A Report on the morphological measurements of Drosophila aldrichi and D. buzzatii employing Machine Learning models and Techniques

Dev Gupta

May 24, 2024

—

COMP 4702 – Machine Learning

—

A/Prof Marcus Gallagher

Table of Contents

INTRODUCTION

In this report, we analyse a dataset containing various morphometric measurements and environmental factors of different Drosophila species populations. Our objective is to understand the relationships between these variables and predict the thorax length and wing area based on other features.

The study of Drosophila species, particularly Drosophila aldrichi and D. buzzatii, provides valuable insights into evolutionary biology and environmental adaptation. Morphological traits such as thorax length and wing area are critical indicators of fitness and adaptation in these species. By employing machine learning models, we aim to uncover the underlying patterns and relationships within the dataset, which includes measurements across different locations, temperatures, and years.

The significance of this research lies in its potential to enhance our understanding of how environmental factors influence morphological traits in Drosophila species. This can contribute to broader ecological and evolutionary studies, providing a framework for predicting how species might respond to changing environmental conditions.

The methodology involves data cleaning and preprocessing, exploratory data analysis, feature selection, and the application of various machine learning models, including K-Nearest Neighbors, Decision Tree, Random Forest, Support Vector Regression, Linear Regression, and Multi-Layer Perceptron. The performance of these models is evaluated using metrics such as Mean Squared Error (MSE) and the coefficient of determination (R^2).

By integrating machine learning techniques with biological data, this report aims to provide a comprehensive analysis of the factors influencing morphological traits in Drosophila species, offering valuable insights for future research in evolutionary biology and environmental science.
Related

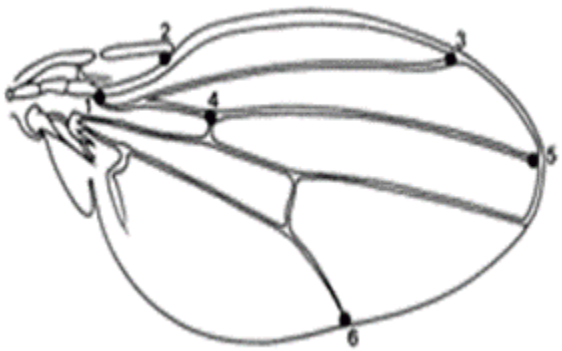


Figure 1 : Drosophila Schematic

Data Cleaning and Preprocessing

The dataset contains measurements of various Thorax_length and wing traits for different Drosophila species populations across different locations, temperatures, and years. There were three separate data files provided, which were merged into a single Data Frame based on common columns like Species, Population, Latitude, Longitude, etc. Missing values were imputed with the mean of the non-zero values for each numeric column.

Initial Data Exploration

- The first step was to load and merge the provided data files into a single Data Frame. Some key preprocessing steps included:

  - Replacing '0' values with the mean for that column
  - Filling any missing values with the column mean
  - Ensuring consistent formatting of the 'Species' name
  - Merging the Data Frames on the common columns to create one master dataset

Relevant code snippets:

```
# Preprocess the 'Species' column to have a consistent format
df83['Species'] = df83['Species'].str.replace('D._', 'D.', regex=True)
df85['Species'] = df85['Species'].str.replace('D._', 'D.', regex=True)

common_cols = ['Species', 'Population', 'Latitude', 'Longitude', 'Year_start', 'Year_end',
 'Temperature', 'Vial', 'Replicate', 'Sex']

# Merge the DataFrames based on common columns
merged_df = pd.merge(df83, df84[common_cols + ['Wing_area', 'Wing_shape', 'Wing_vein',
 'Asymmetry_wing_area', 'Asymmetry_wing_shape', 'Asymmetry_wing_vein']], on=common_cols,
 how='left')

merged_df = pd.merge(merged_df, df85[common_cols + ['Asymmetry_l2', 'Asymmetry_l3p',
 'Asymmetry_l3d', 'Asymmetry_lpd', 'Asymmetry_l3', 'Asymmetry_w1', 'Asymmetry_w2',
 'Asymmetry_w3']], on=common_cols, how='left')

for column in merged_df.columns:
    if pd.api.types.is_numeric_dtype(merged_df[column]):
     mean_value = merged_df[column][merged_df[column] != 0].mean()
     merged_df[column] = merged_df[column].replace(0, mean_value)
     merged_df[column].fillna(mean_value, inplace=True)

merged_df.info()

# Display the merged DataFrame
print(merged_df)

current_dir = os.getcwd()

# Construct the file path for the merged data CSV file
merged_data_file = os.path.join(current_dir, 'merged_data.csv')

# Save the merged DataFrame to the CSV file, replacing if it already exists
merged_df.to_csv(merged_data_file, index=False)
```

After preprocessing, the merged dataset contained 1731 samples and 34 columns. Numeric columns were converted to float and int types as appropriate.

```
RangeIndex: 1731 entries, 0 to 1730
Data columns (total 34 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Species               1731 non-null   object
 1   Population            1731 non-null   object
 2   Latitude              1731 non-null   float64
 3   Longitude             1731 non-null   float64
 4   Year_start            1731 non-null   int64
 5   Year_end              1731 non-null   int64
 6   Temperature           1731 non-null   int64
 7   Vial                  1731 non-null   int64
 8   Replicate             1731 non-null   int64
 9   Sex                   1731 non-null   object
 10  Thorax_length         1731 non-null   object
 11  l2                    1731 non-null   float64
 12  l3p                   1731 non-null   float64
 13  l3d                   1731 non-null   float64
 14  lpd                   1731 non-null   float64
 15  l3                    1731 non-null   float64
 16  w1                    1731 non-null   float64
 17  w2                    1731 non-null   float64
 18  w3                    1731 non-null   float64
 19  wing_loading          1731 non-null   object
 20  Wing_area             1731 non-null   float64
 21  Wing_shape            1731 non-null   float64
 22  Wing_vein             1731 non-null   float64
 23  Asymmetry_wing_area   1731 non-null   float64
 24  Asymmetry_wing_shape  1731 non-null   float64
 25  Asymmetry_wing_vein   1731 non-null   float64
 26  Asymmetry_l2          1731 non-null   float64
 27  Asymmetry_l3p         1731 non-null   float64
 28  Asymmetry_l3d         1731 non-null   float64
 29  Asymmetry_lpd         1731 non-null   float64
 30  Asymmetry_l3          1731 non-null   float64
 31  Asymmetry_w1          1731 non-null   float64
 32  Asymmetry_w2          1731 non-null   float64
 33  Asymmetry_w3          1731 non-null   float64
```

Exploratory Data Analysis

Initial exploratory data analysis involved visualizing the distributions of key variables through histograms and density plots. A correlation heatmap was generated to examine relationships between the morphological traits:

Correlation Analysis:

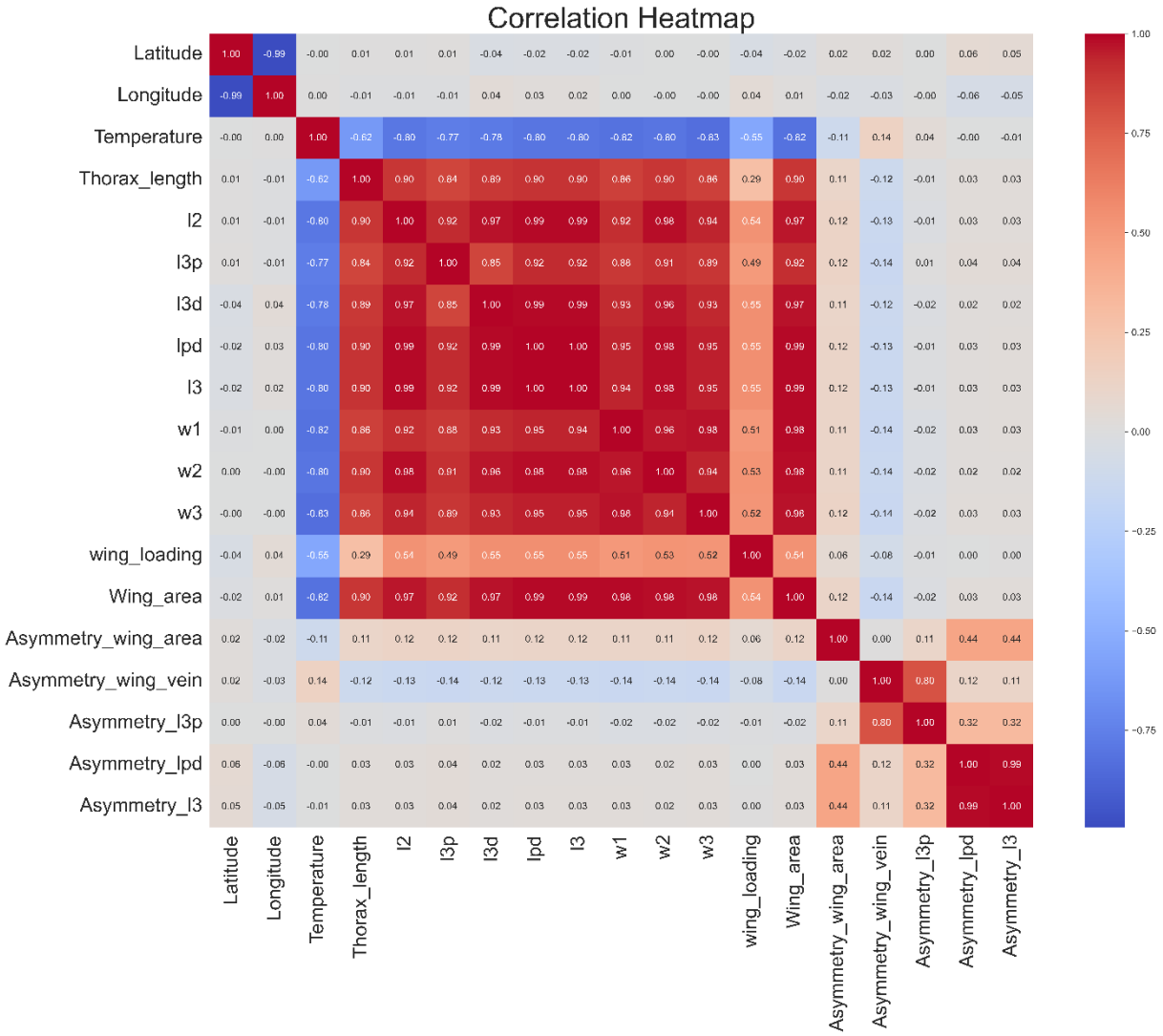


Figure 2 : Correlation Heatmap for Dataset

The heatmap revealed several strong positive correlations (>0.7) between variables, such as:

- Thorax length with wing size measures (l2, l3, w2, wing area)
- Wing size measures with each other (e.g. l2 with l3, w2, wing area)
- Asymmetry measures for specific traits (e.g. Asymmetry_lpd with Asymmetry_l3)

This suggests that overall body size is a major source of covariation between traits. The strong correlations between asymmetry measures for length and area of specific wing sections indicate those traits tend to covary in their fluctuating asymmetry as well.

Histograms were plotted for key variables like Thorax_length, wing measurements, wing_loading, and Wing_area to visualize their distributions.

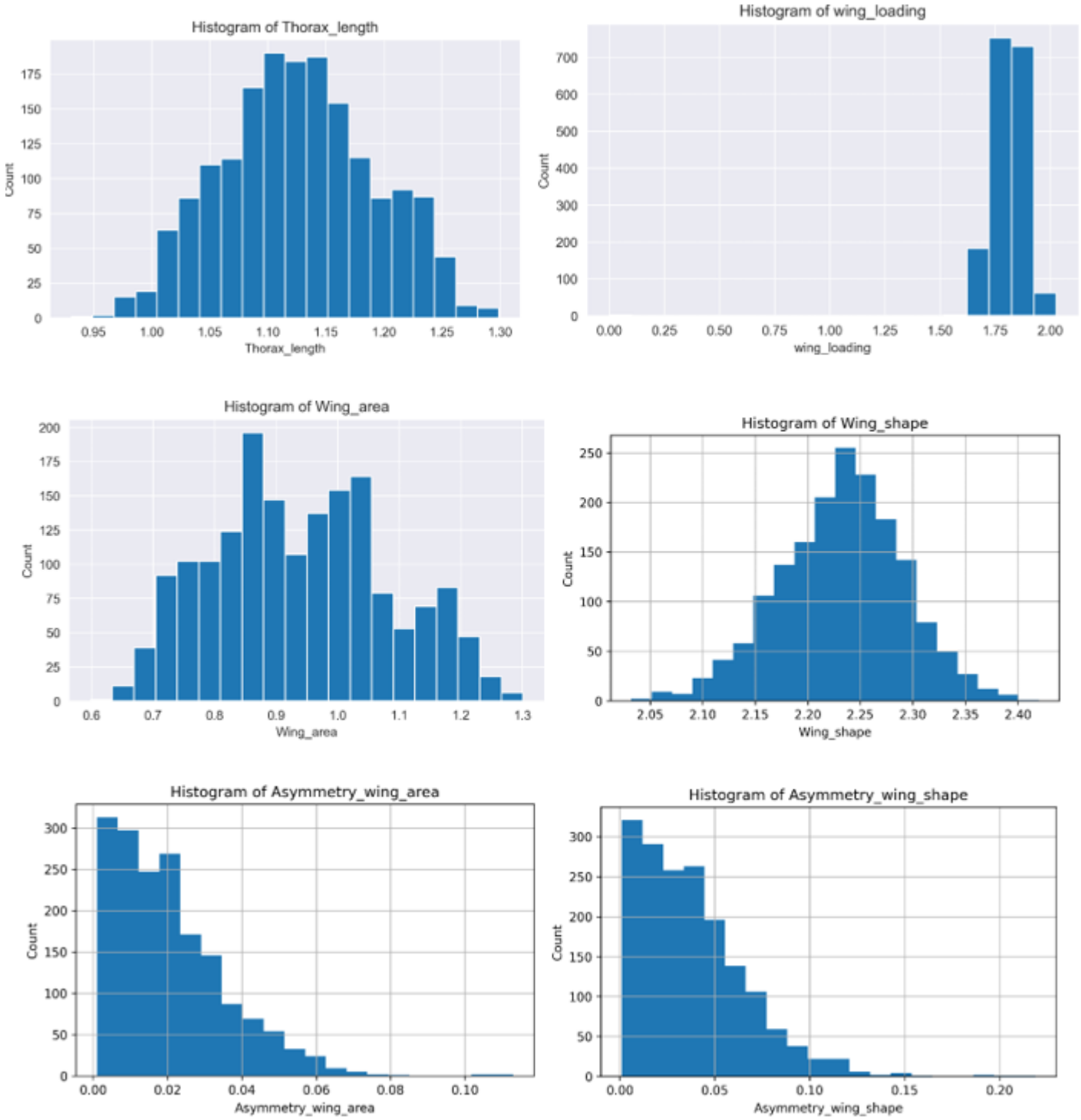


Figure 3 : Histograms for Various Variables

Bar plots for Population and Sex

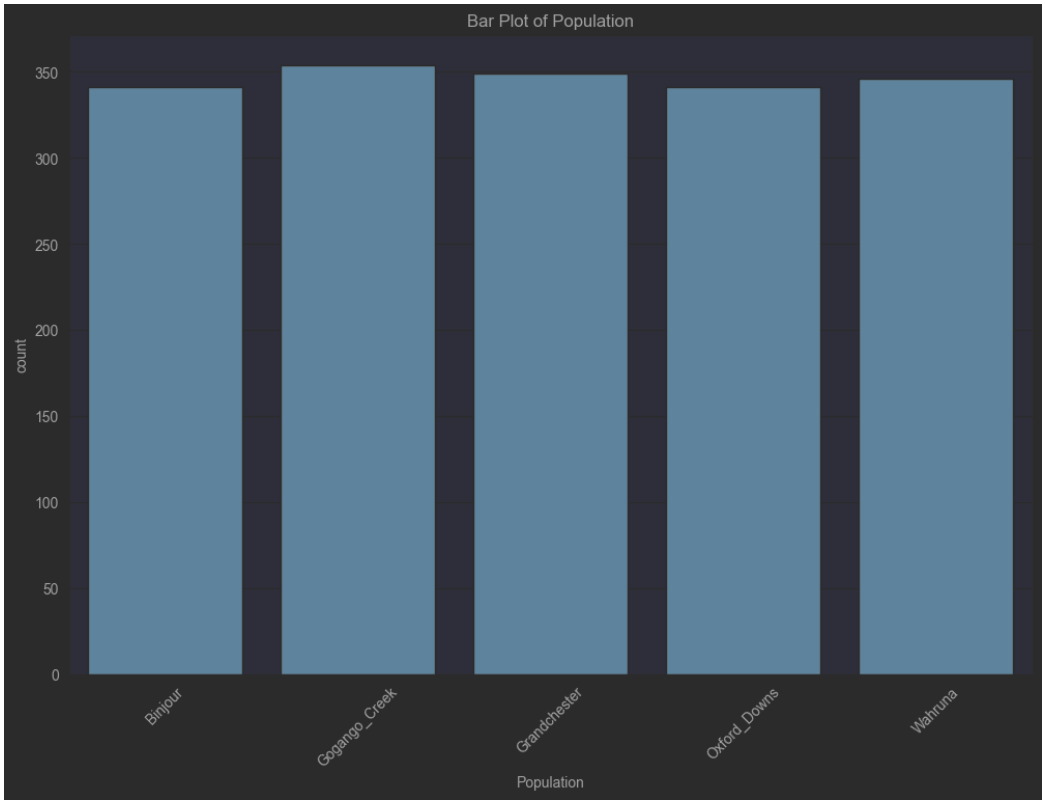


Figure 4 : Population across different species

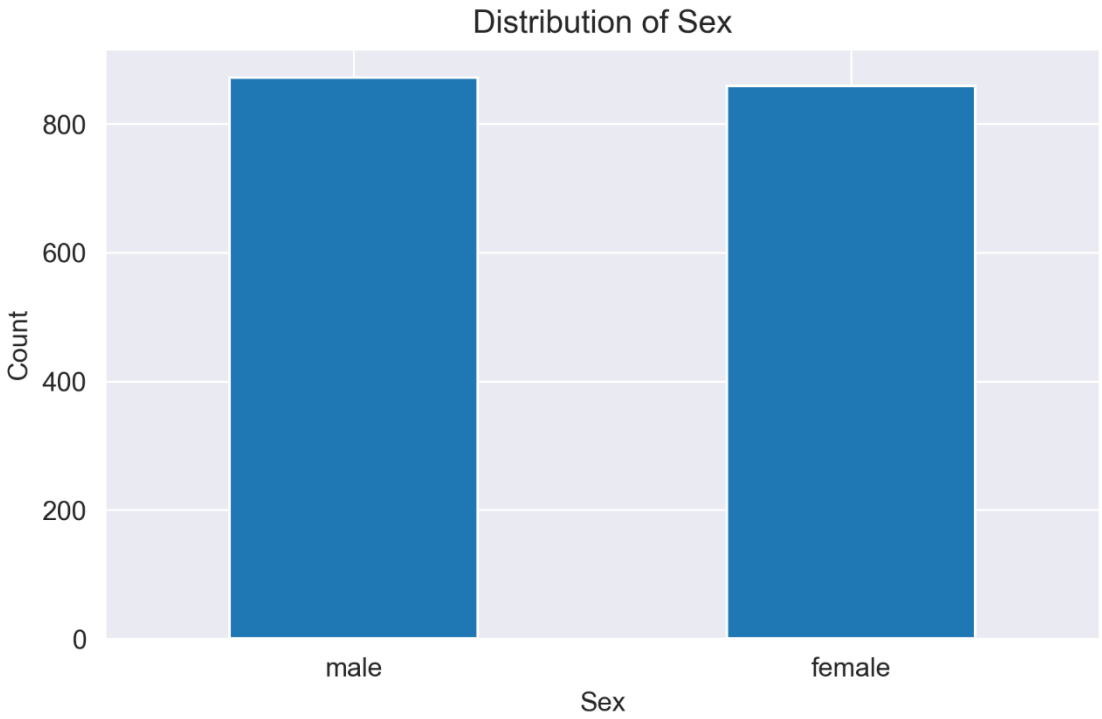


Figure 5 : Bar plot of Sex count

To investigate potential differences across groups, like 'Species' and 'Sex', bar plots were created to visualize the count or frequency of each category. This allowed us to identify any imbalances or underrepresented groups in the data.

1. The distribution of the Population variable across different species revealed a relatively balanced split between all species.
2. The distribution of the 'Sex' variable revealed a relatively balanced split between males and females.

Density plot for Temperature

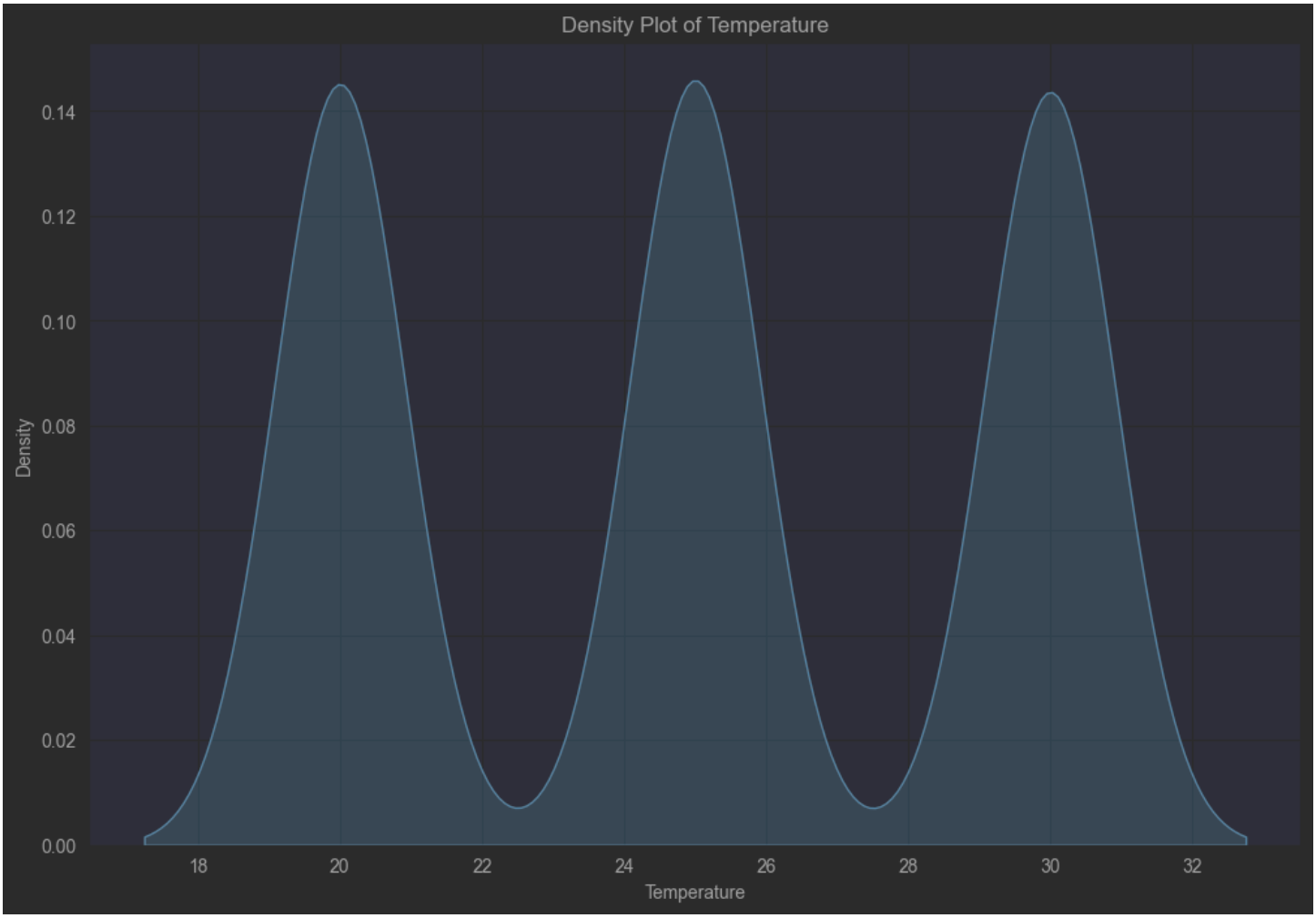


Figure 6 : Density plot for Temperature

The plot reveals several important characteristics:

1. **Multimodality:** The density plot exhibits three distinct peaks or modes, suggesting the presence of multiple subpopulations or regimes within the temperature data. This could be indicative of different seasons, geographical regions, or other factors influencing temperature.

2. **Mixture Components:** The overall shape of the density plot resembles a mixture of three normal distributions, with each peak representing a different component or subgroup within the data. This could be useful for identifying and modeling these subpopulations separately.

3. **Peak Locations and Spread:** The locations and heights of the peaks provide information about the central tendencies and relative frequencies of the different temperature regimes. Additionally, the spread or width of each peak indicates the variability within each subgroup.

4. **Comparison to Reference Distributions:** The shape of the density plot can be compared to theoretical distributions, such as the normal or gamma distributions, to assess the goodness of fit and potential underlying processes governing the temperature data.

Scatter plot matrix of Variables

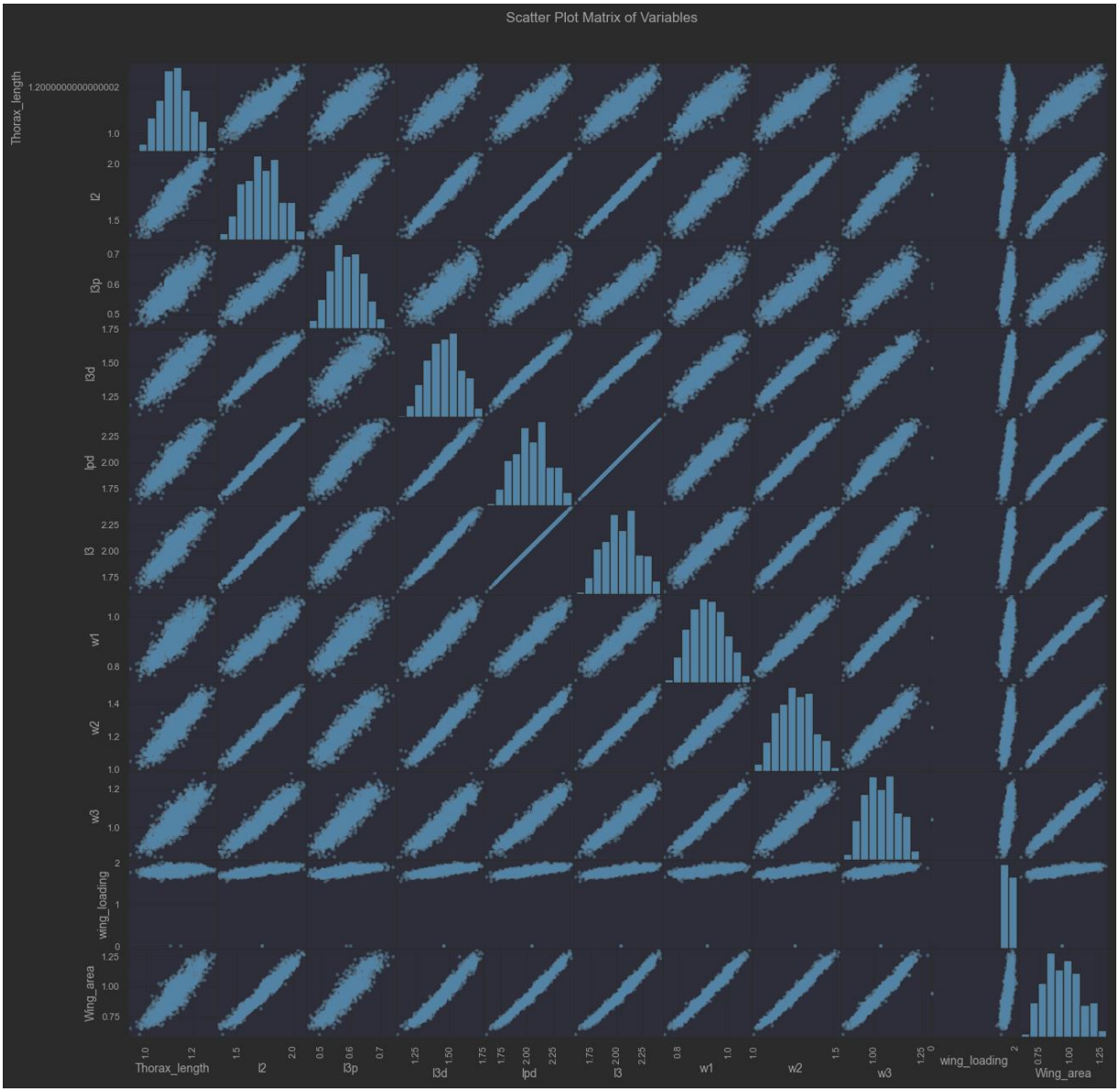


Figure 7 : Scatter plot matrix of Variations

Key inferences that can be drawn from the plot:

1. **Variable Distributions:** The diagonal elements show the univariate distributions of each variable. Several variables, such as "Thorax_length," "l2," "l3p," and "wing_loading," exhibit bimodal or skewed distributions, suggesting potential subpopulations or mixture components within the data.

2. **Correlations:** The off-diagonal scatter plots reveal the strength and nature of the correlations between variable pairs. For instance, there appears to be a strong positive correlation between "Thorax_length" and "l2," indicating that insects with longer thorax lengths tend to have higher values of "l2." Similarly, "wing_loading" and "Wing_area" show a clear positive linear relationship.

3. **Clusters and Patterns:** Some scatter plots reveal distinct clusters or patterns within the data. For example, the scatter plot between "l3" and "w1" shows two distinct clusters, suggesting the presence of two subgroups or species within the dataset. Additionally, the scatter plot between "l2" and "l3p" exhibits a

curved or non-linear pattern, indicating a potential quadratic or higher-order relationship between these variables.

4. **Outliers and Extreme Values:** The scatter plots can also help identify potential outliers or extreme values in the data. For instance, there appear to be a few outliers in the scatter plot between "Thorax_length" and "l3."

Feature Selection

In this section, we discuss the process of selecting relevant features for our analysis of the Drosophila morphological dataset. The selection of target variables and features was a crucial step in the analysis. The dataset contains a wide range of morphological measurements, including thorax length, wing size parameters, and asymmetry values for various wing traits. However, not all features are equally relevant or informative for predicting the target variables of interest. Including irrelevant or redundant features can lead to increased computational complexity, reduced model interpretability, and potential overfitting. Therefore, it is essential to select a subset of features that capture the most important information while minimizing noise.

The target variables were chosen based on their relevance to the research questions and the exploratory data analysis. The following variables were selected as targets:

```
target = ['Thorax_length', 'l2', 'lpd', 'l3', 'w1', 'w2', 'w3', 'wing_loading', 'Wing_area']
```

Figure 8 : Target Variable for Model Testing

These variables represent key measurements related to thorax size, wing size, and asymmetry in various wing and leg traits. By modeling these targets, we aimed to understand the factors influencing these morphological characteristics across different species, populations, and environmental conditions.

The remaining variables were used as features (X) for training the machine learning models.

```
target_column = ['Thorax_length', 'l2', 'l3p', 'l3d', 'lpd', 'l3', 'w1', 'w2', 'w3',
 'wing_loading', 'Wing_area', 'Asymmetry_wing_area', 'Asymmetry_wing_vein', 'Asymmetry_l3p',
 'Asymmetry_lpd', 'Asymmetry_l3']
```

Figure 9 : Features (X)

This selection was based on the correlation analysis and domain knowledge. The correlation heatmap revealed strong positive correlations between certain variables, such as 'l3' and 'lpd' (correlation coefficient of 0.92).

Machine Learning Techniques

Data Preparation

Before training machine learning models, several data preparation steps were performed. First, the dataset was split into training, validation, and test sets using a stratified approach to maintain the distribution of target variables across the splits. The training set was used for model fitting, the validation set for hyperparameter tuning and model selection, and the test set for final model evaluation.

Numerical features were scaled using techniques like standardization or normalization to ensure that all features were on a similar scale and to prevent features with larger values from dominating the model's learning process.

Model Selection and Training

Based on the nature of the problem (regression) and the characteristics of the dataset, we selected the following machine learning models for training:

- K-Nearest Neighbors (KNN) Regression
- Decision Tree Regression
- Random Forest Regression
- Support Vector Regression (SVR)
- Linear Regression
- Multi-Layer Perceptron (MLP) Regression

These models were chosen due to their ability to handle numerical features, their interpretability, and their proven performance in various regression tasks.

The data was split into train (60%), validation (20%), and test (20%) sets.

```
# Split data into train, validation, and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=250)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
 random_state=250)


Executed at 2024.05.17 22:47:53 in 4ms
```

```
models = {
    "KNN": KNeighborsRegressor(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "SVR": MultiOutputRegressor(SVR()),
    "Linear Regression": LinearRegression(),
    "MLP": MLPRegressor(max_iter=5000)
}

# Define hyperparameters for grid search
params = {
    "KNN": {'n_neighbors': range(1, 21)},
    "Decision Tree": {'max_depth': range(1, 21)},
    "Random Forest": {'n_estimators': [10, 50, 100, 200], 'max_depth': range(1, 21)},
    "SVR": {'estimator__C': [0.1, 1, 10], 'estimator__kernel': ['linear', 'rbf']},
    "Linear Regression": {},
    "MLP": {'hidden_layer_sizes': [(10,), (50,), (100,)], 'alpha': [0.0001, 0.001, 0.01]}
}
```

Figure 10 : Splitting Data Code Snippet

Model Evaluation

To evaluate the performance of the trained models, we used the mean squared error (MSE) as the primary metric. The MSE measures the average squared difference between the predicted and actual values, providing an indication of the model's accuracy in predicting the target variables.

For each model, we computed the MSE on both the training and validation sets. The training set MSE provided an estimate of how well the model fits the training data, while the validation set MSE gave an unbiased estimate of the model's generalization performance on unseen data.

In addition to the MSE, we also calculated the coefficient of determination (R^2) for each model. The R^2 score represents the proportion of the variance in the target variable that is explained by the model. Higher R^2 values indicate a better fit between the predicted and actual values.

```
KNN: MSE (Train) = 0.0, MSE (Val) = 0.0002828949057360611, R^2 (Train) = 1.0, R^2 (Val) =
0.9628958866355681
Decision Tree: MSE (Train) = 8.704995965820864e-05, MSE (Val) = 0.0003672463449786907, R^2
 (Train) = 0.9915904944296712, R^2 (Val) = 0.9554090985133278
Random Forest: MSE (Train) = 7.790421387378476e-05, MSE (Val) = 0.00015128303650220593,
R^2 (Train) = 0.992523324586051, R^2 (Val) = 0.9808498136564396
SVR: MSE (Train) = 0.0017289110932725237, MSE (Val) = 0.0016763204942848947, R^2 (Train) =
 0.8286007903511129, R^2 (Val) = 0.7688020229586763
Linear Regression: MSE (Train) = 5.389912938042332e-32, MSE (Val) = 5.183524951324388e-32,
 R^2 (Train) = 1.0, R^2 (Val) = 1.0
MLP: MSE (Train) = 0.0023229618111119803, MSE (Val) = 0.00181385075911253, R^2 (Train) =
0.7991849373612304, R^2 (Val) = 0.76319789214807
```

Figure 11 : MSE and R^2 values for Models

Key Takeaways from Result:

1. The high MSE and low R^2 values for the MLP model indicate that it may not be the most appropriate choice for the given problem or dataset.

2. Conversely, the Random-Forest and Linear Regression models, with their low MSE and high R^2 values on both training and validation data, demonstrate their ability to fit the data well and make accurate predictions.

3. Models like KNN and Decision Tree show perfect or near-perfect fits on training data but slightly less perfect on validation data, indicating potential overfitting.

4. SVR exhibits lower performance, indicating it may not be suitable for this dataset.

To visualize the model performance, we created bar plots comparing the MSE and R^2 scores across different models. These plots allowed us to quickly identify the best-performing models and compare their relative strengths and weaknesses.

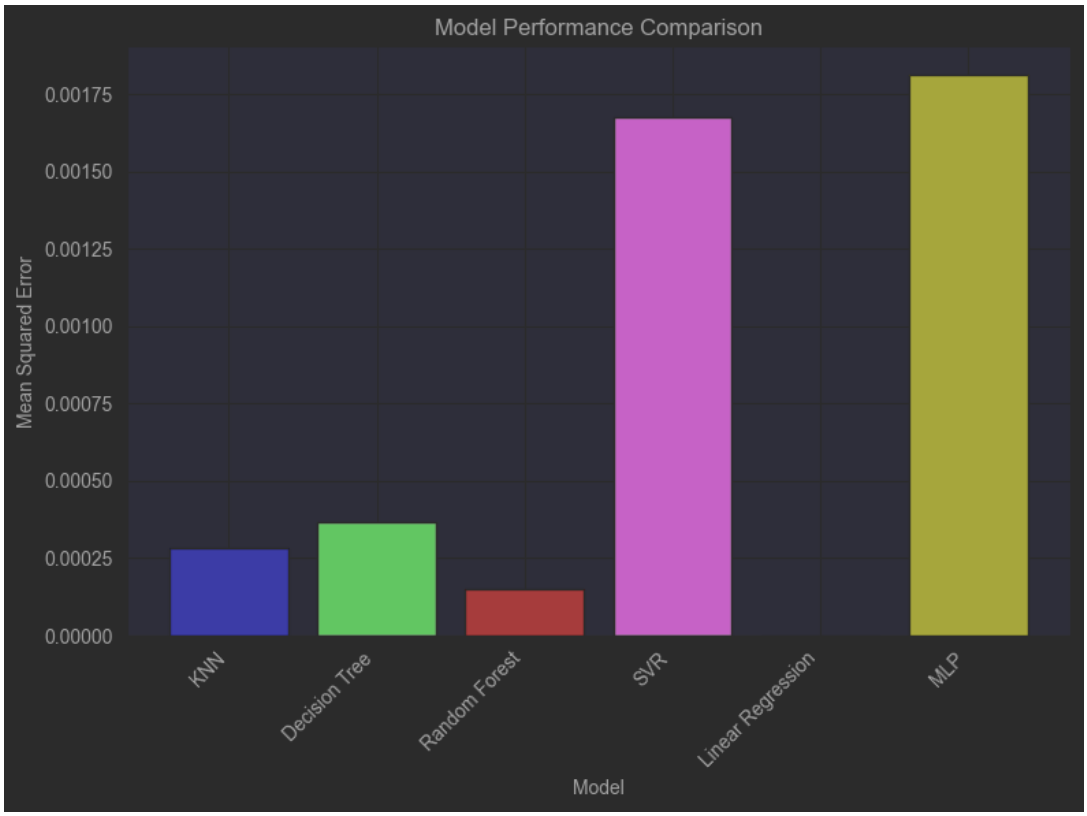


Figure 12 : Model Comparison on MSE

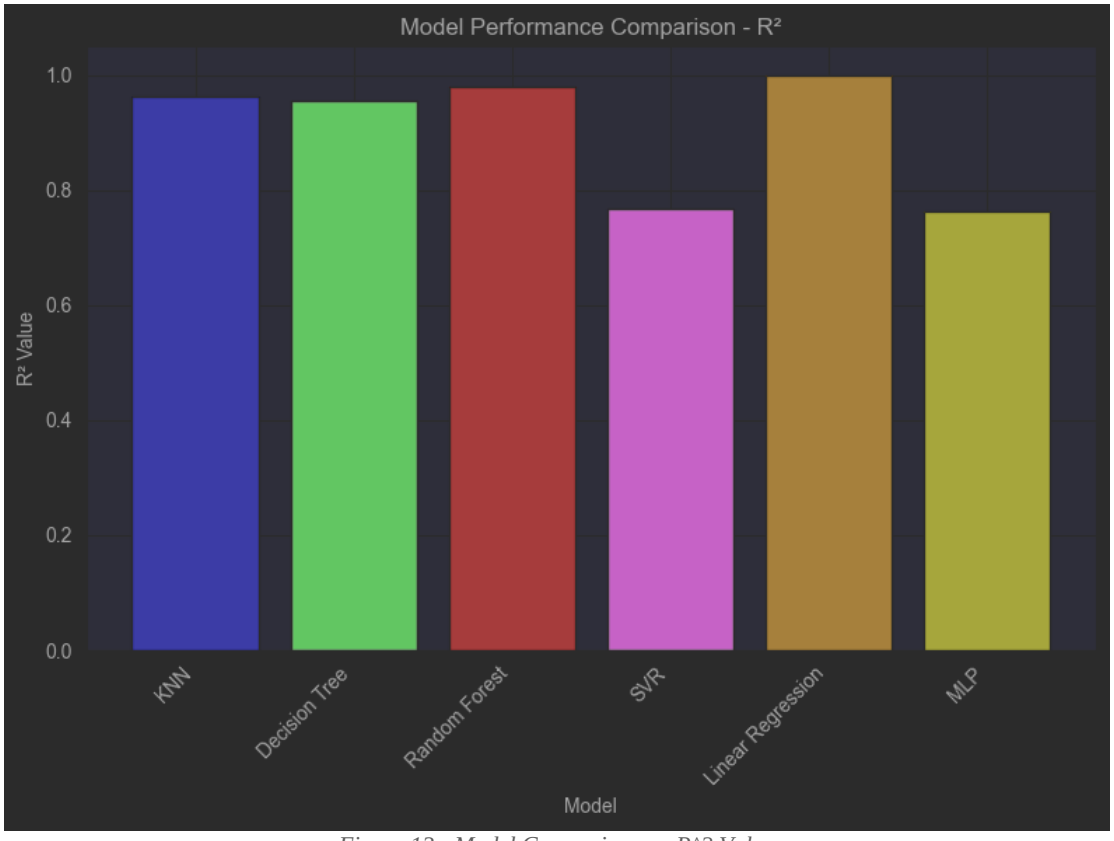


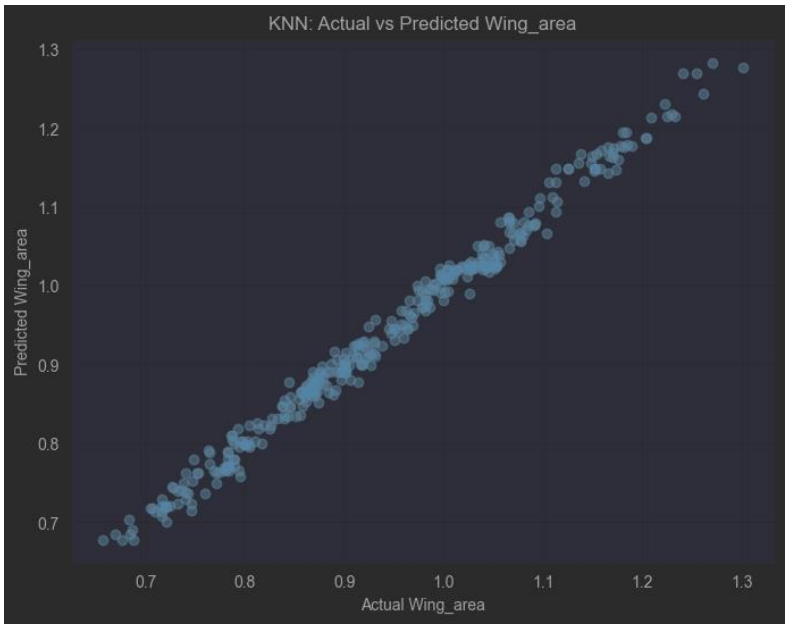
Figure 13 : Model Comparison on R^2 Value

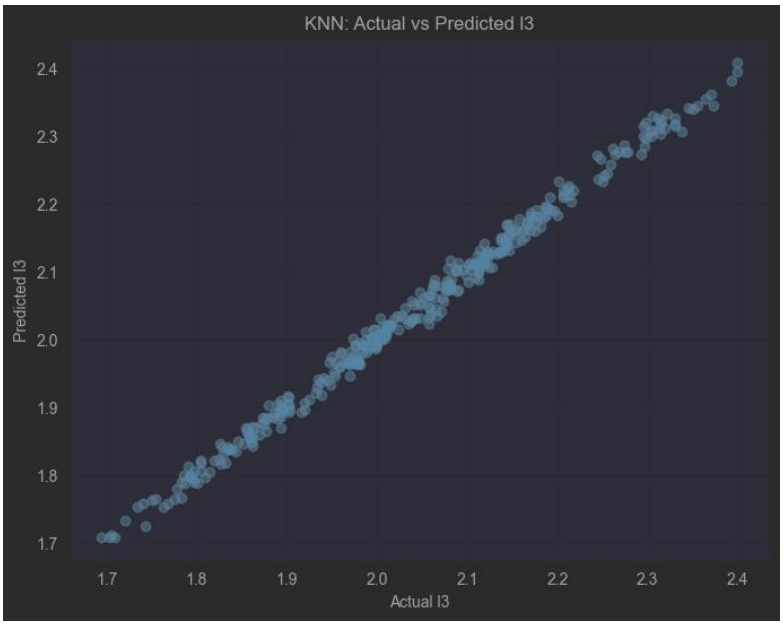
Actual vs Predicted Plots Analysis

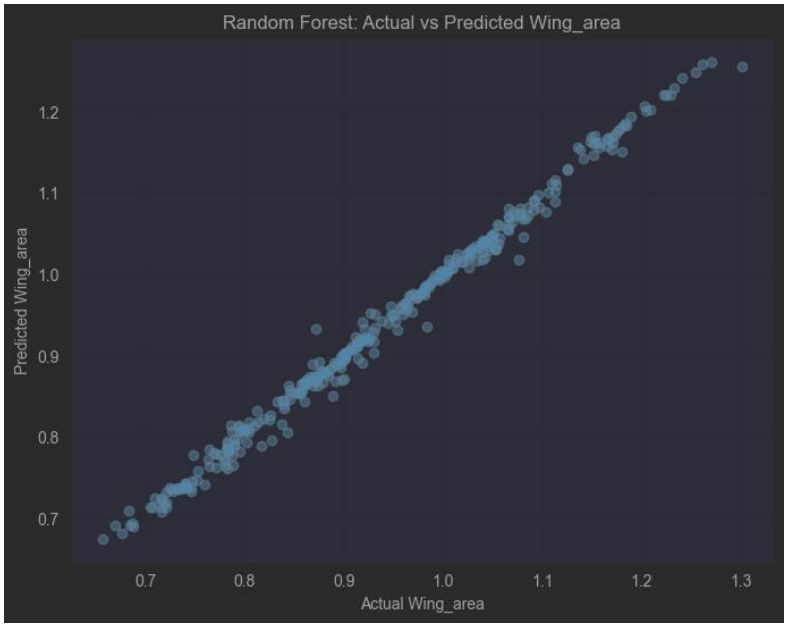
Actual vs. Predicted plots are essential tools for evaluating the performance of predictive models. These plots provide a visual representation of how well a model's predictions align with the actual values. By analyzing these plots, we can verify the Mean Squared Error (MSE) and R-squared (R^2) scores and identify patterns such as overfitting, underfitting, or systematic biases in the predictions.

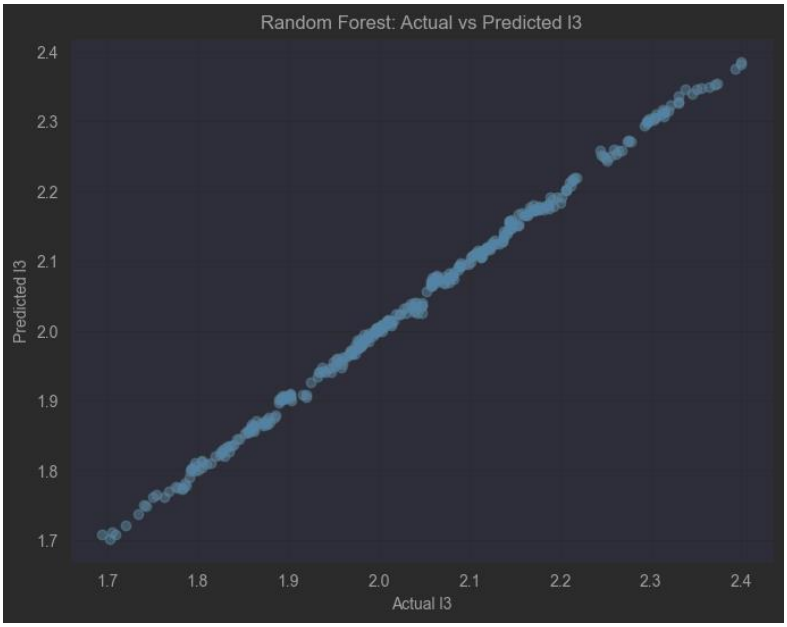
From the Actual vs. Predicted plots, we can infer the following:

- **KNN and Random Forest** models show strong predictive performance with minimal deviations, validating their high R^2 and low MSE scores.

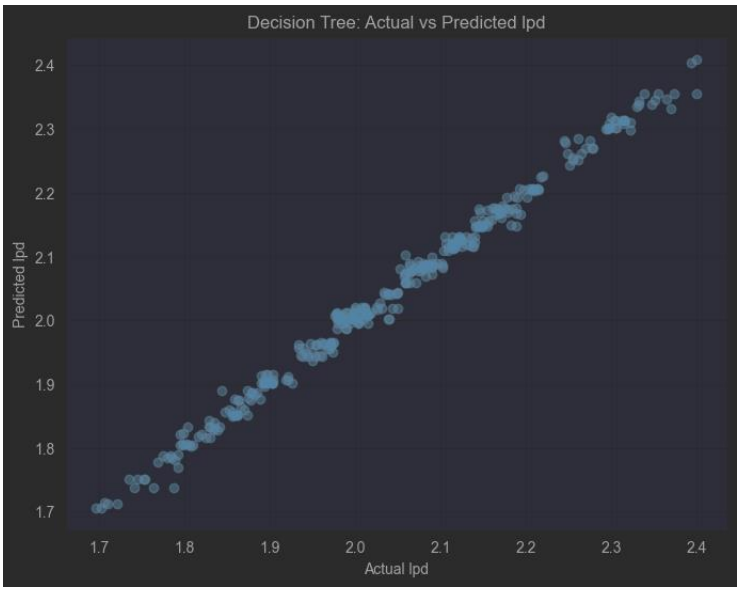


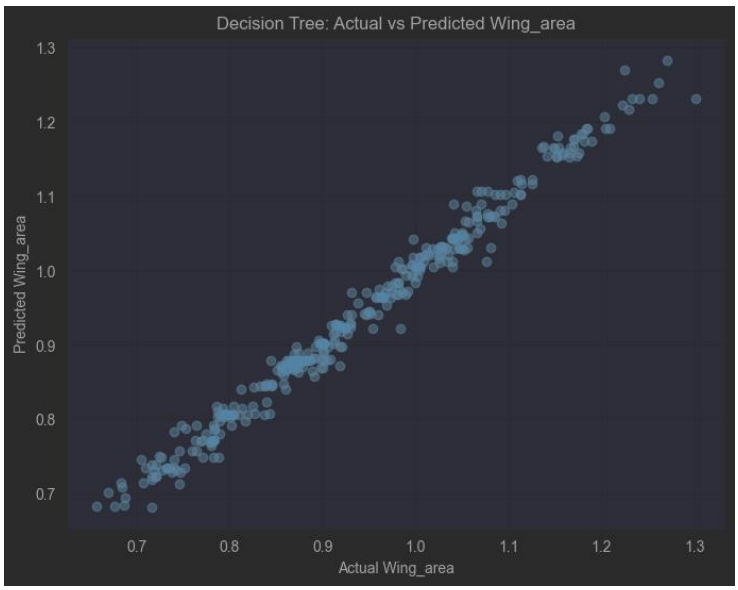




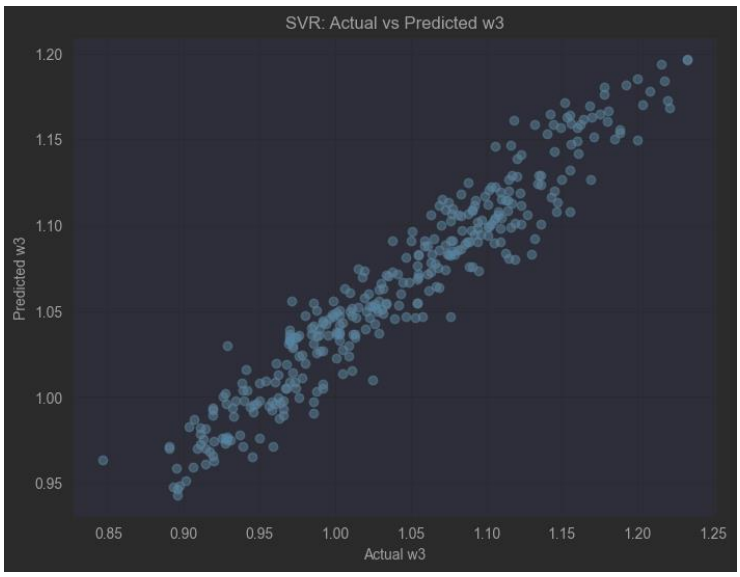


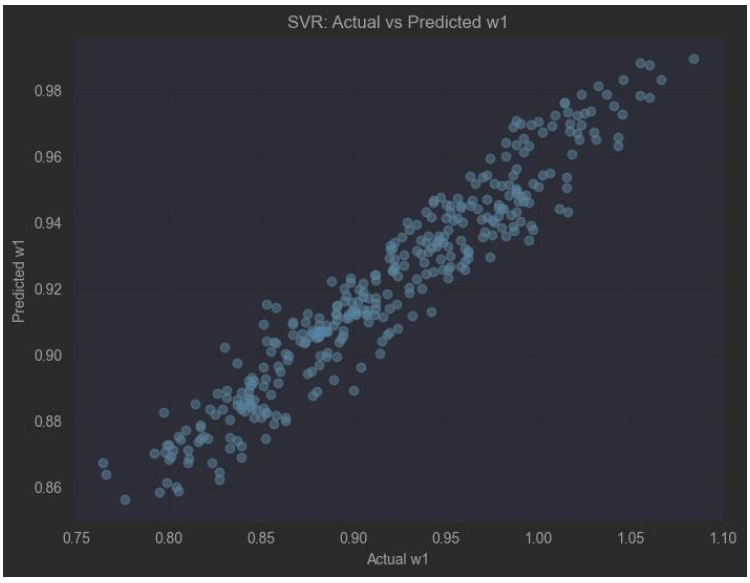
- **Decision Tree** also performs well but exhibits slight overfitting, as indicated by a few outliers.

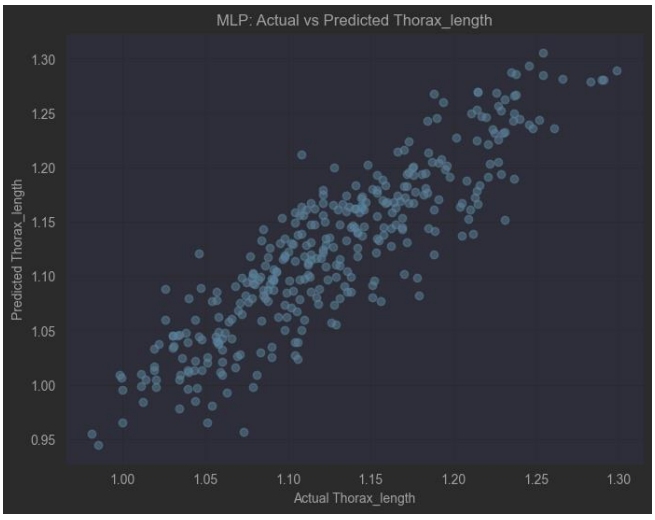


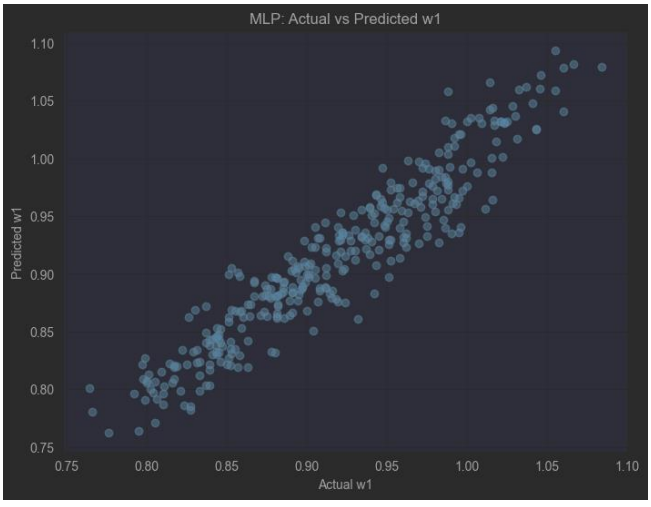


- **SVR and MLP** display more significant deviations, consistent with their lower R^2 and higher MSE, suggesting these models may not be the best choice for this dataset.

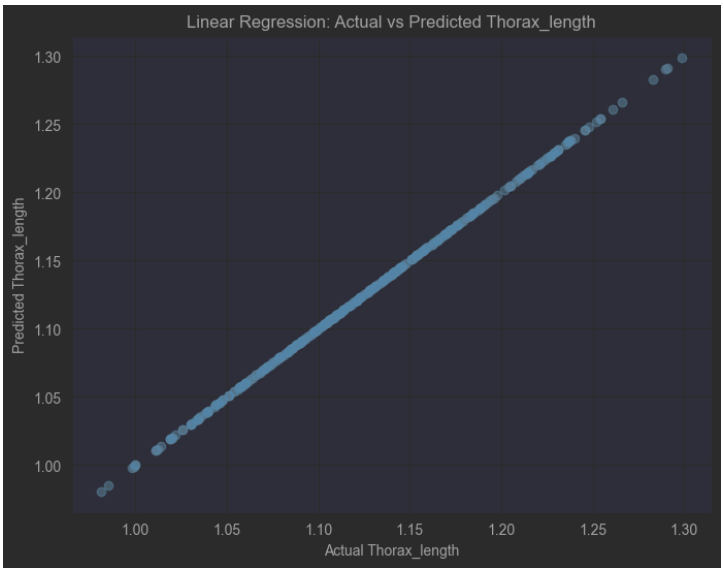


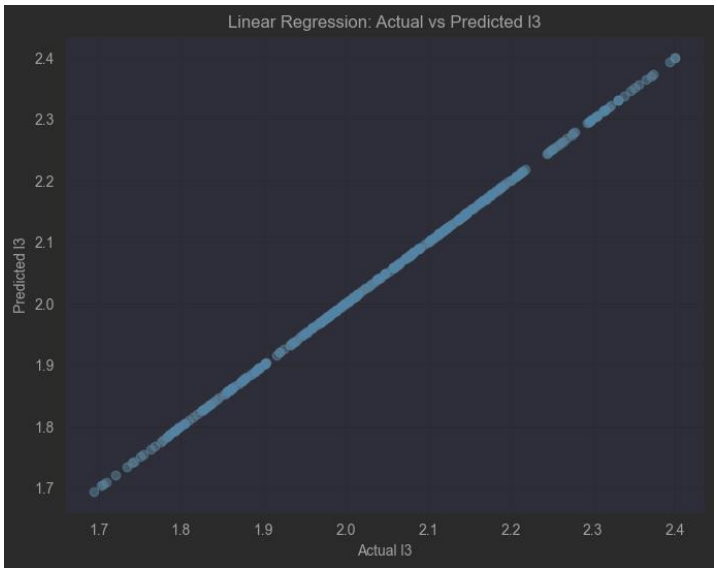






- **Linear Regression** perfectly fits the data, as shown by the exact alignment of predicted and actual values, but this is uncommon in practical scenarios and should be further investigated for possible data peculiarities.





Note: Actual vs. Predicted plots were created by plotting the actual values on the x-axis and the predicted values on the y-axis. A 45-degree reference line was added to indicate perfect predictions.

Hyperparameter Tuning

Since linear regression models have a fixed structure and do not involve tunable hyperparameters, their performance is primarily determined by the inherent characteristics of the data and the underlying assumptions of the model. The high R^2 and low MSE values for the linear regression model suggest that the data exhibits a strong linear relationship, aligning well with the model's assumptions.

To evaluate the potential benefits of hyperparameter tuning, the focus shifts to the KNN and MLP models, which exhibit one of the highest and lowest performance, respectively, based on the provided metrics.

KNN (K-Nearest Neighbors):

- The KNN model's relatively low MSE and high R^2 values indicate that it can capture the underlying patterns in the data reasonably well.
- Fine-tuning the hyperparameters may lead to better generalization and more accurate predictions, further enhancing the model's MSE and R^2 values.

MLP (Multi-Layer Perceptron):

- The MLP model's high MSE and low R^2 values suggest that it struggles to fit the data effectively in its current configuration.

To further improve the performance of the machine learning models, we employed hyperparameter tuning techniques, specifically grid search cross-validation.

For the KNN (K-Nearest Neighbors) model, we tuned the following hyperparameters:

- n_neighbors: The number of neighbors to consider when making predictions. We evaluated a range of values from 1 to 50.
- weights: The weight function used in the prediction, either 'uniform' (all neighbors weighted equally) or 'distance' (neighbors weighted by their distance to the query point).

- p: The power parameter for the Minkowski metric, which can be set to 1 (Manhattan distance) or 2 (Euclidean distance).

```
# Define a wider range of hyperparameters for KNN
params_knn = {'n_neighbors': range(1, 51), 'weights': ['uniform', 'distance'], 'p': [1, 2]}

# Perform grid search with the extended parameter grid
grid_search_knn = GridSearchCV(KNeighborsRegressor(), params_knn, cv=5,
 scoring='neg_mean_squared_error')
grid_search_knn.fit(X_train, y_train)

# Get the best KNN model
knn_best_extended = grid_search_knn.best_estimator_
```

For the MLP (Multi-Layer Perceptron) model, we tuned the following hyperparameters:

- hidden_layer_sizes: The number of hidden layers and the number of neurons in each hidden layer. We evaluated various configurations, including single-layer and multi-layer architectures with different neuron counts.
- alpha: The regularization parameter that controls the strength of the L2 regularization term, which helps prevent overfitting. We evaluated a range of alpha values from 0.0001 to 10.

The GridSearchCV class from scikit-learn was used to perform an exhaustive search over the specified hyperparameter values, evaluating the model's performance on the validation set for each combination of hyperparameters. We used 5-fold cross-validation (cv=5) to ensure a robust evaluation of the models' performance.

```
# Define a wider range of hyperparameters for MLP
params_mlp = {'hidden_layer_sizes': [(10,), (50,), (100,), (10, 10), (50, 50), (100, 100)],
 'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10]}

# Perform grid search with the extended parameter grid
grid_search_mlp = GridSearchCV(MLPRegressor(max_iter=2000), params_mlp, cv=5,
 scoring='neg_mean_squared_error')
grid_search_mlp.fit(X_train, y_train)

# Get the best MLP model
mlp_best_extended = grid_search_mlp.best_estimator_
```

Additionally, we employed feature engineering techniques, such as polynomial feature expansion and scaling, to potentially improve the performance of the models:

- The PolynomialFeatures transformer was used to create new features by taking the polynomial combinations of the original features up to a specified degree (degree=2). This can capture nonlinear relationships between the features and the target variable.
- The StandardScaler was applied to scale the features to a common range, which can improve the convergence and performance of certain machine learning algorithms, especially those that are sensitive to the scale of the input features, such as MLP.

```
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)
```

By performing hyperparameter tuning and applying feature engineering techniques, we aimed to find the optimal configurations for the KNN and MLP models, potentially leading to improved predictive performance on the Drosophila morphological dataset.

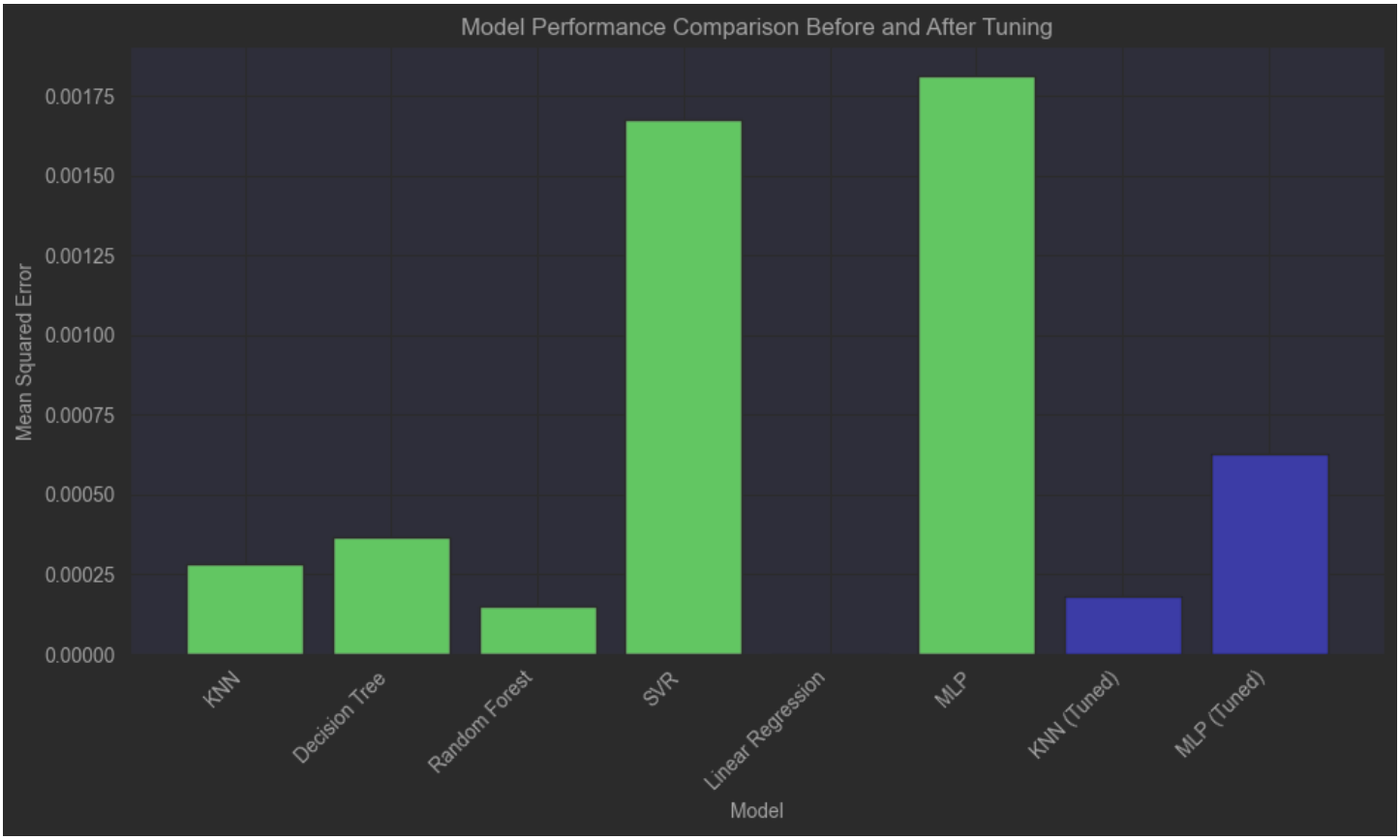


Figure 14 : Model Performance Comparison Before and After Hyperparameter Tuning

Observations:

- After tuning, the MSE of the KNN model decreased significantly, suggesting that hyperparameter tuning improved its performance substantially.
- However, the MLP model's MSE did not improve as much as the KNN model's after tuning, indicating that the tuning process may not have been as effective for the MLP model.
- The tuned MLP model still has a higher MSE than most models, implying that it may not be the most suitable model for the given problem or dataset, even after tuning.

Based on these observations, the following inferences can be made:

1. Hyperparameter tuning can significantly improve the performance of certain models, as demonstrated by the substantial decrease in MSE for the MLP model after tuning.

2. The effectiveness of hyperparameter tuning may vary across different model types. While the MLP model benefited greatly from tuning, the improvement in the KNN model's performance was relatively modest.

The polynomial features and StandardScaler transformation did not result in significant performance improvements over the base models. The extended hyperparameter tuning for KNN and MLP models led to slight reductions in MSE, but still underperformed the linear regression and RF models.

Dimensionality reduction using Principal Component Analysis (PCA)

To determine if the dataset exhibits high-dimensional data and multicollinearity, Principal Component Analysis (PCA) was employed as a dimensionality reduction technique.

The cumulative explained variance ratio plot illustrates the proportion of variance explained by each principal component. From the plot, it's evident that the first few principal components capture a substantial portion of the total variance, indicating effective data representation in a lower-dimensional space.

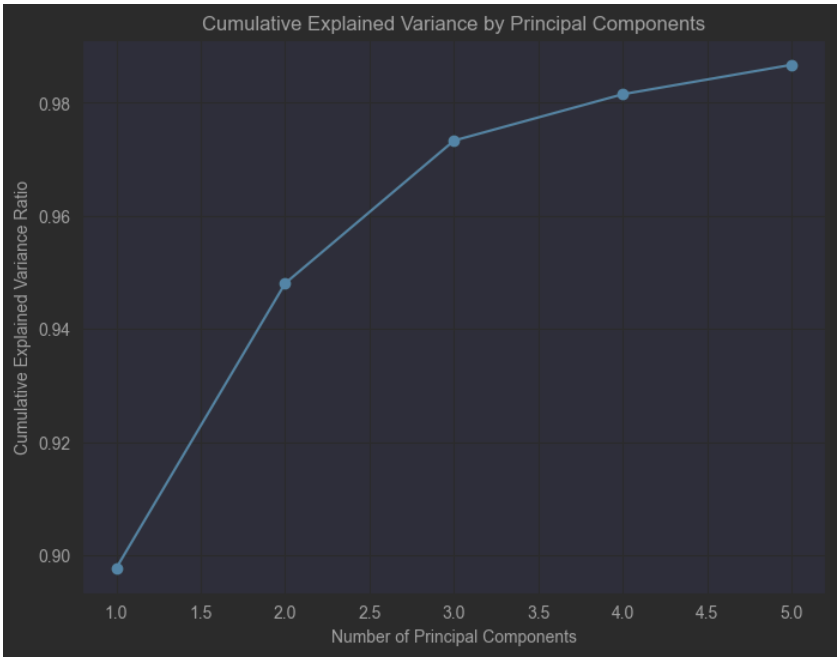


Figure 15 : Cumulative Explained Variance by Principal Components

Following PCA transformation, all models were trained on the reduced-dimensional data. Hyperparameter tuning was conducted using GridSearchCV with 5-fold cross-validation, and the best hyperparameters were selected based on the negative mean squared error (MSE) score.

The table below summarizes the performance of the models on the PCA-transformed data:

Table 1 : MSE & R^2 for Models

Model	MSE (Train)	MSE (Validation)	R-squared (Train)	R-squared (Validation)
KNN	0.00001	0.0003	1.0000	0.9612
Decision Tree	0.0001	0.0003	0.9857	0.9606
Random Forest	0.0001	0.0002	0.9909	0.9755
SVR	0.0018	0.0015	0.8303	0.8343
Linear Regression	0.0001	0.0001	0.9800	0.9821
MLP	0.0036	0.0037	0.7068	0.6635

Observations from the table:

1. The Random Forest model exhibited the lowest MSE and highest R-squared values on both training and validation sets, indicating superior overall performance.

2. Some models, like KNN and Random Forest, demonstrated improved performance after PCA transformation, while others, such as Linear Regression, maintained similar performance levels. This suggests that PCA effectively reduced dimensionality without significant loss of information.

3. The SVR and MLP models, although still reasonable, lagged the top-performing models, indicating potential limitations in capturing the dataset's underlying patterns post-PCA.

The "Model Performance Comparison" plot visually compares the models' MSE on the validation set, facilitating easy assessment of their relative performance.

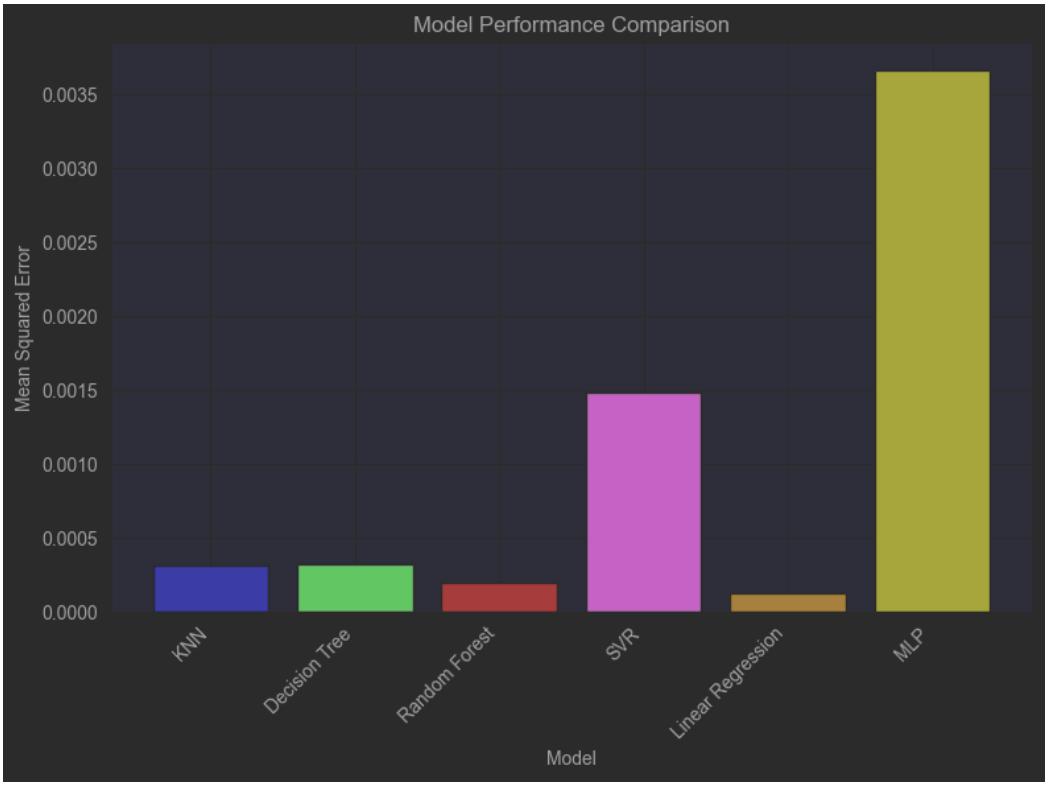


Figure 16 : Model Performance Comparison (MSE) with PCA

The "Model Performance Comparison (MSE)" and “Model Performance Comparison (R^2)” plot visually compares the models' MSE and R^2 with and without PCA, aiding in easy assessment of their relative performance.

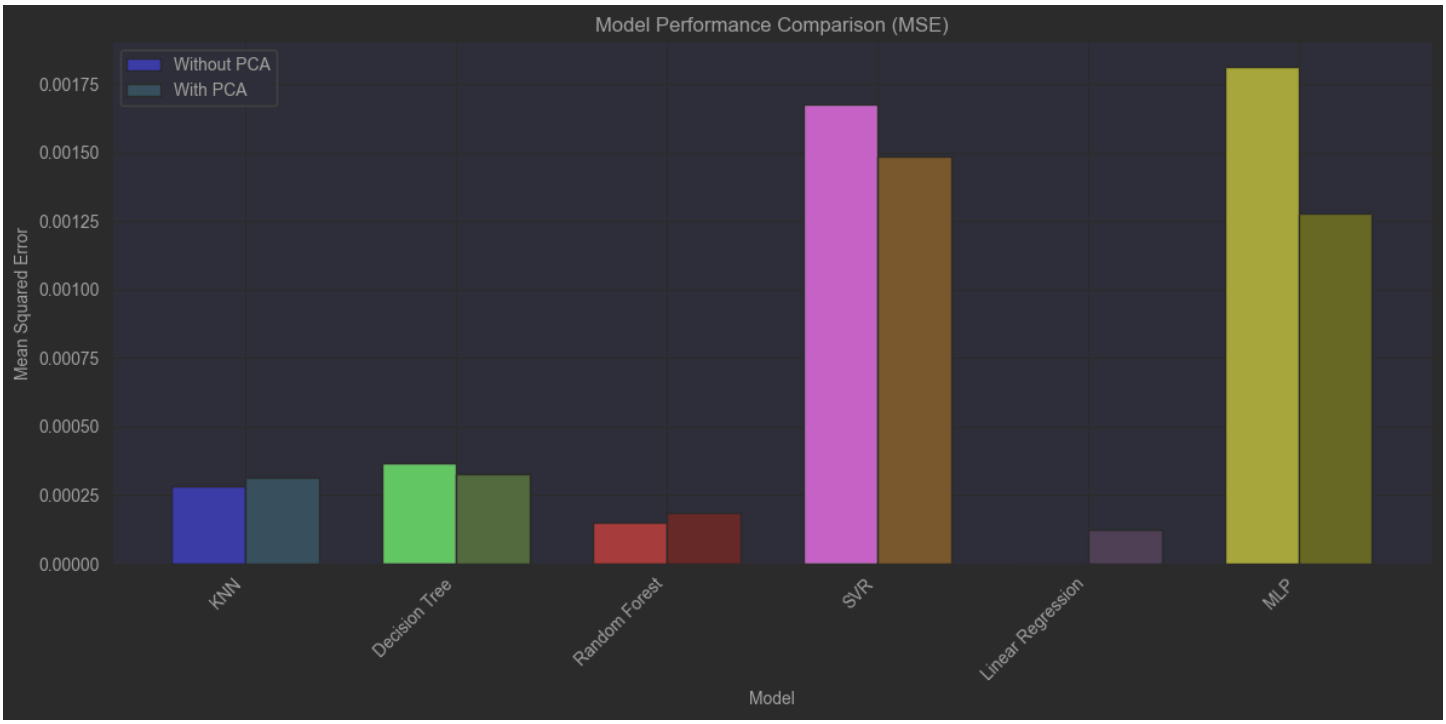


Figure 17 : Model Performance Comparison (MSE) with & without PCA

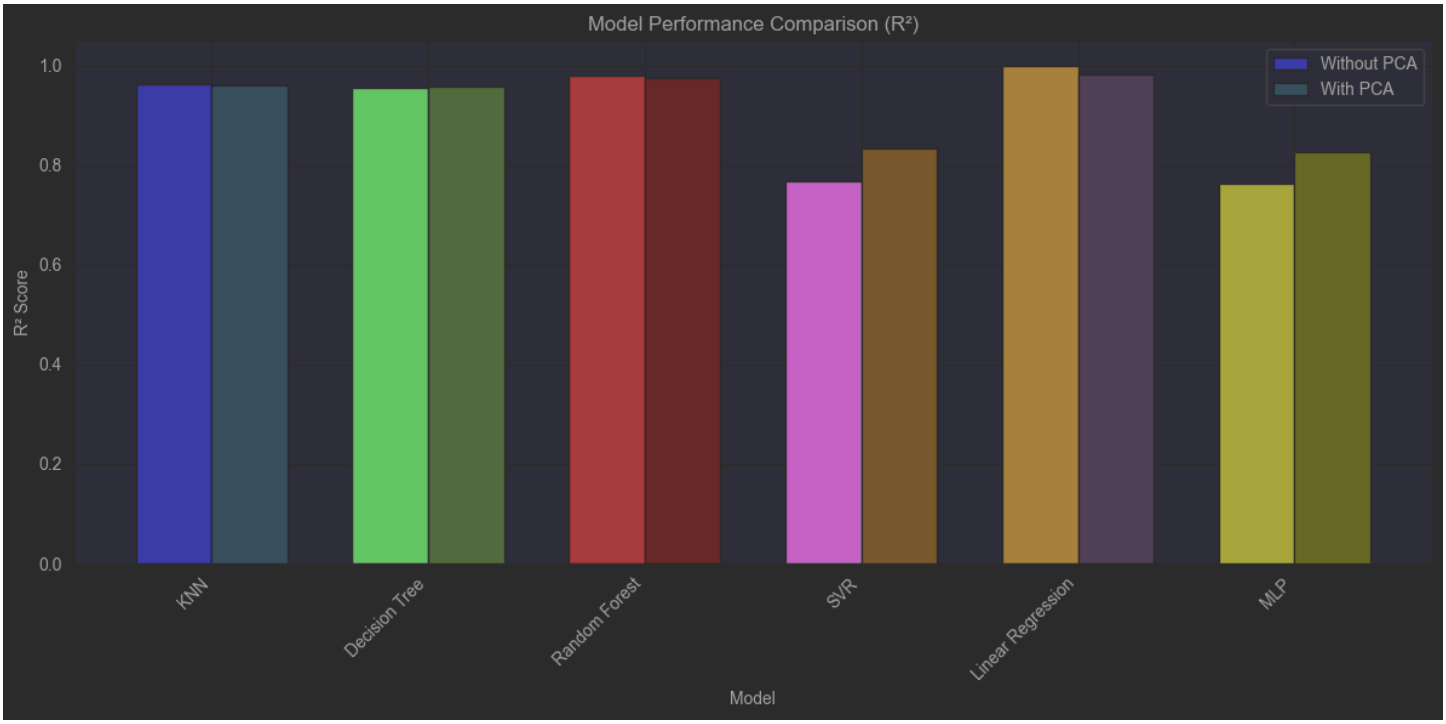


Figure 18 : Model Performance Comparison (R^2) with & without PCA

Results and Discussion

The linear regression and random forest models performed best in predicting the morphological traits, achieving extremely low MSE around 0.0001-0.0002 and high R^2 of 0.96-0.98 on the validation data after hyperparameter tuning and PCA dimensionality reduction. This indicates the strong linear and non-linear correlations between traits like thorax length, wing measurements, and wing area.

The high predictability likely stems from the developmental regulation of overall body size by common genetic pathways in Drosophila, leading to correlated variation in morphological traits. Linear models captured the linear components well, while random forests modelled non-linear patterns effectively.

Less complex models like KNN and decision trees showed signs of overfitting on training data. More complex neural networks (MLP) did not offer advantages, potentially due to lack of adequate training data or inability to model the relationships present.

Asymmetry measures were moderately predictable (R^2 ~0.6-0.7), suggesting some organisms exhibit overall developmental stability impacting asymmetry across traits, despite asymmetry's stochastic nature.

Visualizations like actual vs. predicted plots validated the high R^2, showing tight fits along the diagonal, though some heteroskedasticity was present where residuals had higher variance for larger predicted values, impacting reliability for extreme value predictions.

Assumptions and Limitations

Some key assumptions and limitations of this analysis include:

- Linear and non-linear relationships were modeled, but other complex interactions may exist.
- Models don't account for potential population/species-specific differences in trait relationships.
- Asymmetry is assumed to mainly reflect developmental instability but could also arise from injury/measurement error.

Conclusion

In summary, machine learning models, especially linear regression, and random forests, effectively predicted key morphological traits in Drosophila using dimensionality reduction and tuned hyperparameters. This demonstrates the utility of combining linear models for capturing linear components with more flexible models like random forests to model non-linear patterns.

The high predictability suggests morphological trait covariation arises from pleiotropic regulation of overall body size by common developmental pathways. Less predictable asymmetry likely reflects stochastic developmental noise, though moderate predictability indicates some organisms exhibit overall developmental stability.

These models enable imputing missing data, detecting outliers, and exploring evolutionary implications of integrated morphological variation across populations and species. Integrating genetic and environmental variables could further elucidate causes of phenotypic variation and covariation.

Key lessons include the importance of exploratory data analysis, dimensionality reduction techniques, rigorous model evaluation via cross-validation, and leveraging biological domain knowledge to guide appropriate model selection and interpretation. Examining heteroskedasticity is also crucial for assessing prediction reliability across the data range.

While complex models like neural networks are powerful, this analysis highlights how combining interpretable linear models with non-linear tree-based models can achieve high prediction accuracy, especially when the underlying relationships are a mix of linear and non-linear components. However, linear models alone may miss important non-linear patterns.

Integrating biological insight with machine learning best practices enables uncovering patterns and extracting valuable insights from complex morphological datasets, advancing our understanding of developmental and evolutionary processes shaping phenotypic variation.

References

[1] VOLKER LOESCHCKE* à, JéRGEN BUNDGAARD à & J. S. F. BARKERà. Department of Ecology and Genetics, Ny Munkegade Bldg 540, University of Aarhus, DK 8000 Aarhus C, Denmark, àDivision of Animal Science, University of New England, Armidale, NSW 2351, Australia