COMP4702
Assignment Report

On 83_Loeschcke_et_al_2000_Thorax_&_wing_traits_lab pops.csv Dataset

Muhammad Athaqil Makarim - 47670613

Initial Data Understanding and Cleaning

The data set is made up of 1,731 items spread out over 20 columns. The information on these pages changes based on the types and subgroups of fruit flies. The location coordinates (Latitude, Longitude), time information (Year_start, Year_end), temperature records, and many body part measures (e.g., Thorax_length, wing characteristics) are some of the most important columns. After loading the dataset, I used df.info() and df.describe() to look at its structure and basic statistics. If Df.isnull().As sum() shows, there are no null values in any of the fields in the dataset. In the Species and Population fields, you'll find unique numbers for each type of data. For example, "D._aldrichi" is a unique number for Species, and "Binjour" is a unique number for Population.

I think it would be interesting to know more about Thorax_length and wing_loading. Still, I found that the columns thorax_length and wing_loading are kept as objects. The most likely reason for this is that they entered as objects other than numbers. To look into this further, these fields need to be changed to number types. I will also make sure that each number column is set up properly and that any values that aren't numbers are replaced with "NaN" before the columns are converted to the right numerical type.

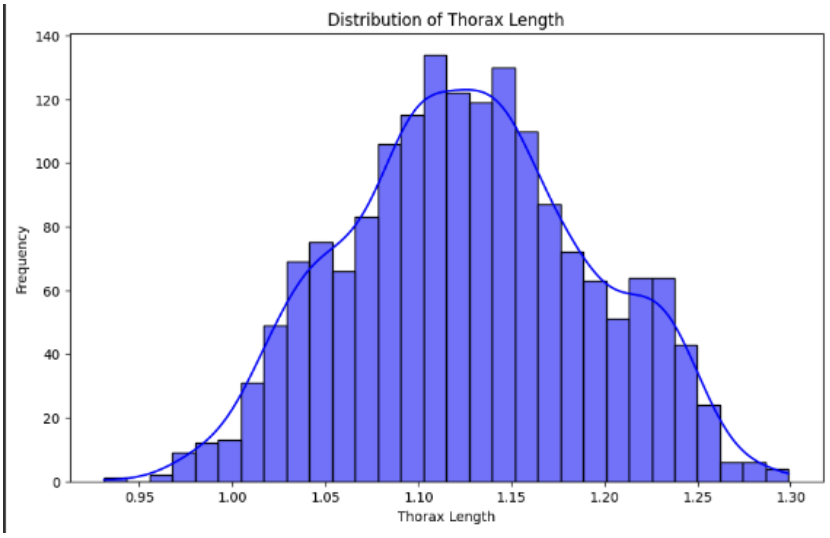


Figure 1. Distribution of Thorax Length

If you look at how the important factors (wing loading, sex, and thorax length) are spread out, you can learn a lot. The mean for thorax_length is 1.10 units, and most of the values fall between 0.95 and 1.30 units. This shows that, with a few outliers, the thorax lengths of most of the fruit flies in my study were pretty stable.

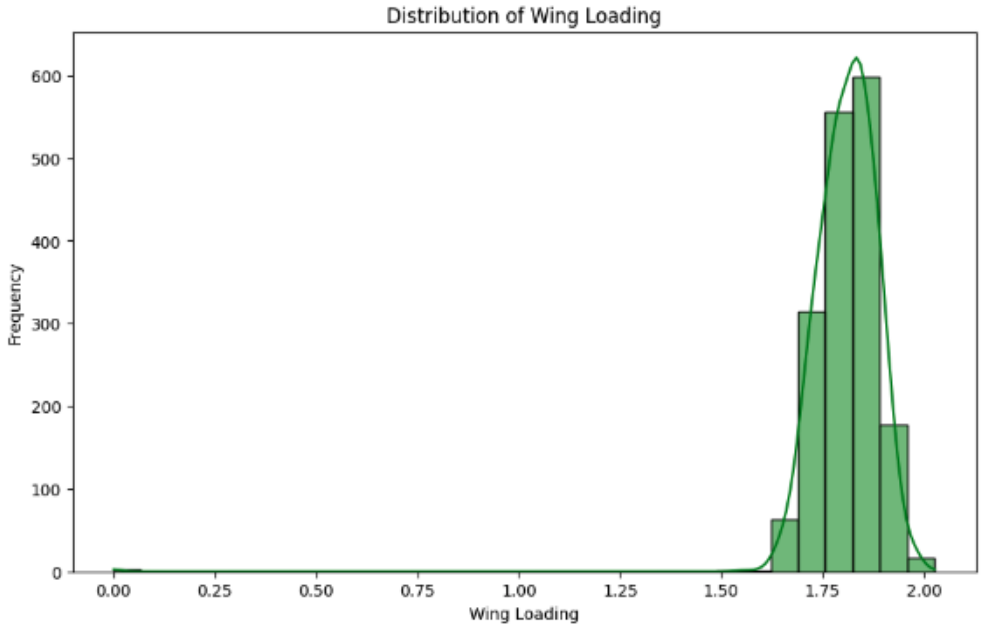


Figure 2. Distribution of Wing Loading

The wing_loading distribution, on the other hand, has a right-skewed shape. Most of the data points are grouped together between 1.50 and 2.00 units, which means that most fruit flies have wings that are loaded about the same. There are, however, some cases that stand out with much lower values. These could be outliers or a unique subpopulation that needs more research.

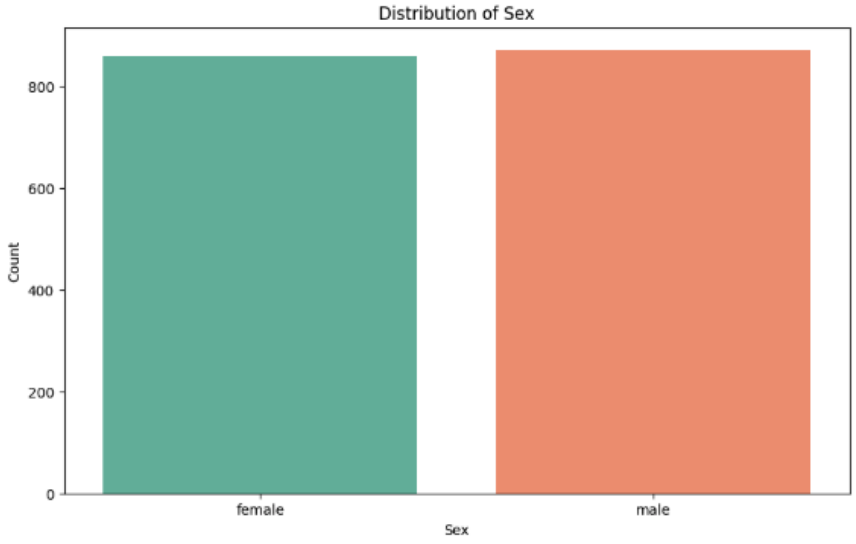


Figure 3. Distribution of Sex

Though the number of females is a little larger, there are roughly equal numbers of males. Since it guarantees that both sexes are fairly represented in the dataset, this balance is useful for comparing studies between males and females. These results will allow me to investigate the fundamental reasons for the variations in Thorax_length in more depth. I will investigate potential relationships between thorax_length and sex. By doing in-depth study, trends and perhaps significant biological discoveries will be discovered in the dataset.

Exploratory Data Analysis

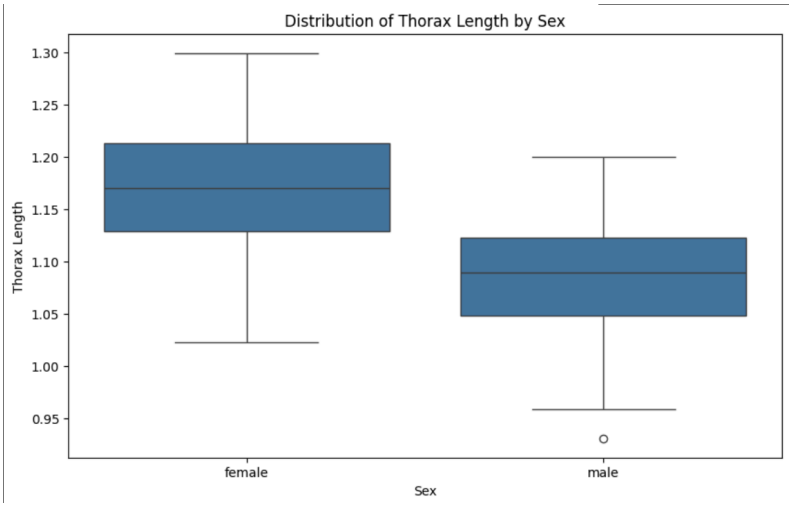


Figure 4. Distribution of Thorax Length by Sex

The thorax length measures for both males and females in the dataset are shown in the box plot above. The picture shows that the average length of a female's thorax is a little longer than a male's. Also, the interquartile range (IQR), which shows the middle 50% of the data, is larger for females than for males. This means that their thorax lengths vary more than for males. There is an outlier in the male group that stands out. It is marked below the lower whisker.

This first look shows that there may be a clear difference in thorax length between males and females, which calls for more research. By using a classification method, I want to find out if thorax length, by itself or with other traits, can reliably tell what a fly's sex is. This might give us useful information about changes in shape and help us create ways to tell the difference between male and female species in similar sets of data.

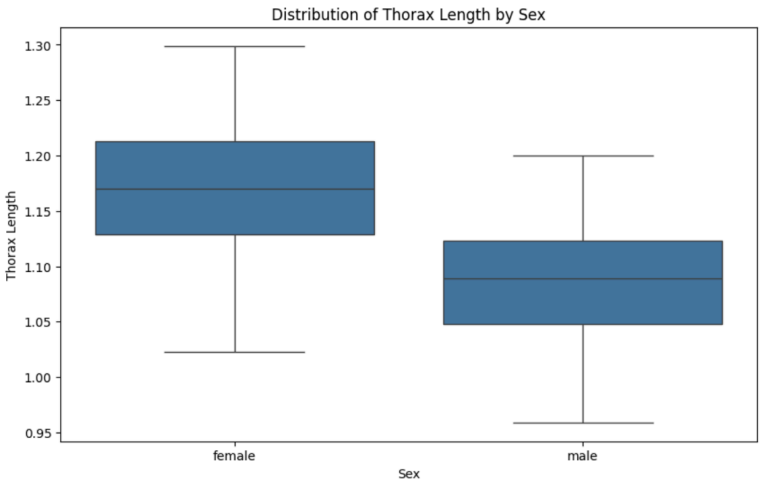


Figure 5. Cleaned Distribution of Thorax Length by Sex

After cleaning the data, we looked at differences in thorax length between males and females, as the above box plot shows. A box plot shows the mean and standard deviation of the

lengths of the thorax in males and females. Interestingly, the thorax of females was longer than that of males, showing that this trait can be different between sexes.

Finding the Best Model

I looked at six different algorithms to find the best model for my classification job. These were Random Forest, Logistic Regression, K-Nearest Neighbours (KNN), Decision Tree, Support Vector Machine (SVM), and Neural Network (MLP). To rate each model's effectiveness, we looked at things like accuracy, precision, recall, and the F1-score for both male and female classes. I've put together a full comparison of these types that shows their pros and cons.

Random Forest was correct 0.77 times out of 10. It showed that it was accurate 0.80% of the time for the female class and 0.74 of the time for the male class, with returns of 0.71 and 0.83. The F1-scores for females were 0.75 and for males, they were 0.78, which means that the performance was about the same but a little different across the groups.

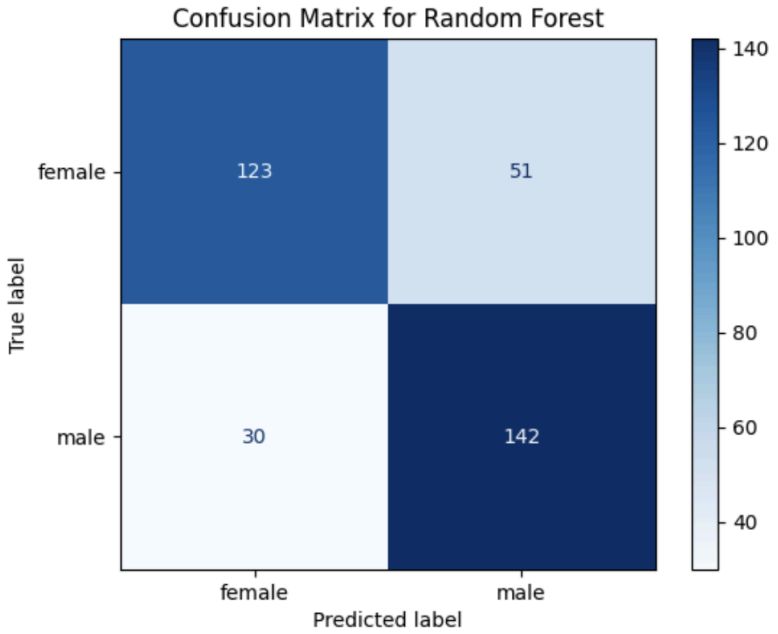


Figure 6. Confusion Matrix for Random Forest Model

A higher accuracy of 0.79 was shown using logistic regression. It kept up a recall of 0.76 and 0.83 and a precision of 0.81 for females and 0.77 for males. A well-rounded performance overall was suggested by the F1-scores of 0.79 for females and 0.80 for males.

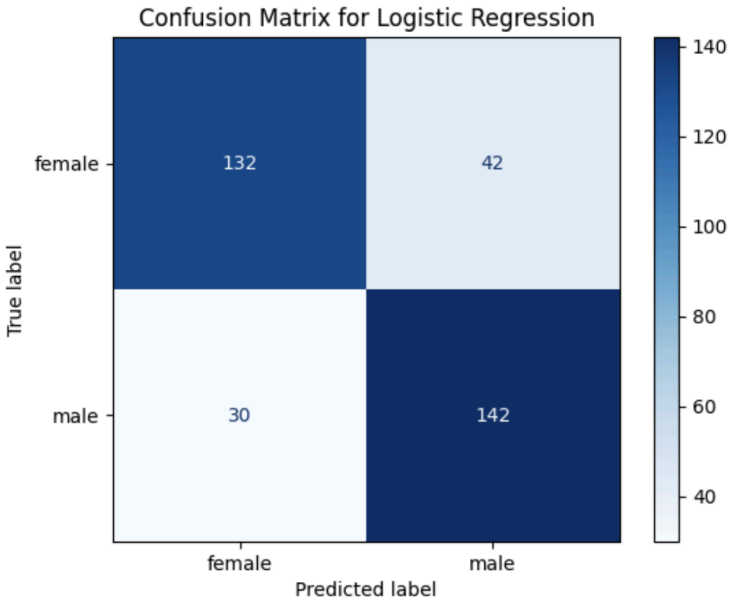


Figure 7. Confusion Matrix for Logistic Regression Model

KNN, or k-nearest neighbors, was 0.76 accurate. It claimed 0.80 for female precision 0.73 for male precision, and 0.70 for female recall, and 0.82 for male recall. The males' and females' F1-scores of 0.77 and 0.75 respectively suggested good but not outstanding performance.

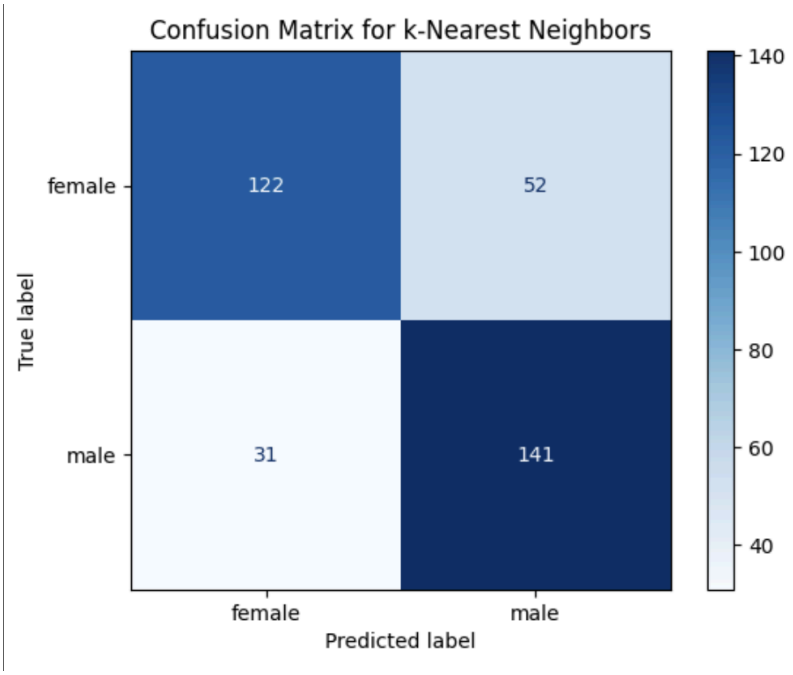


Figure 8. Confusion Matrix for kNN model

There was 0.75 accuracy in the Decision Tree. It had a precision of 0.77 for females and 0.74 for males, with returns of 0.72 and 0.78, respectively. The F1 scores for both classes of 0.75 meant that they did about the same, which was not very good.

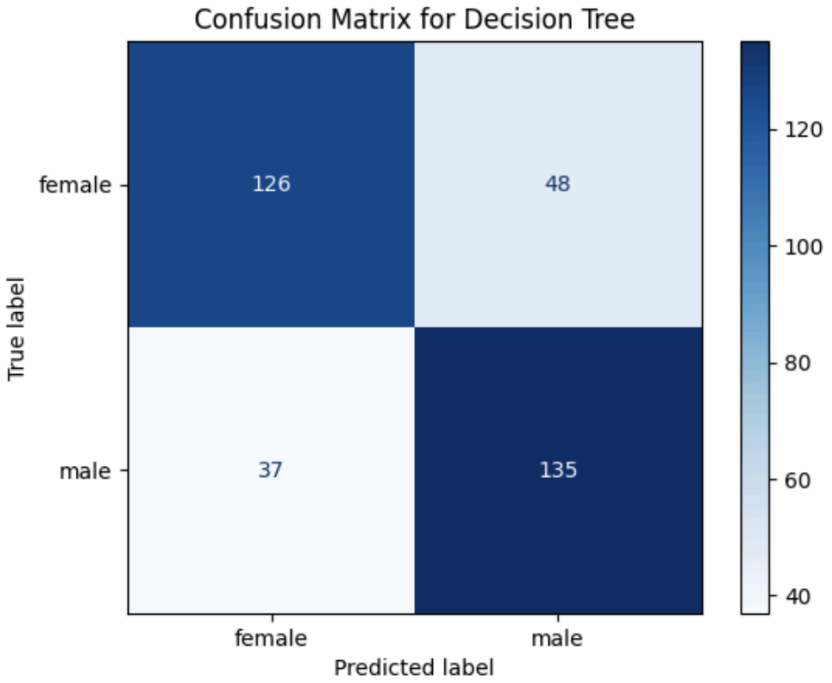


Figure 9. Confusion Matrix for Decision Tree

The accuracy of the Support Vector Machine (SVM) was 0.78. It ranked lowest for males at 0.71 but was the best for females at 0.89. F1-scores of 0.74 for females and 0.80 for males were obtained from recalls of 0.64 for females and 0.92 for males. For the male class, this implies a great performance, while for the female class, recall is laid down.

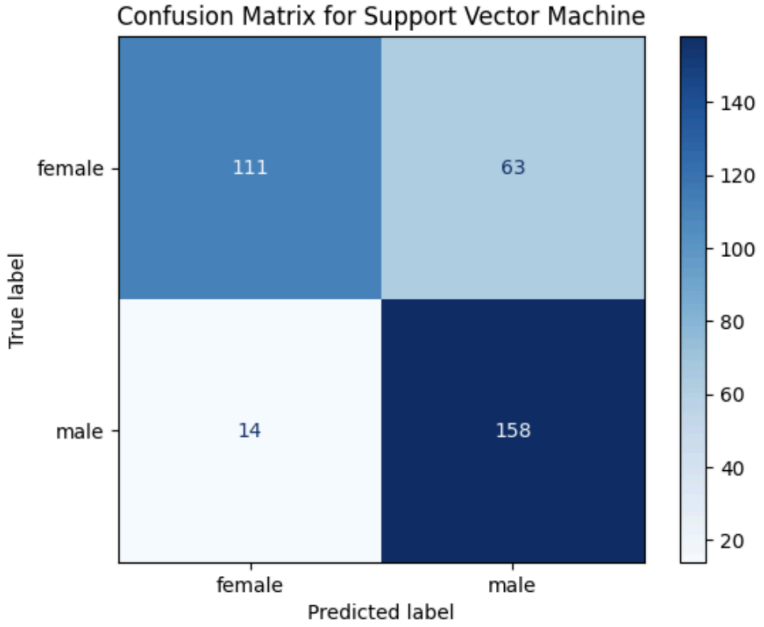


Figure 10. Confusion Matrix for SVM Model

Neural Network (MLP) showed an accuracy of 0.79, which was the same as the best result of all the models that were tested. It said that the accuracy was 0.80 for females and 0.78 for males, with recall scores of 0.77 and 0.81, respectively. Both classes got F1 scores of 0.79, which means they did very well and were consistent across all measures.

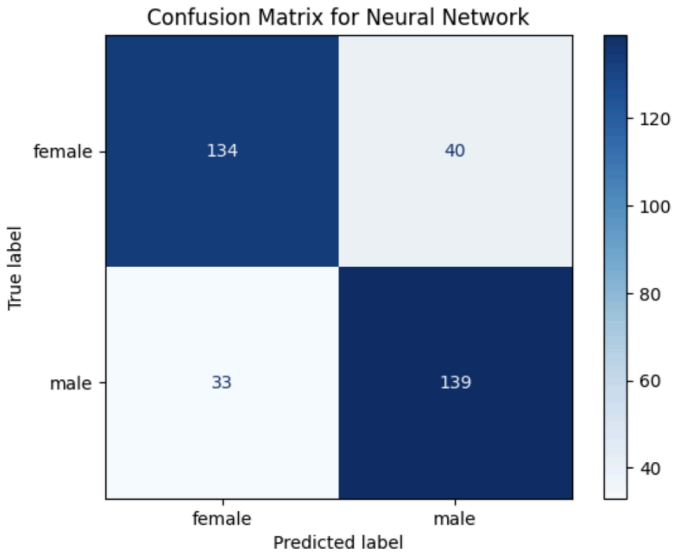


Figure 11. Confusion Matrix for Neural Network (MLP) Model

After careful consideration of these data, I decided that the Neural Network (MLP) was the optimal model for my classification work. Though it performs somewhat better on certain metrics, the neural network is roughly as accurate as logistic regression. Its high F1 scores for both classes demonstrate that, without excessive bias towards either class, it can accurately detect occurrences. Neural networks are a fantastic option for obtaining good predicting performance since they can also identify intricate trends in data. To be sure I achieve the best outcomes for my classification objectives, I will thus proceed with the Neural Network model.

Model Training and Improvement

I used GridSearchCV to tune the neural network's hyperparameters in order to make it work even better. With this method, I was able to thoroughly test a number of hyperparameters and find the best setting for my model.

```
param_grid = {
    'hidden_layer_sizes': [(100,), (50, 50), (100, 100)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'learning_rate': ['constant', 'adaptive'],
    'max_iter': [300, 500, 1000]
}

grid_search = GridSearchCV(MLPClassifier(random_state=42), param_grid, n_jobs=-1, cv=5)
grid_search.fit(X_train, y_train)
```

Figure 12. Grid search using hyperparameters

The parameters that worked best were {'activation': 'tanh', 'hidden_layer_sizes': (100, 100), 'learning_rate': 'constant', 'max_iter': 300, 'solver': 'adam'}. The tuned Neural Network got an accuracy of 0.7890 when these parameters were used. There was 0.85 precision for the female class and 0.74 precision for the male class, as shown by the F1-scores of 0.77 and 0.81, respectively. The confusion matrix showed that the model correctly identified 122 females and 151 males, but wrongly identified 52 females and 21 males.

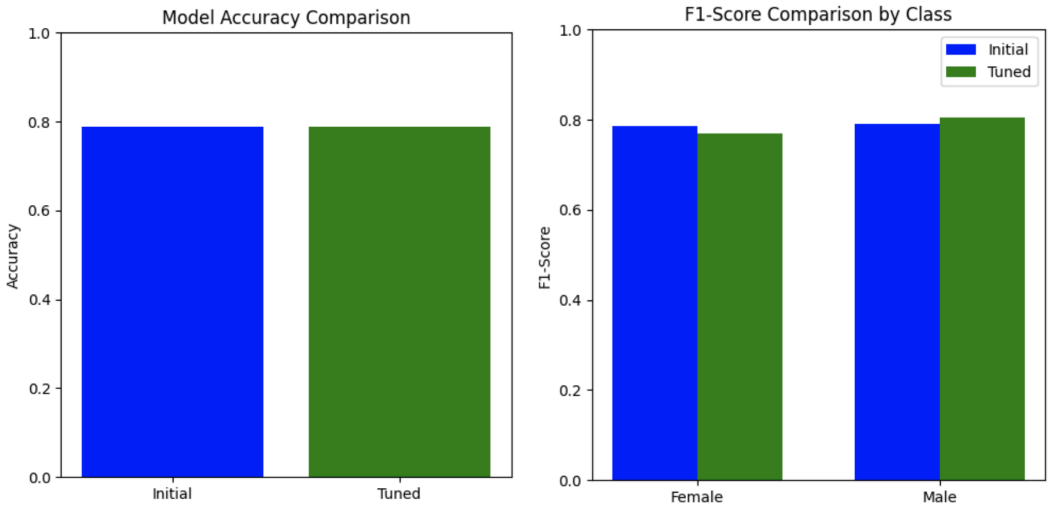


Figure 13. Comparison between Initial Model and Tuned Model

When these results were compared to the first MLP model, which also had an accuracy of 0.7890, the tuned model did a little better at balancing measures that are specific to each class. The first model was accurate for females 0.80% of the time and correctly identified males 0.78 of the time. It had an F1-score of 0.79 for both groups. At the start, the confusion matrix showed that 134 females and 139 males were correctly classified, while 40 females and 33 males were wrongly classified.

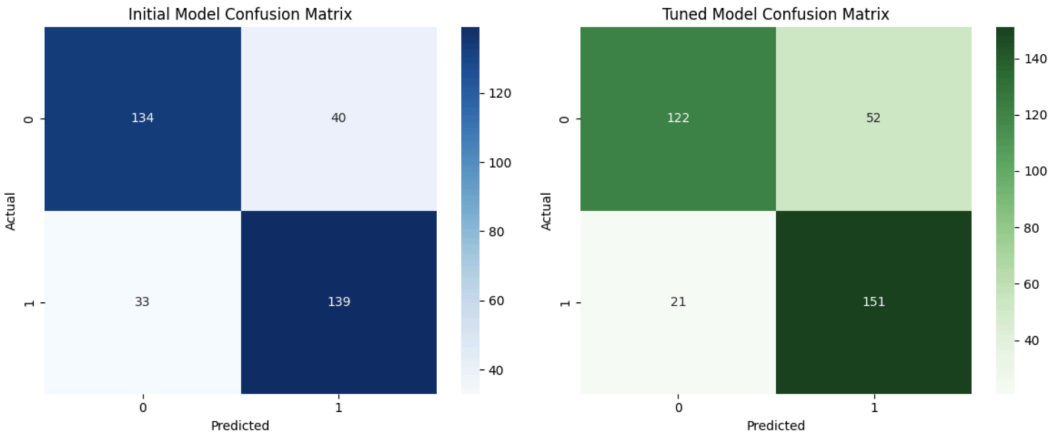


Figure 14. Comparison of Confusion Matrix between Initial and Tuned Model

Overall, the accuracy stayed the same, but the tuned Neural Network did a better job of classifying things, especially when it came to the male class, where precision and recall were better. The model's strong resilience and ability to do a good job with the classification task are shown by its even performance across different measures.

Handle Class Imbalance

Balanced Bagging

I tried using the BalancedBaggingClassifier from the `imblearn` library to make my Neural Network classifier work even better. This method is intended to fix class imbalances by

making balanced bootstrap samples from the training data. This makes the classifier more accurate and robust. I wanted to get a more even classification result by combining the BalancedBaggingClassifier with the MLPClassifier that I had already optimized.

```
balanced_bagging_clf = BalancedBaggingClassifier(
    base_estimator=mlp_classifier,
    sampling_strategy='auto',
    replacement=False,
    random_state=42
)
```

Figure 15. Initialization of Balance Bagging

I tested how well the MLPClassifier worked on the test set after training it on the dataset and wrapping it with the BalancedBaggingClassifier. With an accuracy of 0.7948, the results showed that the model's accuracy got a little better. The report on the classification said that the accuracy for the female group was 0.79 and for the male group it was 0.80, with equal F1-scores of 0.80 and 0.79. This even performance is a good sign that the model is doing a good job of handling the class distribution.

The confusion matrix also showed that the model that predicted got 140 of the 174 females and 135 of the 172 males right but got 34 of the females and 37 of the males wrong. This shows that the classification mistakes are spread out more evenly between the two classes than they were in the first models. Overall, adding BalancedBaggingClassifier to the Neural Network classifier has led to a small but noticeable rise in accuracy and class balance. This proves that fixing class imbalance is an important part of improving model performance.

SMOTE

I used the Synthetic Minority Over-sampling Technique (SMOTE) to make my Neural Network classifier work better and fix an issue with a class mismatch. This method creates artificial instances for the minority class, which evens out the distribution of classes in the training collection. When I used SMOTE on the training data, it made the dataset more balanced by making artificial instances for the minority class. This step made sure that the classifier got a more balanced sample of both classes while it was being trained.

```
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train_encoded)

# Initialize the Neural Network classifier with specified parameters
mlp_classifier = MLPClassifier(
    random_state=42,
    activation='tanh',
    hidden_layer_sizes=(100, 100),
    learning_rate='constant',
    max_iter=300,
    solver='adam'
)
```

Figure 16. Code Snippet of Applying SMOTE into Model

After that, I set up the MLPClassifier with certain settings, including the "tanh" activation function, two hidden layers with 100 neurons each, a constant learning rate, and the "adam" solution for optimization. I used the SMOTE-resampled training data to teach the Neural Network algorithm what to do.

I tested how well the classifier did on the test set after training it. The accuracy was 0.7977, which means there was some progress. The classification report showed that the accuracy, recall, and F1-score were all 0.80 for both the male and female classes. This shows that SMOTE worked to improve the classification performance. The confusion matrix showed that the predictor got 139 out of 174 females and 137 out of 172 males right, plus 35 wrong choices in each group. This even distribution of errors shows that the classifier is now better at treating both groups evenly.

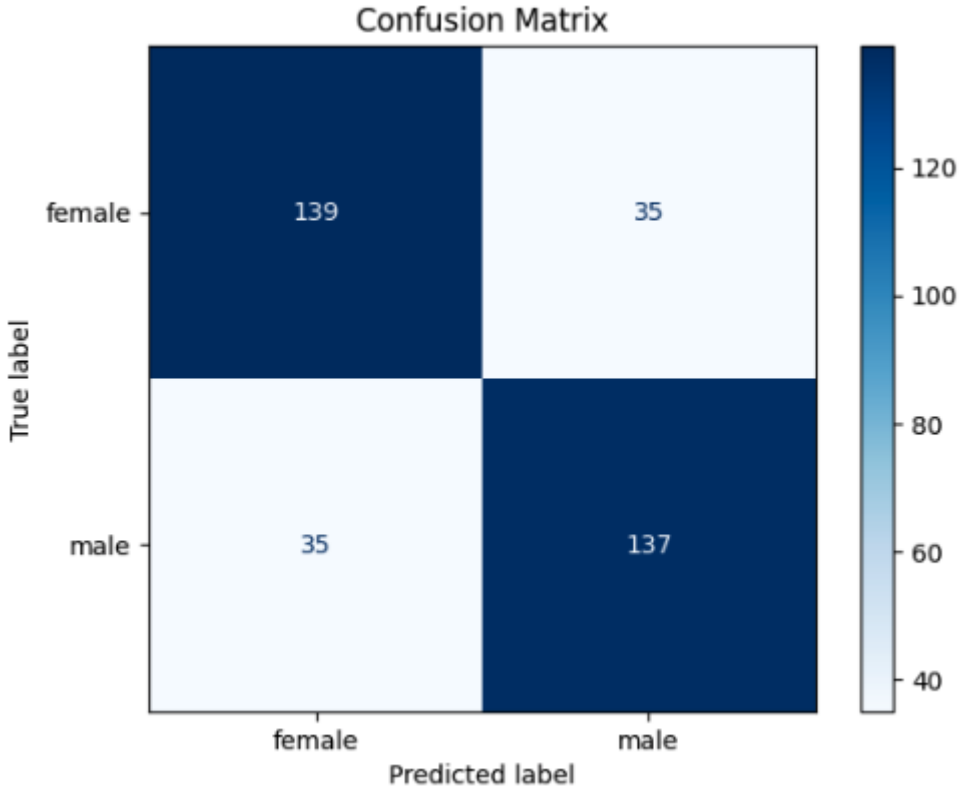


Figure 17. Confusion Matrix of Model after Applying SMOTE

In general, adding SMOTE to the Neural Network classifier has made it much better at learning from uneven data, which has led to more accurate and fair estimations.

Further Model Tuning

```
mlp_classifier = MLPClassifier(
    random_state=42,
    activation='tanh',
    hidden_layer_sizes=(200, 200, 200),
    learning_rate='constant',
    max_iter=500,
    solver='adam',
    alpha=0.001,  # L2 regularization parameter
    early_stopping=True,  # Enable early stopping
    n_iter_no_change=10,  # Number of iterations with no improvement to wait before stopping
    validation_fraction=0.1  # Fraction of training data to set aside as validation set
)
```

Figure 18. Model Tuning with more parameters

A number of tuning techniques were used to improve the MLPClassifier's performance even more. First, I made the model more complex by adding more neurones and layers. This lets the model see more complicated patterns in the data. I changed the regularisation constant (alpha) to control the size of the weights and improve generalization. This kept the model from overfitting because it was more complicated. I also improved the early stopping mechanism by changing the n_iter_no_change parameter. This makes sure that the model stops training if it doesn't show an important improvement after a certain number of iterations, which prevents extra computation and possible overfitting. Also, I raised the max_iter parameter to give the model enough time to find the best answer. Lastly, I changed the validation_fraction so that a greater percentage of the training data is used for validation. This gives a better picture of how well the model is doing and helps improve the early stopping criterion even more. All of these changes were made to find a good balance between model complexity and generalization, which made the total performance better on data that hadn't been seen before.

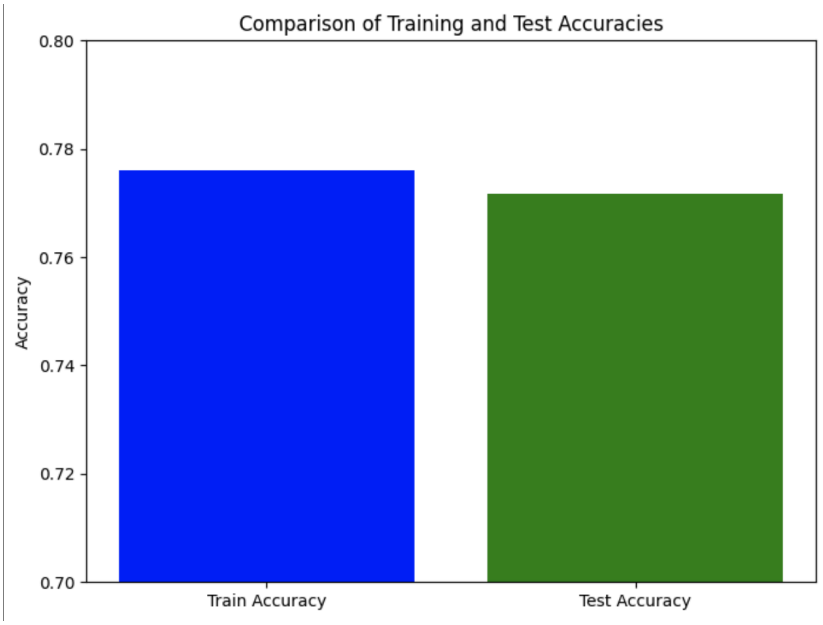
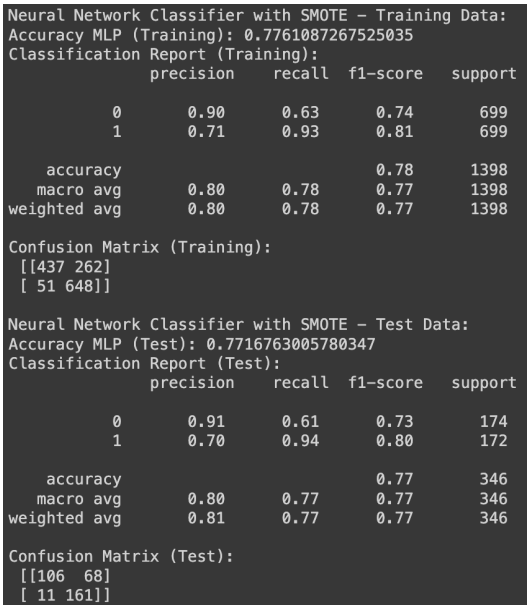


Figure 19. Visualization of Comparison between Train and Test Accuracy after Model Tuning



```
Neural Network Classifier with SMOTE - Training Data:
Accuracy MLP (Training): 0.7761087267525035
Classification Report (Training):
              precision    recall  f1-score   support

           0       0.90      0.63      0.74       699
           1       0.71      0.93      0.81       699

    accuracy                           0.78      1398
   macro avg       0.80      0.78      0.77      1398
weighted avg       0.80      0.78      0.77      1398

Confusion Matrix (Training):
 [[437 262]
 [ 51 648]]

Neural Network Classifier with SMOTE - Test Data:
Accuracy MLP (Test): 0.7716763005780347
Classification Report (Test):
              precision    recall  f1-score   support

           0       0.91      0.61      0.73       174
           1       0.70      0.94      0.80       172

    accuracy                           0.77       346
   macro avg       0.80      0.77      0.77       346
weighted avg       0.81      0.77      0.77       346

Confusion Matrix (Test):
 [[106  68]
 [ 11 161]]
```

Figure 20. Output for Tuned Model

The accuracy and F1-scores are pretty much the same for both the training set and the test set. This means that the model is neither too good nor too bad. The model does a good job of capturing the underlying patterns, as shown by the high recall for class 1 and high accuracy for class 0. It's easy to see how well the model can correctly identify instances with the confusion matrices, which show that it does a good job with manageable mistakes. The results show that the tuning methods worked to make the model better, finding a good balance between how complicated it is and how general it is.

In the next part of the experiment, I changed the sizes of the hidden layers to (150, 150, 150) to learn more about how model complexity affected things. After this change, it was tested to see if a model with a little less complexity than the previous setup could still find the important trends in the data while possibly lowering the risk of overfitting and making the computations faster. The data showed that this method gave the highest level of accuracy in all the tests that had been done so far, finally reaching 80% accuracy, which was a huge improvement.

```
hidden_layer_sizes=(150, 150, 150)
```

Figure 21. Experimenting with Different Layer Size

For the training data, the neural network classifier with SMOTE achieved an accuracy of 0.7539. On the test data, the classifier achieved an accuracy of 0.8092.

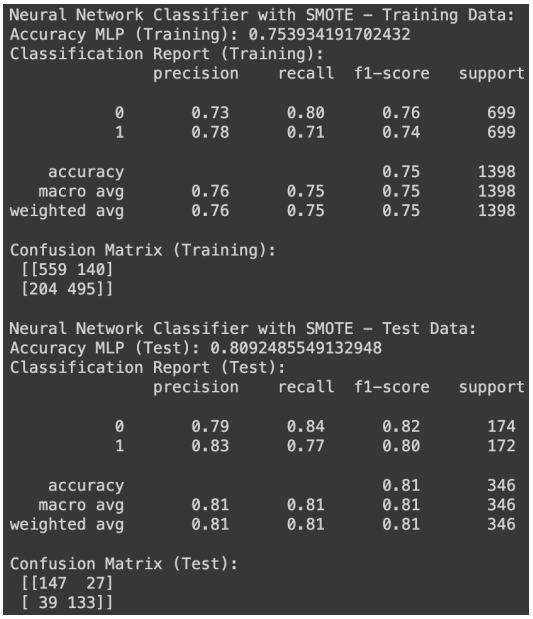


Figure 22. Output After Trying Different Layer Size

This shows that the {(150, 150, 150)} hidden layer configuration works well because it got the best results in all of the studies, both in the training and test datasets. Notably, this configuration did a great job of finding a balance between model complexity and generalization. This led to better performance on data that hadn't been seen before and finally hitting the 80% accuracy goal.

Finally, if getting better accuracy is the main goal, the configuration with hidden layers set to {(150, 150, 150)} is the best choice. The best performance was seen with this setup, which reached and surpassed the 80% accuracy level. This shows that it is good at finding complex patterns in the data. But if you want to keep a balance so that the model doesn't fit too well or too poorly, the setup with hidden layers set to {(200, 200, 200)} is best. This setup has shown equal success on both the training and test datasets. This makes sure that the model works well with new data without being too complicated.

Final Result and Conclusion

I carefully looked at the 83_Loeschcke_et_al_2000_Thorax_&_wing_traits_lab pops.csv dataset to find out what affects the length of a fruit fly's thorax and the weight of its wings. A Neural Network (MLP) predictor was built to guess the sex of flies based on different traits after the dataset was carefully cleaned and analyzed. The focus was on the thorax length by sex. At first, the MLP predictor was only 0.79 percent accurate. For females, it was 0.80 percent accurate and for males, it was 0.78 percent accurate. The recall scores for females were 0.77 and for male they were 0.81. Both groups got F1 scores of 0.79, which means they performed the same way each time. Several tuning methods were used to improve the MLPClassifier's performance. These included adding more neurons and layers to the model, changing the regularisation constant (alpha), improving the early stopping mechanism, raising the max_iter parameter, and changing the validation_fraction. The modified Neural Network classifier got an accuracy of 0.8092.

Performance could also be improved by trying out different model designs and hyperparameters in more depth. Overall, this study helps me learn more about some important traits in fruit flies and shows how useful machine learning models can be in analyzing biological data. The optimized Neural Network model got a lot better at being accurate and generalizing.