Module3 - Graded assignment

February 3, 2025

0.1 Linear Regression for Classification vs Logistic Regression: Palmer Penguins Analysis

0.1.1 Introduction

In this project, I explore the use of **linear regression for classification** using the **Palmer Penguins dataset**. The goal is to classify penguins into three species—**Adélie**, **Gentoo**, and **Chinstrap**—based on features such as **bill length and flipper length**. I also compare the performance of **linear regression** with **logistic regression**, a more suitable classifier for categorical predictions.

The **Palmer Penguins dataset** is a modern, real-world alternative to the classic **Iris dataset** for data exploration and classification tasks. It was collected between **2007 and 2009** by **Dr. Kristen Gorman** as part of the **Palmer Station Long Term Ecological Research (LTER) Program** in Antarctica. The dataset contains **344 penguins** from three species, recorded across **three different islands** in the Palmer Archipelago.
(Source: Palmer LTER)

1 Dataset Selection and Preparation

The dataset includes **biological and environmental measurements** of the penguins, providing valuable features for classification:

Feature	Description
Species	Penguin species (*Adélie*, *Gentoo*, *Chinstrap*)
Island	Island where the penguin was observed (*Biscoe*, *Dream*, *Torgersen*)
Bill Length (mm)	Length of the dorsal ridge of the penguin's bill
Bill Depth (mm)	Depth of the penguin's bill
Flipper Length (mm)	Length of the penguin's flipper
Body Mass (g)	Weight of the penguin
Sex	Gender of the penguin (*male* or *female*)
Year	Year of observation (*2007, 2008, or 2009*)

This dataset is particularly useful for **classification problems** because it contains **well-separated yet slightly overlapping features**, making it an interesting challenge for different classification models.

By applying **linear regression and logistic regression**, I analyze their effectiveness in predicting the correct species and discuss the challenges in feature selection, model limitations, and classification accuracy.

1.0.1 Key Steps:

- Loaded the dataset using `seaborn`.
- Handled missing values using `SimpleImputer` (mean imputation for numerical features and mode imputation for categorical features).
- Selected `bill_length_mm` and `flipper_length_mm` as predictors and one-hot encoded the target variable (`species`).
- Train-Test split.
- Build Linear and logistic models and compare their results.

[1]:
```
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Required for modeling and evaluation
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
 ↪recall_score, classification_report
from sklearn.impute import SimpleImputer
from matplotlib.colors import ListedColormap
from matplotlib.lines import Line2D

# Load the Palmer Penguins dataset
penguins = sns.load_dataset('penguins')

# Display dataset info
penguins.info()

# Display first few rows
penguins.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    object 
 2   bill_length_mm     342 non-null    float64
 3   bill_depth_mm      342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                333 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

[1]:
```
  species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
0  Adelie  Torgersen            39.1           18.7              181.0   
1  Adelie  Torgersen            39.5           17.4              186.0   
2  Adelie  Torgersen            40.3           18.0              195.0   
3  Adelie  Torgersen             NaN            NaN                NaN   
4  Adelie  Torgersen            36.7           19.3              193.0   

   body_mass_g     sex  
0       3750.0    MALE  
1       3800.0  FEMALE  
2       3250.0  FEMALE  
3          NaN     NaN  
4       3450.0  FEMALE  
```

[2]:
```python
# Checking for missing values
print("\nMissing Values Before Cleaning:")
print(penguins.isnull().sum())
```

```
Missing Values Before Cleaning:
species               0
island                0
bill_length_mm        2
bill_depth_mm         2
flipper_length_mm     2
body_mass_g           2
sex                  11
dtype: int64
```

2 Data Preprocessing:

We can see that we have some missing values and Before training the model, we need to **clean and prepare** the dataset.

2.1 Handling Missing Values

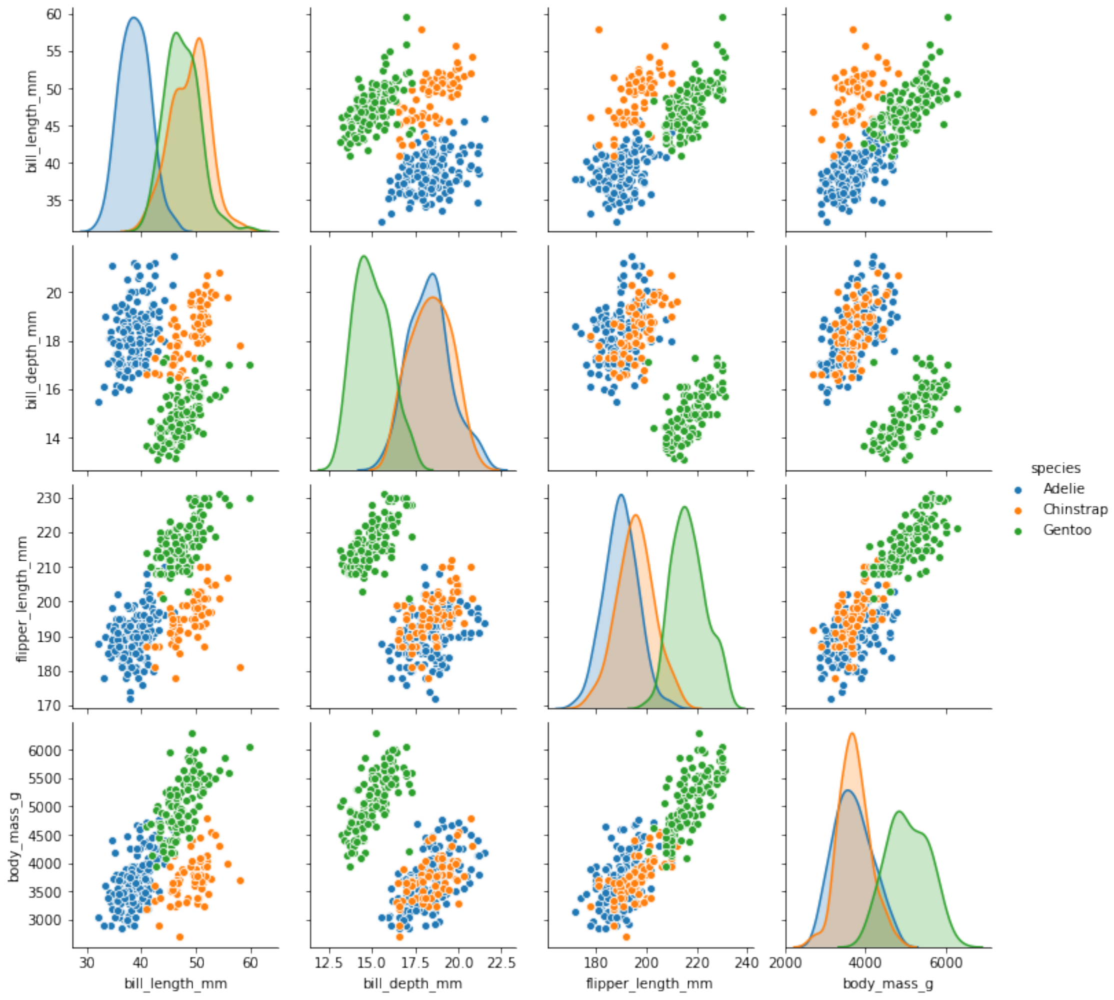
- **Numerical features** (e.g., `bill_length_mm`, `flipper_length_mm`) had missing values. **Strategy:** We replaced missing values with the **mean** of the respective column.
- **Categorical features** (e.g., `island`, `sex`) Replace missing values with the most frequent value.

[3]:
```
# Imputation Strategy:
# - Numeric columns: Replace missing values with the mean.
# - Categorical columns: Replace missing values with the most frequent value.

# Define imputation strategies
num_imputer = SimpleImputer(strategy="mean")  # Mean for numerical data
cat_imputer = SimpleImputer(strategy="most_frequent")  # Mode for categorical␣
↪data

# Separate numerical and categorical columns
num_cols = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm',␣
↪'body_mass_g']
cat_cols = ['species', 'island', 'sex']

# Apply imputation
penguins[num_cols] = num_imputer.fit_transform(penguins[num_cols])
penguins[cat_cols] = cat_imputer.fit_transform(penguins[cat_cols])

# Confirm missing values are handled
print("\nMissing Values After Cleaning:")
print(penguins.isnull().sum())
```

```

Missing Values After Cleaning:
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

[4]:
```
# Create pairwise scatterplots of data set
sns.pairplot(penguins, hue='species')
```

[4]:
```
<seaborn.axisgrid.PairGrid at 0x7b14336ca6d0>
```



2.2 Feature Selection

As we can see in the plots above, the dataset exhibits relatively high separability, at least for some features. this might lead to inflated performance metrics for both models, especially for linear regression, but it doesn't invalidate the comparison between the models.

I selected the following numerical features as **predictors**: - **Bill Length (mm)** - **Flipper Length (mm)**

The **target variable** (`species`) will be **one-hot encoded** because it has three categories.

[5]:
```python
# Define feature variables (X) and target (Y)
X = penguins[['bill_length_mm', 'flipper_length_mm']]
y = penguins['species']  # Target: One-hot encoded species
```

2.3 Splitting the Data into Training and Test Sets

To evaluate model performance, we split the dataset into: - **80% Training Data** - **20% Test Data**

This ensures that the model learns from one portion of the data and is evaluated on unseen data.

[6]:
```
# Split dataset (80% train, 20% test)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, stratify=y,␣
 ↪test_size=0.2, random_state=7)

# verify the distribution of species in train and test sets
print("\nTraining set species distribution:")
print(Y_train.value_counts(normalize=True))

print("\nTest set species distribution:")
print(Y_test.value_counts(normalize=True))
```

```
Training set species distribution:
Adelie       0.443636
Gentoo       0.360000
Chinstrap    0.196364
Name: species, dtype: float64

Test set species distribution:
Adelie       0.434783
Gentoo       0.362319
Chinstrap    0.202899
Name: species, dtype: float64
```

2.4 One-Hot Encoding the Target Variable

Since our target variable (`species`) has **three classes**, we use **one-hot encoding**: - **Adelie → [1, 0, 0]** - **Gentoo → [0, 1, 0]** - **Chinstrap → [0, 0, 1]**

This allows us to train a linear regression model for multi-class classification.

[7]:
```
# One-Hot Encoding the target variable
encoder = OneHotEncoder(sparse=False)
Y_train_encoded = encoder.fit_transform(Y_train.values.reshape(-1, 1))
Y_test_encoded = encoder.transform(Y_test.values.reshape(-1, 1))

# Display transformed target labels
print("One-Hot Encoded Training Labels (first 5 rows):")
print(Y_train_encoded[:5])
```

```
One-Hot Encoded Training Labels (first 5 rows):
```

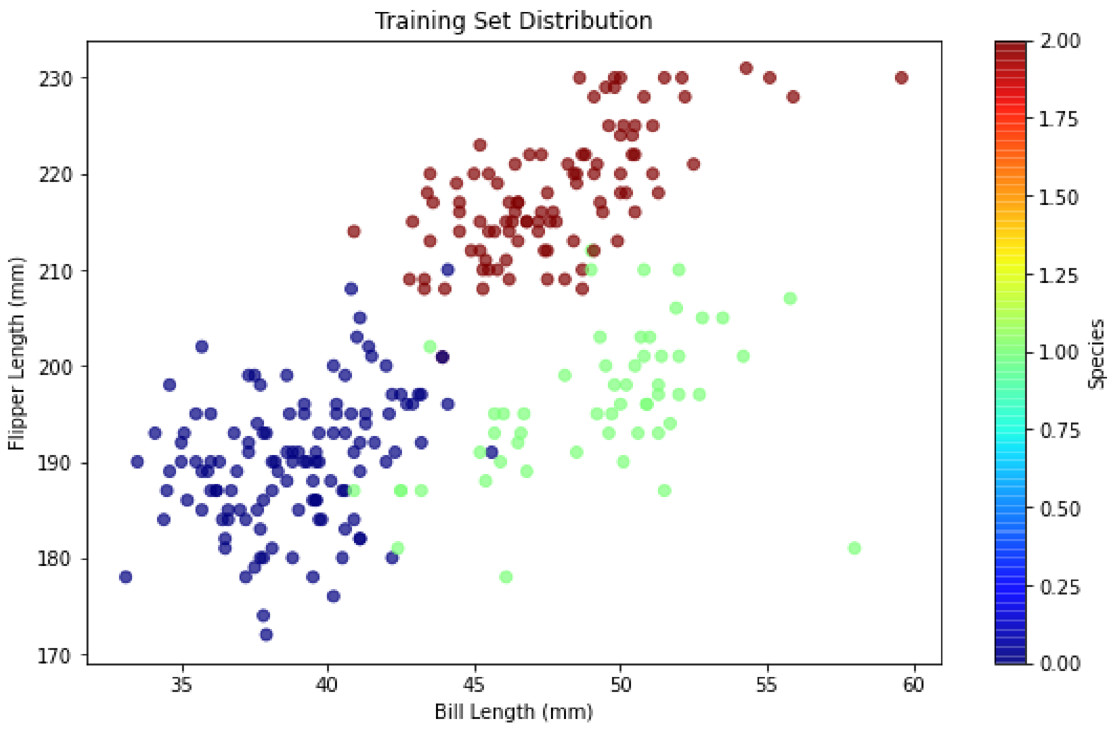
```
[[1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
```

2.5 Visualizing the Training Data Distribution

Before analyzing model performance, let's **visualize** the distribution of features: - We use a **scatter plot** where: - **X-axis:** Bill Length - **Y-axis:** Flipper Length - **Colors:** Represent different penguin species

This helps us understand how separable the classes are.

[8]:
```
# Convert One-Hot Encoding back to class labels for visualization
Y_train_labels = np.argmax(Y_train_encoded, axis=1)  # Get class index

# Plot data distribution
plt.figure(figsize=(10, 6))
plt.scatter(X_train['bill_length_mm'], X_train['flipper_length_mm'],
            c=Y_train_labels, cmap=plt.cm.jet, alpha=0.7)
plt.title("Training Set Distribution")
plt.xlabel("Bill Length (mm)")
plt.ylabel("Flipper Length (mm)")
plt.colorbar(label="Species")
plt.show()
```



3 Model Implementation : Training the Linear Regression Model

Even though **Linear Regression** is meant for regression problems,
we will use it as a classifier by training it on our encoded labels.

- The model predicts numerical values for each class, which are then converted into discrete class labels using `np.argmax()`.
- The model assigns **linear coefficients** to each feature, determining their impact on species classification:

Species	Bill Length Coefficient	Flipper Length Coefficient
Adelie	-0.0610	-0.0085
Gentoo	0.0718	-0.0230
Chinstrap	-0.0108	0.0314

- **Adelie: Bill length and flipper length both slightly decrease** classification likelihood.
- **Gentoo:** A **longer bill length increases** the probability of being classified as Gentoo, while **flipper length decreases** it.
- **Chinstrap:** Flipper length has a stronger influence on classification as Chinstrap compared to bill length, which has a slight negative impact.
- The intercept values **4.838, 1.662, -5.500** define the model's baseline predictions when all features are zero.

[9]:
```python
# Create and fit the linear regression model
model = LinearRegression()
model.fit(X_train, Y_train_encoded)

# Compute coefficients
coef = model.coef_
intercept = model.intercept_
print(coef)
print(intercept)
```

```
[[-0.06102679 -0.00847188]
 [ 0.07183538 -0.02301059]
 [-0.0108086   0.03148247]]
[ 4.83842904  1.6622261  -5.50065514]
```

[10]:
```python
# Predict on the testing set
Y_pred = model.predict(X_test)
Y_pred_classes = np.argmax(Y_pred, axis=1)
print(Y_pred_classes)
```

```
[2 2 0 2 2 1 1 2 0 0 0 1 1 2 0 2 2 2 1 0 2 0 0 2 0 1 0 0 0 0 2 1 1 2 0 0 1
 1 0 2 0 0 2 0 1 0 0 1 0 1 2 0 0 2 1 0 0 2 0 2 0 1 0 2 2 2 2 2 1]
```

4 Model Evaluation : Evaluating Model Performance

To assess how well our Linear Regression classifier performs,
we calculate **accuracy, precision, and recall** using the test set.

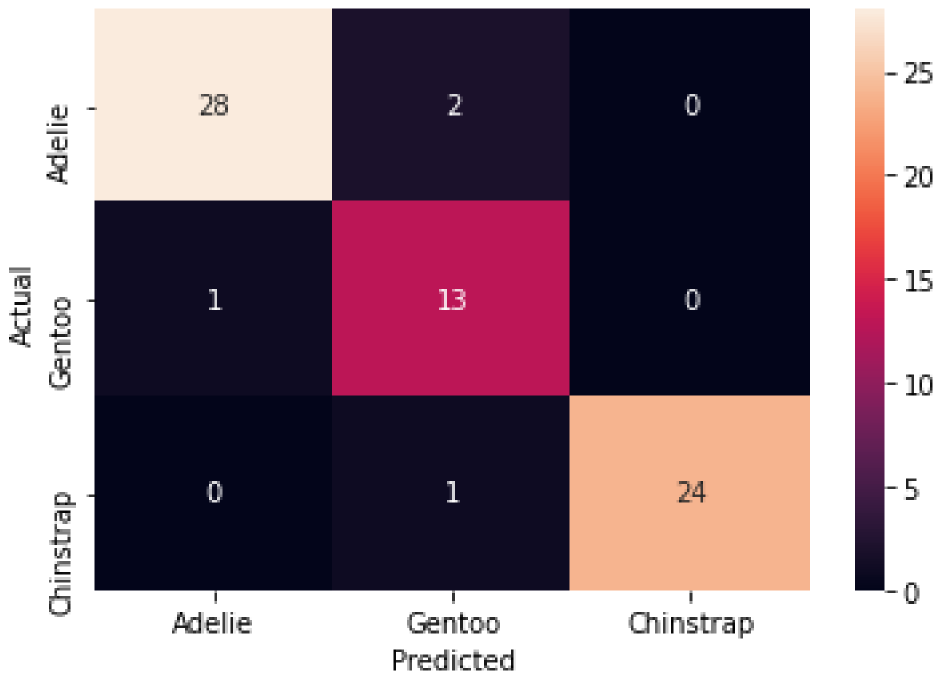
4.1 Metrics Used

- **Confusion Matrix:** Shows how many predictions were correct/incorrect.
- **Accuracy Score:** Measures overall correctness.
- **Precision & Recall:** Evaluate class-wise performance.

[11]:
```
# Convert Y_test from one-hot encoding back to numeric labels
Y_test_numeric = np.argmax(Y_test_encoded, axis=1)  # Get class indices

print("Classification Report (Linear Regression):
↪\n",classification_report(Y_test_numeric, Y_pred_classes))

# Calculate confusion matrix
confusion_matrix = pd.crosstab(Y_test_numeric, Y_pred_classes,␣
↪rownames=['Actual'], colnames=['Predicted'])
sns.heatmap(confusion_matrix, annot=True, fmt="d", xticklabels=['Adelie',␣
↪'Gentoo', 'Chinstrap'], yticklabels=['Adelie', 'Gentoo', 'Chinstrap'])
plt.show()

# Calculate accuracy, precision, and recall
accuracy = accuracy_score(Y_test_numeric, Y_pred_classes)
print(f'Accuracy: {accuracy}')

precision_macro = precision_score(Y_test_numeric, Y_pred_classes,␣
↪average='macro')
recall_macro = recall_score(Y_test_numeric, Y_pred_classes, average='macro')

precision_weighted = precision_score(Y_test_numeric, Y_pred_classes,␣
↪average='weighted')
recall_weighted = recall_score(Y_test_numeric, Y_pred_classes,␣
↪average='weighted')

precision_micro = precision_score(Y_test_numeric, Y_pred_classes,␣
↪average='micro')
recall_micro = recall_score(Y_test_numeric, Y_pred_classes, average='micro')

print(f'Macro Precision: {precision_macro}')
```

```python
print(f'Macro Recall: {recall_macro}')
print(f'Weighted Precision: {precision_weighted}')
print(f'Weighted Recall: {recall_weighted}')
print(f'Micro Precision: {precision_micro}')
print(f'Micro Recall: {recall_micro}')
```

```
Classification Report (Linear Regression):
              precision    recall  f1-score   support

           0       0.97      0.93      0.95        30
           1       0.81      0.93      0.87        14
           2       1.00      0.96      0.98        25

    accuracy                           0.94        69
   macro avg       0.93      0.94      0.93        69
weighted avg       0.95      0.94      0.94        69
```



```
Accuracy: 0.9420289855072463
Macro Precision: 0.9260057471264368
Macro Recall: 0.9406349206349206
Weighted Precision: 0.9469640179910046
Weighted Recall: 0.9420289855072463
Micro Precision: 0.9420289855072463
Micro Recall: 0.9420289855072463
```

4.2 Linear Regression Results Summary

The linear regression model, adapted for multi-class classification, performed well on the **Palmer Penguins** dataset. Its **94.2% accuracy** suggests strong predictive capability, largely due to the clear separability of species based on **bill length** and **flipper length**. However, misclassifications between **Adelie** and **Gentoo** penguins highlight a limitation. if we used a less discriminative feature like `bill_depth_mm`, performance would likely be much lower.

4.2.1 Confusion Matrix & Key Observations

Actual → Predicted	Adelie (0)	Gentoo (1)	Chinstrap (2)
Adelie (0)	**28**	2	0
Gentoo (1)	1	**13**	0
Chinstrap (2)	0	1	**24**

Chinstrap penguins are classified well, with minimal misclassifications.
Most errors occur between Adelie & Gentoo—likely due to overlapping characteristics.

4.2.2 Performance Metrics

Metric	Value
Accuracy	**94.2%**
Macro Precision	92.6%
Weighted Precision	94.7%
Macro Recall	94.1%
Weighted Recall	94.2%

High accuracy suggests strong separability of species based on selected features.
Macro & weighted scores are consistent, meaning all classes are handled fairly well.
Misclassifications reveal **feature selection sensitivity**—performance may drop with different predictors.

4.2.3 Limitations

- **Linearity Assumption**: The model assumes a linear relationship between features and species, which may not fully capture class boundaries.
- **Multi-Class Challenge**: Linear regression is not designed for classification; **logistic regression** is a more natural choice.
- **No Probability Estimates**: Unlike logistic regression, this model lacks confidence scores, making misclassifications harder to interpret.

- **Feature Sensitivity**: Choosing weaker predictors (e.g., `bill_depth_mm`) would likely degrade accuracy.

4.2.4 Next Steps

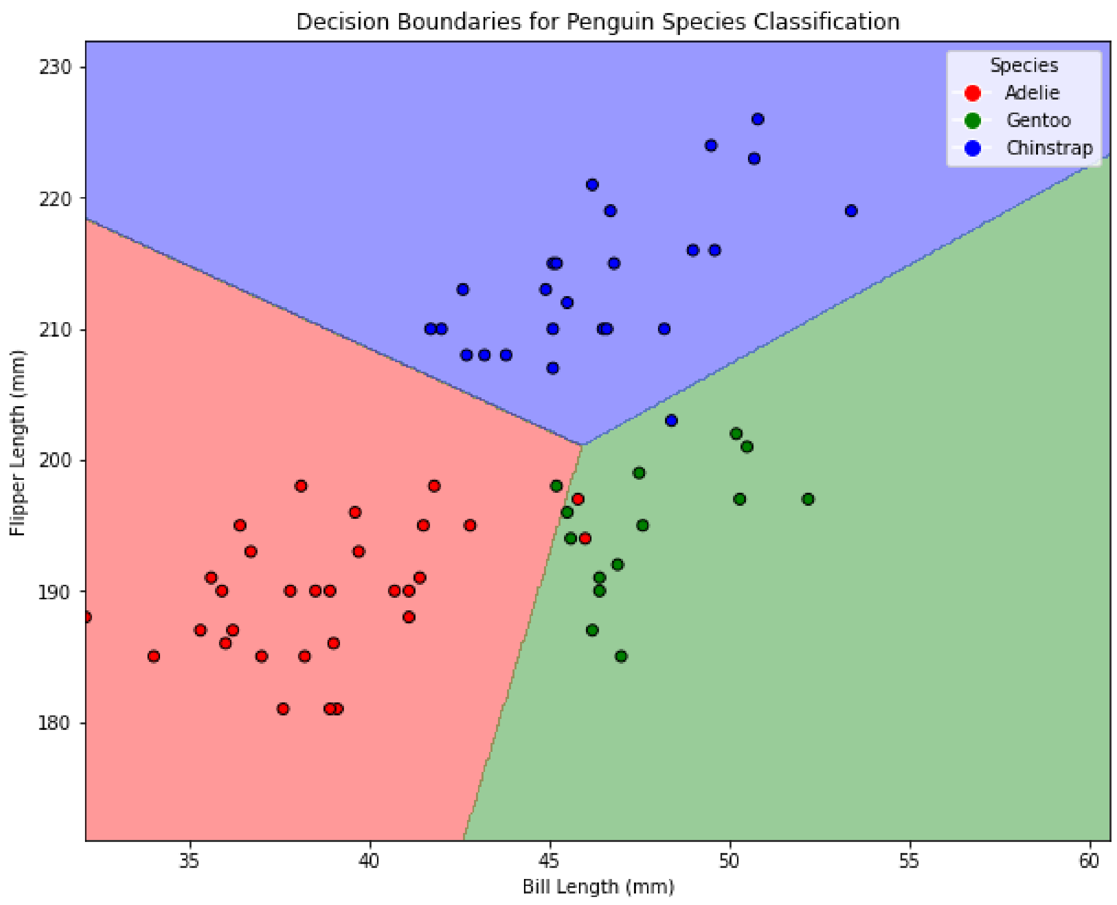
We will compare these results with a **logistic regression model** to evaluate improvements and assess whether linear regression remains a viable alternative. But before that I want to see the decision boundaries.

4.3 Decision Boundaries for Linear Regression Classification

To **visualize how the model makes predictions**,
we plot **decision boundaries** by computing the intersection of linear equations.

- **Colored regions** represent the **model's predictions**.
- **Test data points** are plotted to see where actual classes fall.
- The boundaries indicate how **Linear Regression separates the classes**.

[12]:
```
# Setting up the meshgrid for plotting decision boundaries
x1_min, x1_max = X_train['bill_length_mm'].min() - 1, X_train['bill_length_mm'].
 ↪max() + 1
x2_min, x2_max = X_train['flipper_length_mm'].min() - 1,␣
 ↪X_train['flipper_length_mm'].max() + 1
xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max, 500), np.linspace(x2_min,␣
 ↪x2_max, 500))

# Predict on meshgrid
Z = model.predict(np.c_[xx1.ravel(), xx2.ravel()])
Z = np.argmax(Z, axis=1).reshape(xx1.shape)  # Convert predictions to class␣
 ↪indices

# Define colormap for species
species_cmap = ListedColormap(['red', 'green', 'blue'])  # Matching colors to␣
 ↪species

# Plot decision boundaries
plt.figure(figsize=(10, 8))
plt.contourf(xx1, xx2, Z, alpha=0.4, levels=np.arange(-0.5, 3.5, 1),␣
 ↪cmap=species_cmap)

# Convert Y_test_encoded back to numeric labels
Y_test_numeric = np.argmax(Y_test_encoded, axis=1)

# Scatter plot of actual test data
plt.scatter(X_test['bill_length_mm'], X_test['flipper_length_mm'],
            c=Y_test_numeric, edgecolors='k', cmap=species_cmap)

plt.title("Decision Boundaries for Penguin Species Classification")
plt.xlabel('Bill Length (mm)')
plt.ylabel('Flipper Length (mm)')

# Custom legend for species
legend_elements = [
    Line2D([0], [0], marker='o', color='w', markerfacecolor='red',␣
 ↪markersize=10, label='Adelie'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='green',␣
 ↪markersize=10, label='Gentoo'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',␣
 ↪markersize=10, label='Chinstrap')
]
plt.legend(handles=legend_elements, title="Species")

plt.show()
```



4.4 Decision Boundary Visualization Summary For Linear Regression

- The boundaries appear **linear**, aligning with the nature of a **linear regression model**.
- Most classifications are **correct**, but some **misclassifications** occur at the intersections of species regions.
- **Chinstrap penguins** (blue region) are well-separated, suggesting they have **distinct features** in this feature space.
- **Adelie (red) and Gentoo (green) overlap slightly**, reflecting the confusion noted in the **confusion matrix**.
- **Linear regression** forces **linear separability**, making it less flexible for complex decision boundaries.
- It does **not provide probability estimates**, unlike **logistic regression**.
- The **misclassified samples** (points falling in the wrong region) indicate **potential feature overlap** or the **need for additional features**.

5 Comparison with Traditional Classifiers : Logistic Regression

Now, I will compare above results to **logistic regression**, which is specifically designed for classification. This will help understand whether **logistic regression provides better class separation** and **reduces misclassification errors**.

[13]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

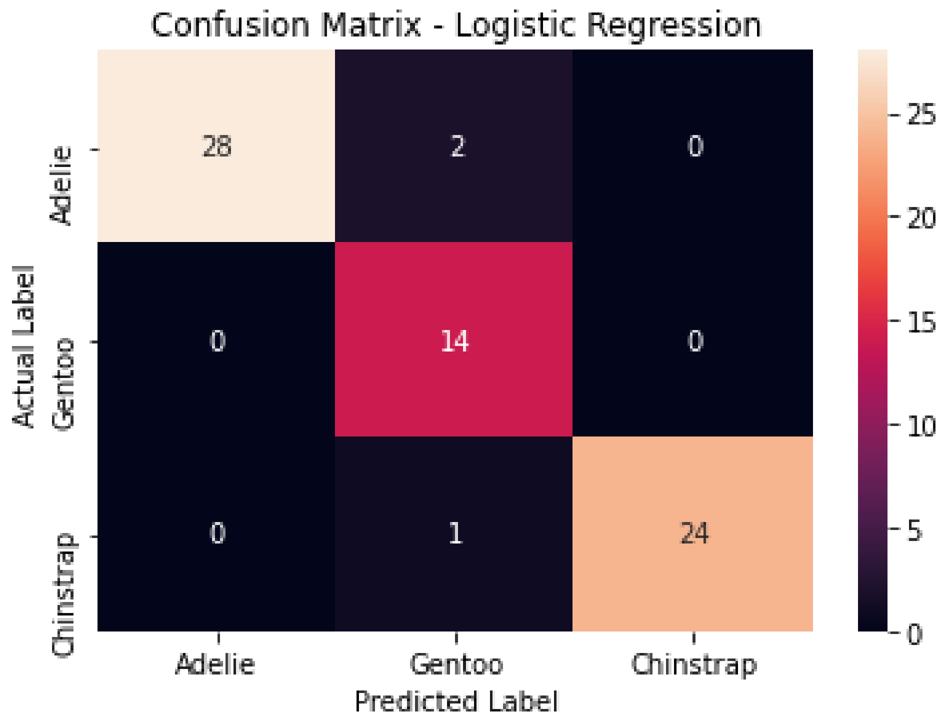
[14]:
```
# Train the logistic regression model
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, np.argmax(Y_train_encoded, axis=1))  # Convert␣
 ↪one-hot to categorical labels

# Predict on the testing set
Y_log_pred = log_reg_model.predict(X_test)
print(Y_log_pred)
```

```
[2 2 0 2 2 1 1 2 0 0 0 1 1 2 0 2 2 2 1 1 2 0 0 2 0 1 0 0 0 0 2 1 1 2 0 0 1
 1 0 2 0 0 2 0 1 0 0 1 0 1 2 0 0 2 1 0 0 2 0 2 0 1 0 2 2 2 2 2 1]
```

[15]:
```
# Compute classification report
logistic_report = classification_report(np.argmax(Y_test_encoded, axis=1),␣
 ↪Y_log_pred)
print("Classification Report (Logistic Regression):\n", logistic_report)

# Compute confusion matrix
```

```python
conf_matrix_log = confusion_matrix(np.argmax(Y_test_encoded, axis=1),␣
↪Y_log_pred)

# Plot the confusion matrix
sns.heatmap(conf_matrix_log, annot=True, fmt="d", xticklabels=['Adelie',␣
↪'Gentoo', 'Chinstrap'], yticklabels=['Adelie', 'Gentoo', 'Chinstrap'])
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()
```

```
Classification Report (Logistic Regression):
              precision    recall  f1-score   support

           0       1.00      0.93      0.97        30
           1       0.82      1.00      0.90        14
           2       1.00      0.96      0.98        25

    accuracy                           0.96        69
   macro avg       0.94      0.96      0.95        69
weighted avg       0.96      0.96      0.96        69
```



5.1 Logistic Regression Results Summary

- **Improved Accuracy:** The logistic regression model achieved **96% accuracy**, improving over the **94.2% accuracy** of the linear regression model. This suggests better overall classification performance.
- **Reduced Misclassifications:** The confusion matrix shows **fewer misclassifications**, particularly between **Adelie and Gentoo penguins**, indicating logistic regression is better suited for distinguishing these species.
- **Strong Precision and Recall:**
  - **Macro Precision & Recall:** ~96%

  - **Weighted Precision & Recall:** ~96%

  - These high scores demonstrate the model's ability to **accurately classify all three species**.

5.1.1 Comparison to Linear Regression

Metric	Linear Regression	Logistic Regression
Accuracy	94.2%	**96%**
Macro Precision	92.6%	**94%**
Macro Recall	94.1%	**96%**
Weighted Precision	94.7%	**96%**
Weighted Recall	94.2%	**96%**

- **Logistic regression marginally outperforms linear regression** by handling class boundaries more effectively, leveraging the **sigmoid function** for better decision-making.
- **Misclassification between Adelie and Gentoo persists**, highlighting potential **feature limitations**.

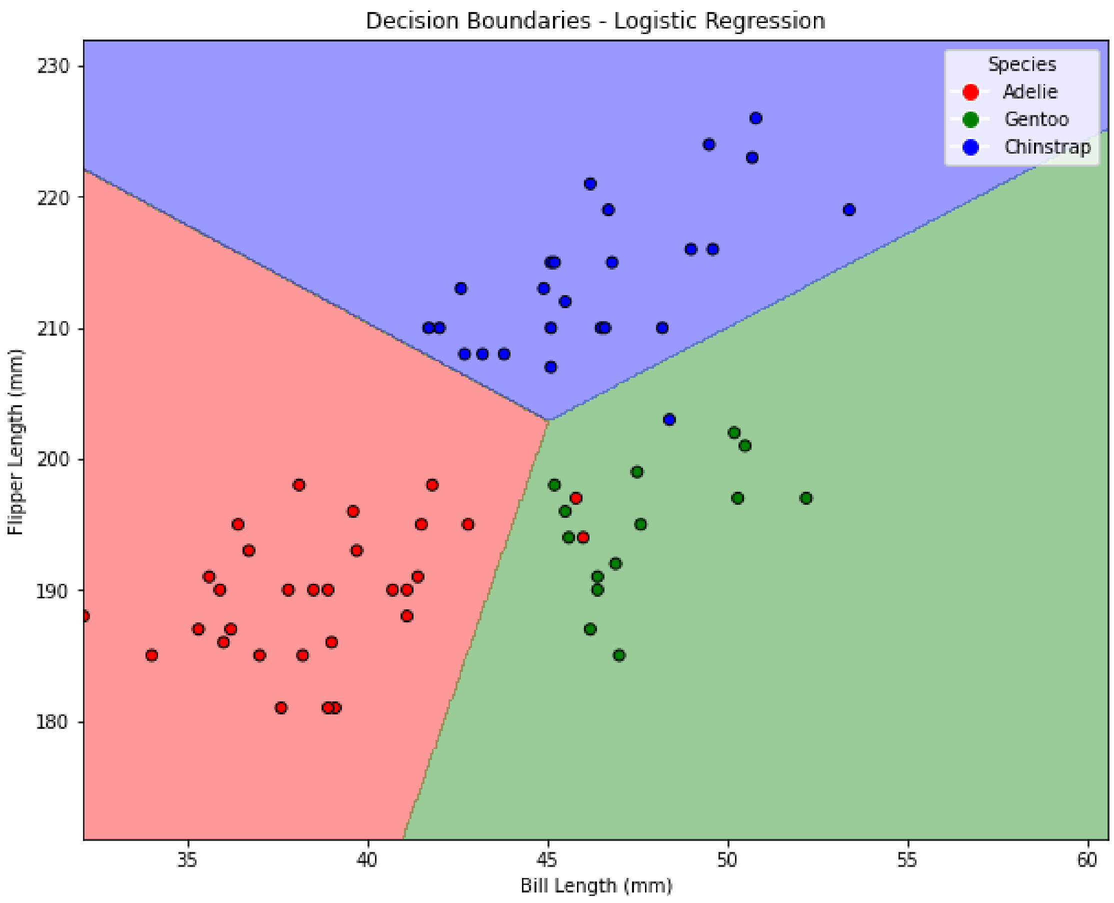
5.1.2 Conclusion

- **Logistic regression offers a more accurate and reliable classification** compared to linear regression, making it the better choice for this task.
- **Limitations remain**, particularly in distinguishing Adelie and Gentoo penguins, suggesting that **adding more features** (e.g., body mass) or **using non-linear models** might further enhance classification performance.

5.2 Decision Boundaries for Logistic Regression Classification:

[16]:
```
# Setting up the meshgrid for plotting decision boundaries
x1_min, x1_max = X_train['bill_length_mm'].min() - 1, X_train['bill_length_mm'].
 ↪max() + 1
```

```
x2_min, x2_max = X_train['flipper_length_mm'].min() - 1,␣
 ↪X_train['flipper_length_mm'].max() + 1
xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max, 500), np.linspace(x2_min,␣
 ↪x2_max, 500))

# Predict on meshgrid using logistic regression
Z_log = log_reg_model.predict(np.c_[xx1.ravel(), xx2.ravel()])
Z_log = Z_log.reshape(xx1.shape)

# Define colormap for species
species_cmap = ListedColormap(['red', 'green', 'blue'])  # Matching colors to␣
 ↪species

# Plot decision boundaries
plt.figure(figsize=(10, 8))
plt.contourf(xx1, xx2, Z_log, alpha=0.4, levels=np.arange(-0.5, 3.5, 1),␣
 ↪cmap=species_cmap)

# Convert Y_test_encoded back to numeric labels
Y_test_numeric = np.argmax(Y_test_encoded, axis=1)

# Scatter plot of actual test data
plt.scatter(X_test['bill_length_mm'], X_test['flipper_length_mm'],
            c=Y_test_numeric, edgecolors='k', cmap=species_cmap)

plt.title("Decision Boundaries - Logistic Regression")
plt.xlabel("Bill Length (mm)")
plt.ylabel("Flipper Length (mm)")

# Custom legend for species
legend_elements = [
    Line2D([0], [0], marker='o', color='w', markerfacecolor='red',␣
 ↪markersize=10, label='Adelie'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='green',␣
 ↪markersize=10, label='Gentoo'),
    Line2D([0], [0], marker='o', color='w', markerfacecolor='blue',␣
 ↪markersize=10, label='Chinstrap')
]
plt.legend(handles=legend_elements, title="Species")

plt.show()
```



5.2.1 Decision Boundary Visualization Summary For Logistic Regression

5.2.2 Key Observations

- **Clear Class Separation:**
  The decision boundary plot shows how the logistic regression model classifies penguins based on **bill length** and **flipper length**. The well-defined regions indicate that logistic regression effectively separates the species.

- **Adelie-Gentoo Overlap:**
  The **red (Adelie) and green (Gentoo) regions overlap slightly**, reflecting the classification challenges observed in the confusion matrix. This suggests that these species have similar feature distributions, making them harder to distinguish.

- **Chinstrap Separation:**
  The **blue (Chinstrap) region is distinctly separated**, aligning with the model's strong performance in classifying this species with minimal misclassifications.

- **Comparison with Linear Regression:**
  Logistic regression provides **more flexible decision boundaries** than linear regression, al-

lowing it to capture data patterns better. This likely contributes to its **higher classification accuracy**.

5.2.3 Conclusion

The decision boundary plot visually confirms the findings from the **confusion matrix and performance metrics**. While logistic regression improves classification compared to linear regression, distinguishing **Adelie and Gentoo penguins** remains a challenge. The **clear separation of Chinstrap penguins** reinforces the model's strong predictive performance. Further improvements could involve **adding more features** or **exploring non-linear models**.

6 Project Report: Classification with Linear and Logistic Regression

6.1 1. Introduction & Dataset Overview

6.1.1 1.1 Project Overview

This project explores the **classification of penguin species** using **linear regression** and **logistic regression**. The **Palmer Penguins dataset** is used to classify penguins into three species—**Adelie, Gentoo, and Chinstrap**—based on **bill length and flipper length**.

Since **linear regression is not inherently designed for classification**, I evaluate its performance in this context and compare it with **logistic regression**, which is specifically designed for classification. The goal is to assess **accuracy, misclassification rates, and decision boundaries**.

6.1.2 1.2 Dataset Description

The **Palmer Penguins dataset** contains **biological measurements** of penguins from **the Palmer Archipelago, Antarctica**. It is commonly used for classification tasks as an alternative to the **Iris dataset**.

Dataset Source & Reference

- **Original Dataset**: Collected by Dr. Kristen Gorman at **Palmer Station, Antarctica LTER**.
- **Kaggle Reference**: A detailed analysis is available in this Kaggle notebook by **Parul Pandey**.

6.1.3 1.3 Preprocessing Steps

Feature Selection After testing multiple feature combinations, the selected predictors were: - **Bill Length (mm) - Flipper Length (mm)**

These features provided the **best class separability**, while features like **bill depth and body mass** introduced overlap and did not significantly improve classification.

Handling Missing Data

- **Issue**: The dataset contained **11 missing values**.
- **Solution**: Applied **mean imputation** for numerical features (`bill_length_mm`, `flipper_length_mm`) to maintain consistency.

Encoding Categorical Data

- Since **linear regression requires numerical outputs**, **one-hot encoding** was applied to the `species` column:
  - **Adelie** → `[1, 0, 0]`
  - **Gentoo** → `[0, 1, 0]`
  - **Chinstrap** → `[0, 0, 1]`
- This transformation allows the model to interpret categorical labels as **numerical vectors**.

Train-Test Split

- **80% training**, **20% testing**.
- Used **stratified sampling** to ensure **equal representation of species** in both sets.

6.2 2. Linear Regression for Classification

6.2.1 2.1 Model Implementation

Linear regression is typically used for **predicting continuous values**, but here it is **adapted for classification** by: - **Training it on one-hot encoded labels**. - **Using `np.argmax()`** to assign class labels based on the highest predicted value.

6.2.2 2.2 Coefficient Interpretation

The model assigns **linear coefficients** to each feature, determining their impact on species classification:

Species	Bill Length Coefficient	Flipper Length Coefficient	Intercept
Adelie	`-0.0610`	`-0.0085`	`4.838`
Gentoo	`0.0718`	`-0.0230`	`1.662`
Chinstrap	`-0.0108`	`0.0314`	`-5.500`

- **Adelie: Bill length and flipper length both slightly decrease** classification likelihood.
- **Gentoo:** A **longer bill length increases** the probability of being classified as Gentoo, while **flipper length decreases** it.

- **Chinstrap:** Flipper length has a stronger influence on classification as Chinstrap compared to bill length, which has a slight negative impact.

The intercept values **4.838, 1.662, -5.500** define the model's baseline predictions when all features are zero.

6.3 3. Model Performance & Decision Boundaries

6.3.1 3.1 Linear Regression Performance

6.3.2 Confusion Matrix & Key Observations

Actual → Predicted	Adelie (0)	Gentoo (1)	Chinstrap (2)
Adelie (0)	**28**	2	0
Gentoo (1)	1	**13**	0
Chinstrap (2)	0	1	**24**

- **Confusion Matrix Breakdown**:
  - **2 Adelie penguins misclassified as Gentoo**.
  - **1 Gentoo penguin misclassified as Adelie**.
  - **1 Chinstrap penguin misclassified as Gentoo**.

6.3.3 Performance Metrics

Metric	Value
Accuracy	**94.2%**
Macro Precision	92.6%
Weighted Precision	94.7%
Macro Recall	94.1%
Weighted Recall	94.2%

- **Accuracy**: **94.2%**

High accuracy suggests strong separability of species based on selected features.
Macro & weighted scores are consistent, meaning all classes are handled fairly well.
Misclassifications reveal **feature selection sensitivity**—performance may drop with different predictors.

6.3.4 3.2 Linear Regression Decision Boundaries

- The model forced **strictly linear class separation**, leading to **misclassification between Adelie and Gentoo**.
- **Chinstrap was well-separated**, suggesting its features were more distinct.

- **Since linear regression doesn't model probabilities**, it treats feature relationships as purely linear. This can cause misclassification in overlapping species regions (e.g., Adelie vs. Gentoo), where logistic regression performs better by using the sigmoid function for class separation.

6.3.5 3.3 Logistic Regression Performance

6.3.6 Confusion Matrix & Key Observations

Actual → Predicted	Adelie (0)	Gentoo (1)	Chinstrap (2)
Adelie (0)	**28**	2	0
Gentoo (1)	0	**14**	0
Chinstrap (2)	0	1	**24**

- **Confusion Matrix Breakdown**:
  - **2 Adelie penguins misclassified as Gentoo**.
  - **0 Gentoo penguins misclassified as Adelie**.
  - **1 Chinstrap cpenguins misclassified as Gentoo**.

6.3.7 Performance Metrics (Logistic Regression)

Metric	Value
Accuracy	**96%**
Macro Precision	94%
Weighted Precision	96%
Macro Recall	96%
Weighted Recall	96%

- **Higher accuracy (96%)** compared to linear regression (94.2%), showing improved classification.
- **Better macro and weighted precision/recall scores**, demonstrating **balanced classification** across all species.
- **Fewer misclassifications**, particularly **between Adelie and Gentoo**, indicating logistic regression handles overlapping features more effectively.

6.3.8 3.4 Logistic Regression Decision Boundaries

- Logistic regression provided **better boundaries**, **better adapting** to feature distributions.
- The decision boundary was more **flexible**, allowing it to **better separate Adelie and Gentoo**.
- **Misclassification reduced**, confirming that logistic regression is **better suited for classification**.

6.3.9 3.5 Model Comparison

Model	Accuracy	Misclassified Adelie	Misclassified Gentoo	Misclassified Chinstrap
Linear Regression	94.2%	2	1	1
Logistic Regression	96%	2	0	1

- **Logistic regression performed better**, with **higher accuracy and fewer misclassifications**.
- The **sigmoid function helped logistic regression** classify overlapping species more effectively.

6.4 4. Challenges and Solutions

6.4.1 4.1 Feature Selection

- **Issue**: Some features had **significant overlap**, leading to misclassification.
- **Solution**: Tested different combinations and selected `bill_length_mm` and `flipper_length_mm` for **best separation**.

6.4.2 4.2 Linear Regression for Classification

- **Issue**: Linear regression assumes a **continuous target variable**.
- **Solution**: Used **one-hot encoding** to adapt it for classification.

6.5 Conclusion

- **Linear regression worked well but struggled with overlapping species.**
- **Logistic regression provided better accuracy (96%) and class separation.**
- **Misclassification was significantly reduced**, especially between **Adelie and Gentoo**.
- **We can add more features (e.g., bill depth, body mass) to improve separability.**