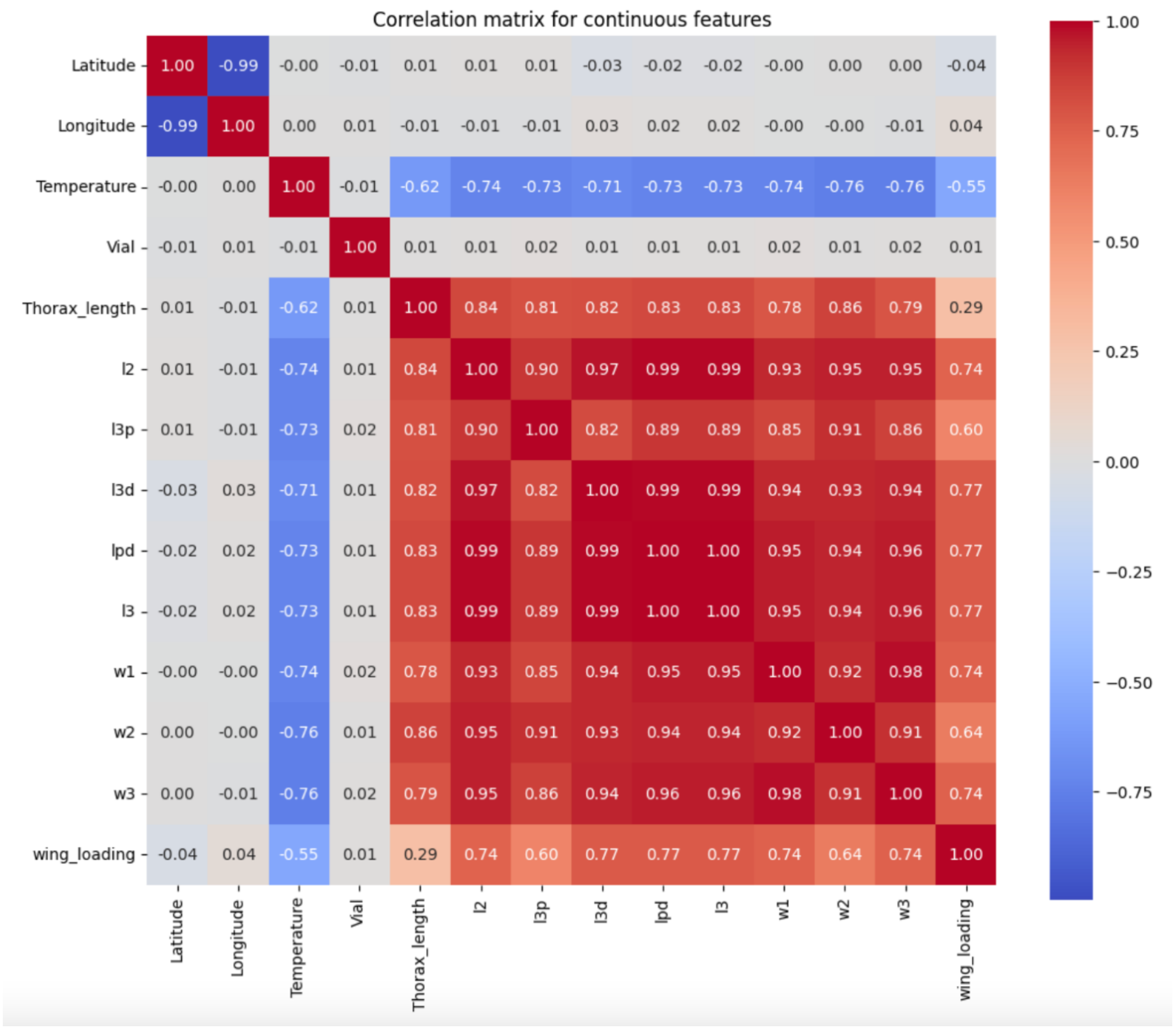
Table of Contents

Introduction and Preprocessing

The data in “83_Loeschcke_et_al_2000_Thorax_&_wing_traits_lab pops.csv” has various column characteristics about fruit flies, there are 20 columns and 1731 rows. The 20 columns represent the characteristics, such as geography, sex, and various measurements and the 1731 rows represent each a unique a fruit fly. The goal of this analysis is to understand how to accurately predict if a fruit fly is male or female based off these characteristics. It was discovered that there were 872 males and 858 females in the dataset. After exploring the data types of each of the columns, I discovered that both thorax_length and wing_loading were object types, therefore I converted them to a float type in order to enable numerical operations and statistical computations. After changing the object type it was found that there was a null value for thorax_length and wing_loading, this was removed as it wouldn’t help in our ML model, leaving 1730 rows remaining. The class label in this analysis is the sex column as this is a classification problem and our goal is predicting whether or not the fruit fly is female or male using the column characteristics. Therefore, the sex column was converted into binary, to simplify the modeling process, a 1 was assigned to male and 0 for female. Further one hot encoding was used on the population and species column to change the data from a categorical type to numerical, making it easier for numerical algorithms.

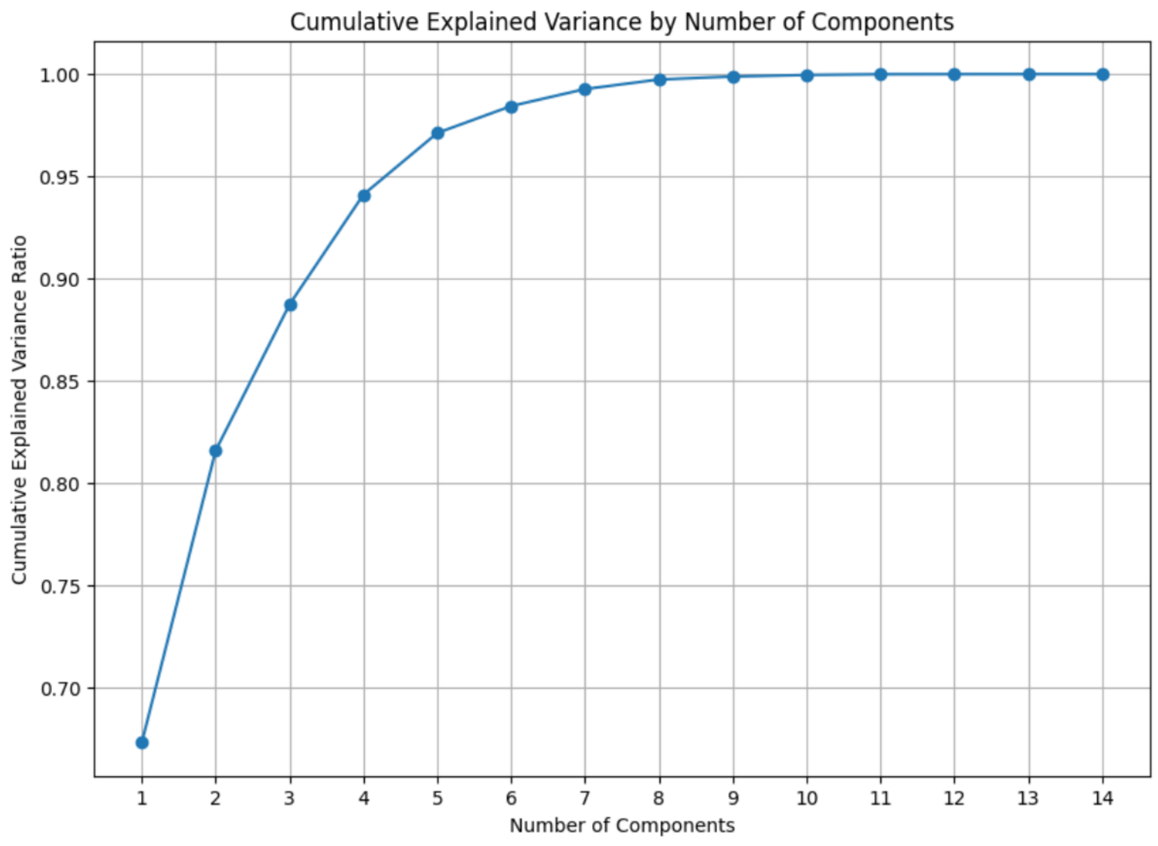
Exploratory Data Analysis

A correlation matrix was used to understand the relationships within the dataset. It was discovered that there was a strong positive correlation with the measurement columns as it was close to 1. Further there was a weak correlation for stuff like temperature, vial, and thorax_length.



	Latitude	Longitude	Temperature	Vial	Thorax_length	l2	l3p	l3d	lpd	l3	w1	w2	w3	wing_loading
Latitude	1.00	-0.99	-0.00	-0.01	0.01	0.01	0.01	-0.03	-0.02	-0.02	-0.00	0.00	0.00	-0.04
Longitude	-0.99	1.00	0.00	0.01	-0.01	-0.01	-0.01	0.03	0.02	0.02	-0.00	-0.00	-0.01	0.04
Temperature	-0.00	0.00	1.00	-0.01	-0.62	-0.74	-0.73	-0.71	-0.73	-0.73	-0.74	-0.76	-0.76	-0.55
Vial	-0.01	0.01	-0.01	1.00	0.01	0.01	0.02	0.01	0.01	0.01	0.02	0.01	0.02	0.01
Thorax_length	0.01	-0.01	-0.62	0.01	1.00	0.84	0.81	0.82	0.83	0.83	0.78	0.86	0.79	0.29
l2	0.01	-0.01	-0.74	0.01	0.84	1.00	0.90	0.97	0.99	0.99	0.93	0.95	0.95	0.74
l3p	0.01	-0.01	-0.73	0.02	0.81	0.90	1.00	0.82	0.89	0.89	0.85	0.91	0.86	0.60
l3d	-0.03	0.03	-0.71	0.01	0.82	0.97	0.82	1.00	0.99	0.99	0.94	0.93	0.94	0.77
lpd	-0.02	0.02	-0.73	0.01	0.83	0.99	0.89	0.99	1.00	1.00	0.95	0.94	0.96	0.77
l3	-0.02	0.02	-0.73	0.01	0.83	0.99	0.89	0.99	1.00	1.00	0.95	0.94	0.96	0.77
w1	-0.00	-0.00	-0.74	0.02	0.78	0.93	0.85	0.94	0.95	0.95	1.00	0.92	0.98	0.74
w2	0.00	-0.00	-0.76	0.01	0.86	0.95	0.91	0.93	0.94	0.94	0.92	1.00	0.91	0.64
w3	0.00	-0.01	-0.76	0.02	0.79	0.95	0.86	0.94	0.96	0.96	0.98	0.91	1.00	0.74
wing_loading	-0.04	0.04	-0.55	0.01	0.29	0.74	0.60	0.77	0.77	0.77	0.74	0.64	0.74	1.00

Due to the high correlation among a lot of the features, it may be best to try principal component analysis (PCA) as this reduces the number of inputs and noise. This is a technique that transforms data into a smaller set of principal components while retaining as much variation as possible. In order to do this, the data is first normalized, so that it takes a value between 0 and 1 putting all the features on the same scale. I get 5 components for the PCA seen in the graph below. The number of components was stopped after hitting the 95% variance mark, which this is telling us that 95% of the data can be explained with 5 components.



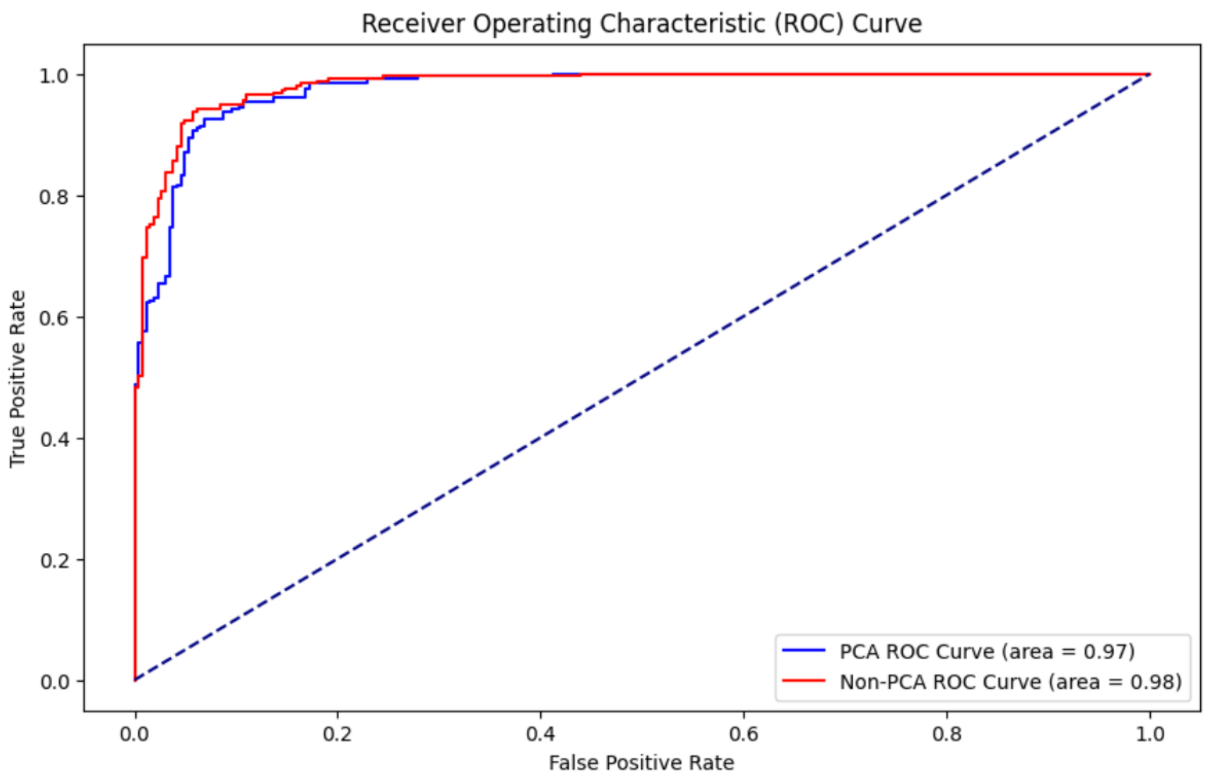
Therefore, after the PCA was performed the data was divided into PCA data and the regular cleaned data. This was to determine which dataset was more accurate and to compare and see which models had a better accuracy with PCA or without PCA depending on their complexity. A 10-fold cross validation will be used to effectively compare the accuracy of the various models. It was chosen, due to its effectiveness and ability to balance computational speed and efficiency.

Linear Regression

As a baseline model, linear regression will be used due to its simplicity and interpretability. The Non-PCA data achieved an accuracy of 93.8% whereas the PCA data achieved an accuracy of 92.2%. The non-PCA data had the exact same recall and precision for both female and male. Whereas the PCA data the precision was higher for female and lower for female and vice versa for recall.

To select the best model out of the different techniques we must have some sort of analysis technique to compare. Therefore using 10-fold cross validation we will use the cross-validation test loss to compare and choose the best model. The cross-validation train loss will be used to understand any overfitting or underfitting that is happening. This process helps provide a rough estimate of the model's performance by averaging the results across 10 trials. A cross validation test loss will be analyzed for both the PCA and non-PCA datasets. A log loss will be used, and it is a metric for classification models. It measures the accuracy of probability predictions. A lower log loss indicates better performance.

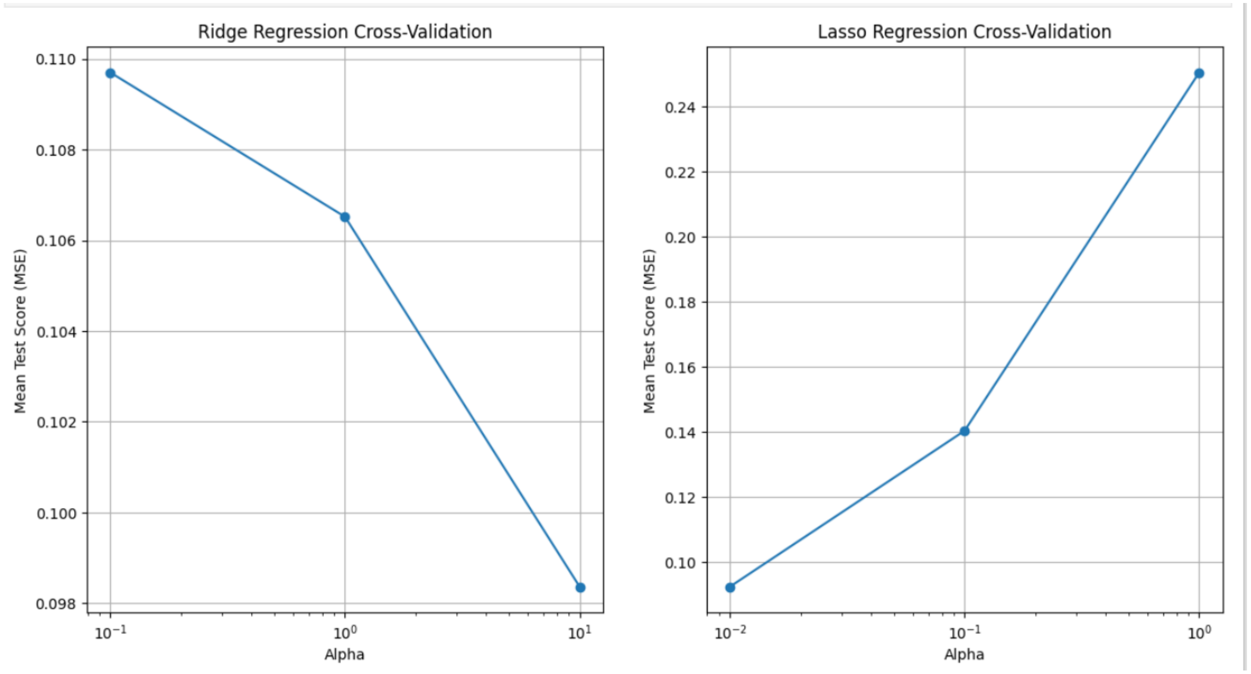
The PCA data achieved a cross validation test loss of 0.216 and a cross validation train loss of 0.201. The PCA test loss is slightly higher than train loss, which suggests a good fit without significant overfitting. To understand the tradeoff between sensitivity (true positive rate) and specificity (False positive rate) an ROC curve will show with different thresholds. We can see in figure (x) that both the PCA and non-PCA dataset are close to the top left corner of the plot, which indicates a high true positive rate and a low false positive rate. The non-PCA model (0.98) slightly outperforms the PCA model (0.97) which tells us that keeping the original features without reducing to number of components is an advantage when determining a fruit flies sex.



To enhance the performance of the model, I used Ridge and Lasso regression to incorporate regularization. Regularization is great at preventing overfitting and controlling the complexity of a model. Which can lead to better results on unseen data.

I tuned the hyperparameter alpha, which controls the strength of the regularization. Using GridSearchCV (performs hyperparameter tuning) with a 10-fold cross validation, I searched for an optimal alpha value across predefined ranges.

For Ridge regression the best value was 10.0, which resulted in a mean cross validation – mean squared error (MSE) of 0.0983. For Lasso regression, 0.01 was the optimal alpha value. It had a cross validation MSE of approximately 0.0925. Both regularization techniques effectively improve the model's performance, indicated by these results. The optimal alpha values minimize the cross-validation MSE, therefore below we can visualize.

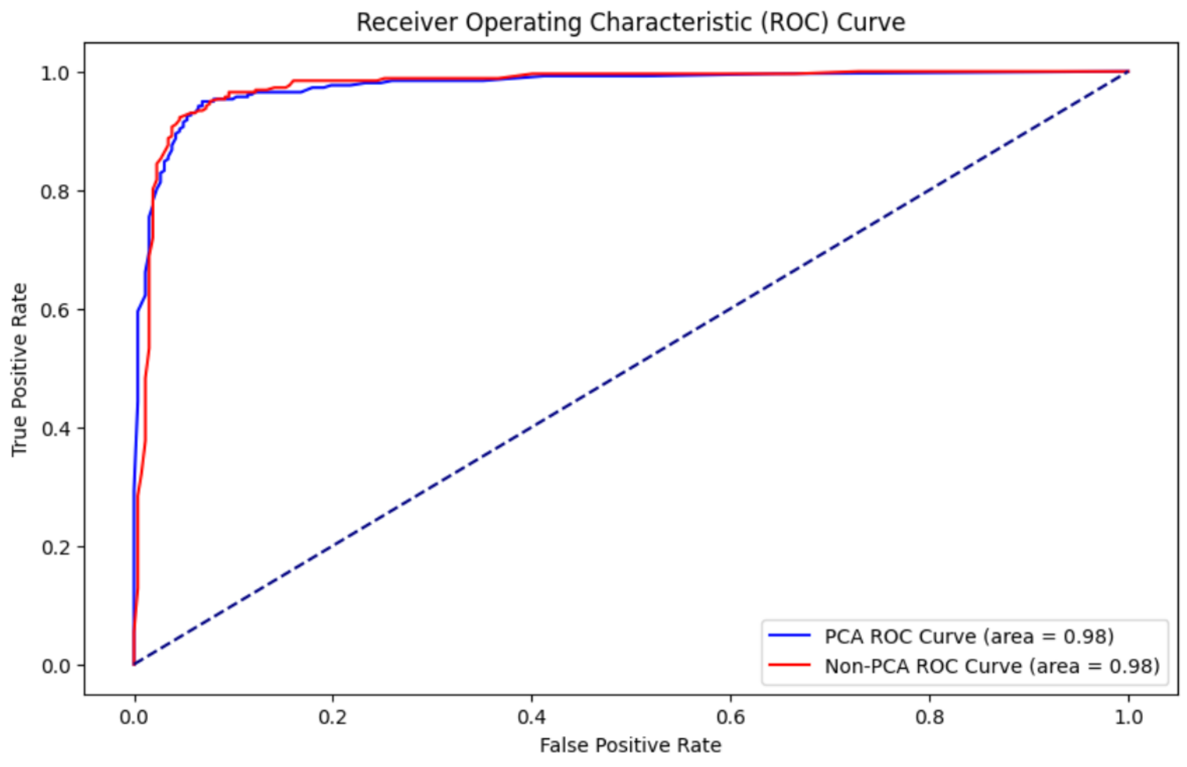


This plot shows the cross validation MSE for different alpha values for Ridge and Lasso regression. For Ridge regression the MSE decreases with increasing alpha, which indicates that higher regularization helps improve the model performance. For Lasso regression the MSE increases with increasing alpha indicating lower regularization is optimal. This shows how regularization helps to improve a model's performance.

Random Forest

Random forest will be the next model used as it's a very versatile, robust algorithm that handles data and overfitting well. The Non-PCA data achieved an accuracy of 93.2% whereas the PCA data achieved an accuracy of 93.8%. The slight increase in overall accuracy for PCA suggests that reducing the dimensionality helped in this case, which might be from reducing the noise or any distractions to the model. This benefits a random forest model directly benefits from a noise reduced, simplified space. However, the differences are not huge, meaning they are both fit for this classification task.

A 10-fold cross validation will again be used to help choose the best model. The PCA data achieved a cross validation test loss of 0.241 and a cross validation train loss 0f 0.0449. The low train log loss indicated that the training data fits the model very well and has effectively found trends in the data. The test log loss is also still low, suggesting that the model generalizes reasonably well to unseen data. The higher test loss is expected as the model generally performs better on data they were trained on. However, the large difference could suggest overfitting in the model and techniques will be used to adjust the parameters. The non-PCA data achieved a cross validation test loss of 0.221 and a cross validation train loss of 0.0541. To understand the tradeoff between sensitivity (true positive rate) and specificity (False positive rate) an ROC curve will show with different thresholds again. We can see below that both curves are very close to the top left corner of the plot indicating a high true positive rate and low false positive rate. The area under both the blue and red line is almost identical indicating that both are well fit for the classification of the two classes.



To optimize the performance of this model hyperparameter tuning can be used. Specifically, I used RandomizedSearchCV for this. It searches across a wide variety by sampling combinations of parameter and evaluating their performance using cross validation. I have put the definition and the variable name of a couple of these hyper tuning parameters.

- **Number of Trees (n_estimators)**: Varying the number of trees helps balance the model's complexity and computation time.
- **Maximum Depth (max_depth)**: Limiting the depth of each tree prevents overfitting by controlling the model's complexity.
- **Minimum Samples Split (min_samples_split)**: This parameter determines the minimum number of samples required to split an internal node.

- **Minimum Samples Leaf (min_samples_leaf)**: Ensures that leaf nodes have a minimum number of samples, which helps prevent overfitting.
- **Bootstrap Method (bootstrap)**: Using bootstrap samples to build trees can improve model robustness and accuracy.

50 iterations with a 10 cross validation was done to ensure a good evaluation of the combinations. The best combination was:

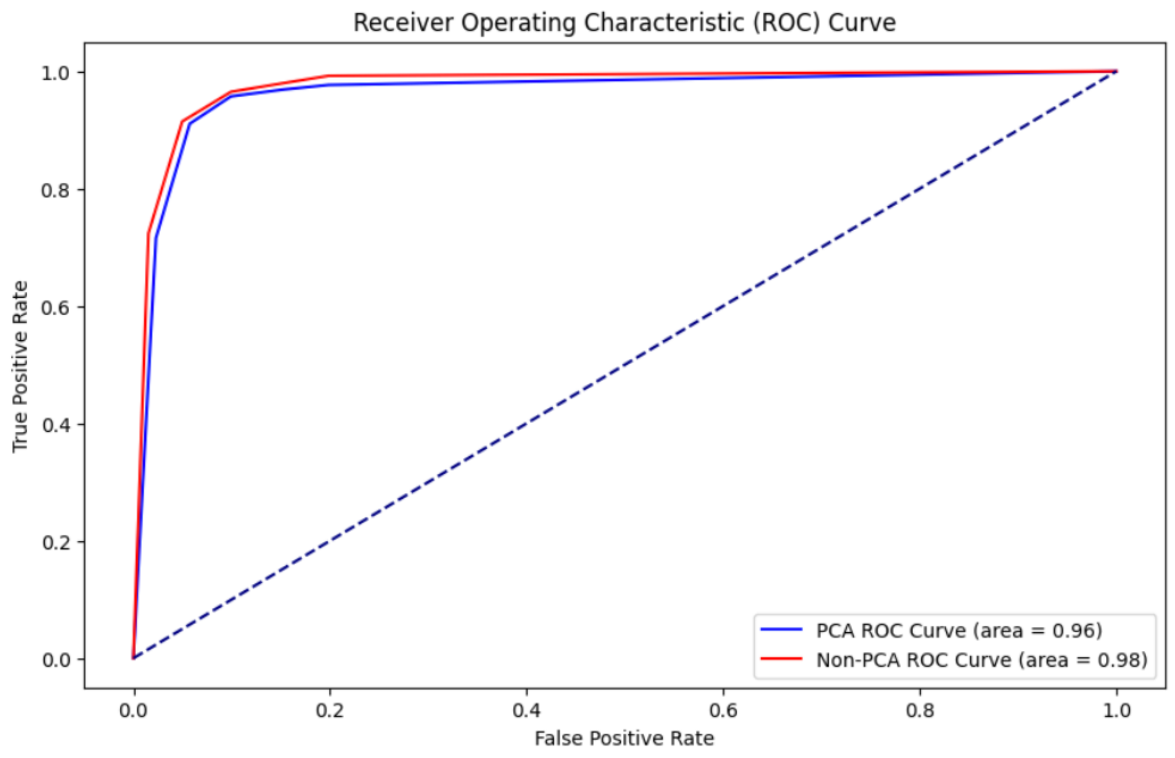
n_estimators: 100
min_samples_split: 2
min_samples_leaf: 2
max_depth: 10
bootstrap: True

These hyperparameters performed a cross validation accuracy of 93.9% which is slightly better than our non-hyperparameter tuning model. Which demonstrates why it can be very important to select hyperparameters to improve model performance.

K Nearest Neighbor

K Nearest Neighbor will be the next model used and this is another good model for classification where its easily understandable but affective. The Non-PCA data achieved an accuracy of 93.2% whereas the PCA data achieved an accuracy of 92.8%. This is also interesting as the non-PCA data achieved a better accuracy. This slight decrease in accuracy for the PCA data suggests that reducing the dimensionality does not improve the performance of the model very much. This is most likely due to the KNN model being a simple algorithm that relies heavily on distance metrics. Further reducing some of the dimensions might have led to some loss of information that was crucial for accurate classification.

To further understand the tradeoff between sensitivity (true positive rate) and specificity (false positive rate), an ROC curve was plotted with different thresholds. The ROC curves for both PCA and non-PCA data are shown below. Both curves are very close to the top left corner of the plot, indicating a high true positive rate and low false positive rate. The area under the curve (AUC) for both models is nearly identical, suggesting that both models are well-suited for the classification task at hand.



The low train log loss for both PCA and non-PCA data indicates that the training data fits the model well, capturing a lot of the trends. The test log loss, although it was higher than the train loss remains low, which suggests that the model performs reasonably well to unseen data. The gap between the train and test log losses does suggest there is some degree of overfitting going on, which will be adjusted through hyperparameter tuning.

Hyperparameter Tuning can be done for K-Nearest Neighbors, and to optimize the performance GridSearchCV can be used. It searches over a specified parameter grid to find what the optimal combination of hyperparameters are that maximize accuracy. Before diving into the results, let's talk about the key hyperparameter being tuned.

- **Number of Neighbors (n_neighbors)**: This parameter determines the number of nearest neighbors considered for making predictions.
- **Weights (weights)**: This parameter specifies whether all neighbors contribute equally (**uniform**) or if closer neighbors have more influence (**distance**).
- **Distance Metric (metric)**: The distance metric used to calculate distances between points, with options including **'euclidean'** and **'manhattan'**.

Again, using a cross validation approach, I searched for the best combination of parameters and found out through the result of my code that:

- **n_neighbors**: 9

- **weights**: 'distance'
- **metric**: 'euclidean'

These parameters resulted in a cross validation of approximately 92.7%. Therefore, again we can see that hyperparameter tuning has increased the accuracy of predicting whether a fruit fly is a male or a female.

Model Comparison/Conclusion

The goal of this analysis was to help predict the sex of fruit flies on various characteristics given, using different machine learning models. Three models were explored, Linear Regression, Random Forest, and K-Nearest Neighbors (KNN). The models were compared on their performance on PCA data as well as non-PCA data.

Linear Regression with Ridge and Lasso regularization provided accuracies of 93.8% and 92% for non-PCA and PCA data. Regularization improved the model’s performance by preventing overfitting and controlling the complexity.

Random Forest achieved accuracies of 93.2% and 93.8% for non-PCA and PCA data. Hyperparameter tuning using RandomizedSearchCV further improved its accuracy to 93.9%, making it the model with the highest accuracy.

K-Nearest Neighbors (KNN) achieved accuracies of 93.2% and 92.8% for non-PCA and PCA data. Hyperparameter tuning with GridSearchCV identified the best parameters, resulting in an accuracy of 92.7%.

Overall, the Random Forest model with hyperparameter tuning was the most accurate out of the three machine learning models for predicting the sex of fruit flies. This also shows the power of hyperparameter tuning and how it can be used to improve accuracy. Future work could involve looking at scaling and normalization techniques that could further enhance model performance.

GitHub code

https://github.com/Jack-Wallace10/COMP4702

(Above is the code for my assignment) – Has more graphs + Analysis if wondering where numbers come from.