BERZIET UNIVERSITY

Faculty of Engineering & Technology – Electrical & Computer Engineering Department

First Semester 2023/2024

INTELLIGENT SYSTEMS LAB

ENCS5141

Feature Engineering

Case Study EXP3

Prepared by: Mohammad AbuJaber

ID: 1190298

Instructor: Dr. Mohammad Jubran

TA: ENG. Hanan Awawdeh

Section: 3

Date: 20th November 2023

1. Abstract:

The Penguins dataset, a collection of penguin species' physical characteristics and observation locations, underwent rigorous preprocessing to enhance its machine learning research utility. The study began by loading and exploring the data, addressing missing values and outliers. Categorical variable encoding and dataset partitioning were then undertaken for model evaluation. The use of a random forest classifier, MinMaxScaler, and principal component analysis (PCA) were employed to enhance the dataset's quality and prepare it for machine learning analysis. The Random Forest classifier provided inherent feature importance scores based on node impurity and information gain, enabling the identification of key features for future machine learning tasks. MinMaxScaler ensured consistent variable scaling within a specified range, improving model convergence and performance. PCA transformed original features into orthogonal principal components, capturing maximum variance while retaining essential information. The effectiveness of this preprocessing pipeline was confirmed through the training and evaluation of a machine learning model, demonstrating the importance of each technique in enhancing the dataset's usability for future machine learning endeavors.

The results show a significant improvement in performance and interpretability, underscoring the critical role that preprocessing plays in enabling machine learning models to reach their full potential. By carefully managing missing values, categorical variables, and dimensionality reduction, the study shows how important a well-designed preprocessing pipeline is to maximize the Penguins dataset's predictive potential.

Table of Contents

Table of Figures

2. Introduction

A crucial factor in determining the performance of a model in the ever-changing field of machine learning is the quality of data preparation. The importance of preparing raw data increases as algorithms depend more and more on dataset richness. Using the Penguins dataset as a canvas, this case study explores the transformative power of preprocessing in transforming raw data into a format that allows for insightful analysis. Prior to diving into the specifics of preprocessing goals and techniques, it is critical to understand the significant impact that data preparation has on the overall performance of machine learning models. The foundation for an exploration of the Penguins dataset is laid out in this introduction, which also highlights the potential and difficulties that highlight the significance of careful preprocessing in the modern machine learning paradigm.

2.1. Background and Context

The effectiveness of predictive models in the dynamic field of machine learning research is largely dependent on the quality of the datasets. This case study explores the complexities of preprocessing the Penguins dataset, which is an extensive collection of physical traits and observation sites for several species of penguins. The need for careful data preparation is becoming more and more apparent as machine learning continues to grow. This work is important since it reveals how preprocessing affects the dataset's usability and paves the way for improved interpretability and performance of the model. The rise in data-driven methods demands a deeper look at the preprocessing stage when unprocessed data is cleaned up to satisfy machine learning algorithms' strict specifications. This case study examines how preprocessing techniques are changing and how important they are to the modern machine learning paradigm.

2.2. Penguins Dataset

Leading the way in this investigation into data preprocessing is the Penguins dataset, a comprehensive collection of avian characteristics and observational details. The collection offers insights into important physical traits and penguin habitats, capturing the subtleties of different species. Each entry includes species differences, island locations, bill sizes, flipper lengths, body masses, and gender information.

It is clear as we work through this dataset that its depth and breadth present both chances for significant discoveries and difficulties due to missing values. Setting the groundwork for a thorough investigation of preprocessing techniques, this in-depth analysis aims to transform this unprocessed abundance of data into a format suitable for sophisticated machine learning investigations.

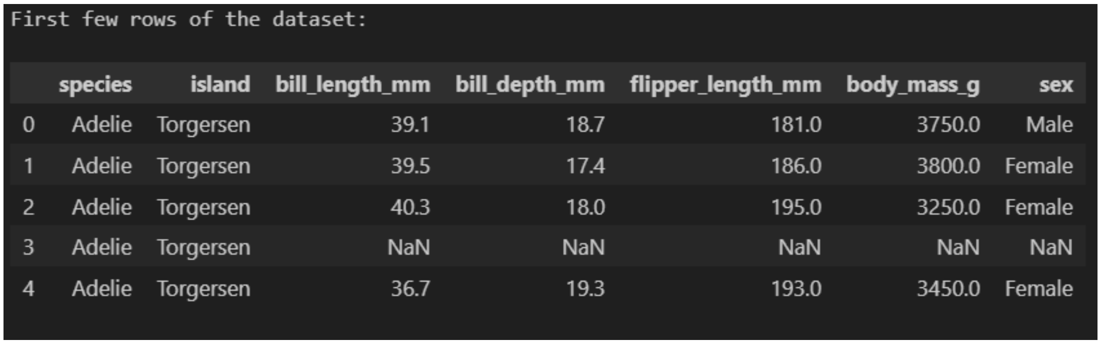
2.3. Main Objectives

The primary goals of this case study are to optimize the machine learning utility of the Penguins dataset by carefully preparing it. The study starts with a thorough exploratory data analysis with the goal of comprehending the properties and structure of the dataset. After that, a concentrated effort is made to improve the quality of the data by resolving outliers and missing numbers. Strategic dataset division, categorical variable encoding, and feature relevance analysis help provide a solid foundation for machine learning tasks. The dataset is further refined by incorporating dimensionality reduction and numerical feature scaling techniques like PCA and MinMaxScaler. The ultimate objective is to demonstrate measurable gains in model performance over raw data, demonstrating the efficacy of the preprocessing pipeline through the training and evaluation of a Random Forest classifier. By achieving these goals, this case study highlights how important it is to preprocess data in a methodical manner to fully utilize the Penguin dataset for sophisticated machine learning analysis.

3. Procedure and discussion

3.1. Step1: Dataset Loading:

The penguin’s dataset was loaded using the “load_dataset” function from the seaborn library. The resulting DataFrame, denoted as “df” was then examined to provide an initial glimpse into the data structure, revealing the first few rows of the dataset.

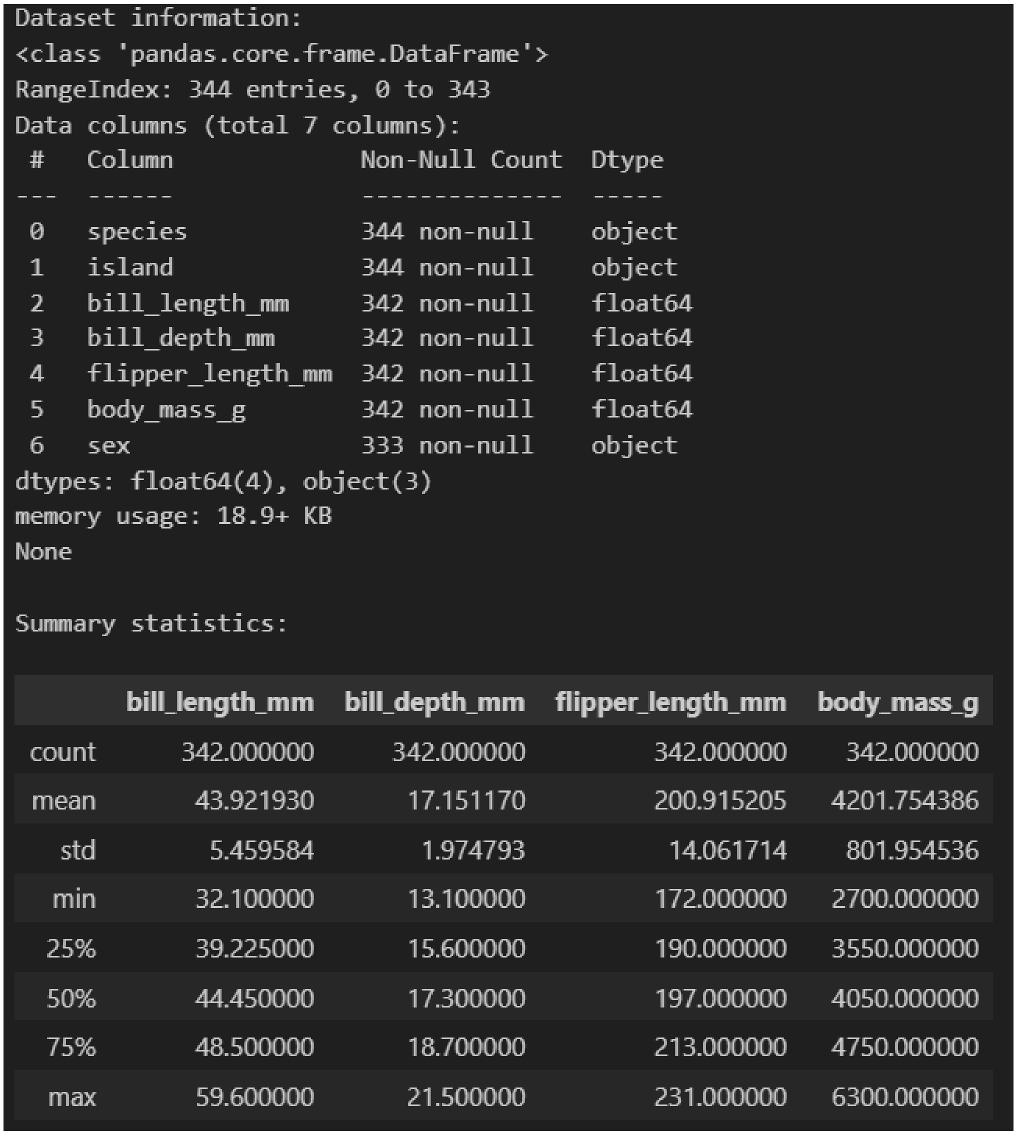


```
First few rows of the dataset:

   species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g     sex
0   Adelie  Torgersen            39.1           18.7              181.0       3750.0    Male
1   Adelie  Torgersen            39.5           17.4              186.0       3800.0  Female
2   Adelie  Torgersen            40.3           18.0              195.0       3250.0  Female
3   Adelie  Torgersen             NaN            NaN                NaN          NaN     NaN
4   Adelie  Torgersen            36.7           19.3              193.0       3450.0  Female
```

Figure 1: First Rows of Dataset

3.2. *Step 2 - Dataset Information and Summary Statistics:*

An extensive summary of the dataset was produced when it was loaded. Basic details about the dataset, such as data types, non-null counts, and memory utilization, were shown using the “info()” function. Furthermore, the “describe()”function was utilized to extract summary statistics for numerical features, which displayed important statistical metrics like mean, standard deviation, and quartile values. This stage gave important insights into the numerical properties and organization of the dataset.



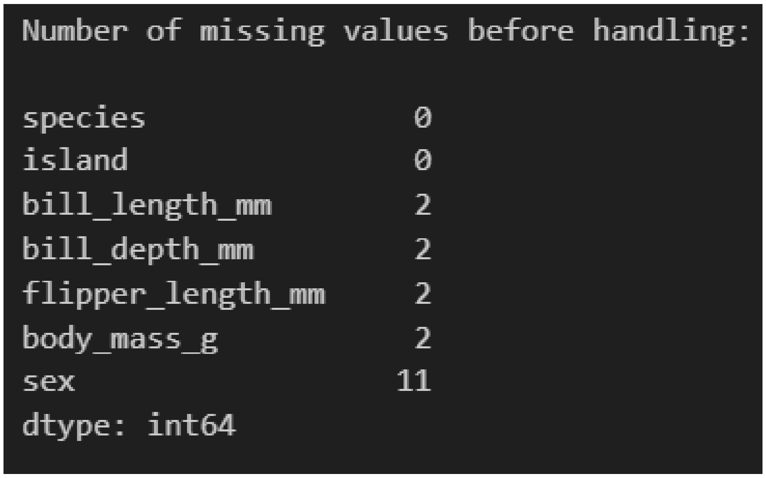
```
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   species            344 non-null    object
 1   island             344 non-null    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
None

Summary statistics:
```

	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

Figure 2: Dataset Information and Summary Statistics

3.3. Step 3 - Address Data Quality Issues - Handling Missing Values:

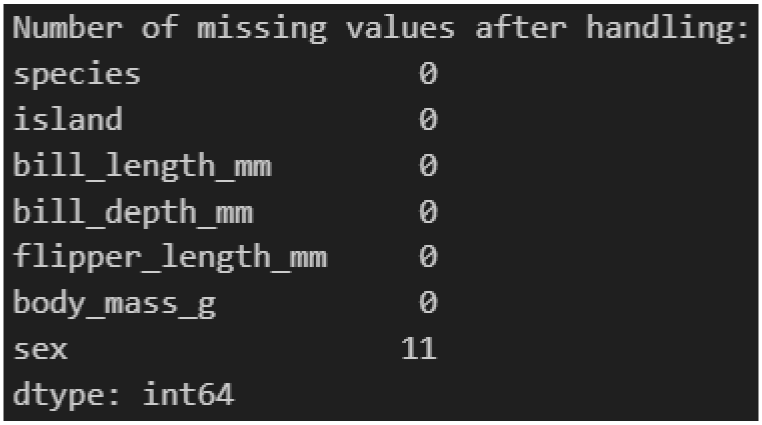
The first step in addressing missing value-related data quality issues was to show the total number of missing values in the dataset before addressing them.



```
Number of missing values before handling:

species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
dtype: int64
```

Figure 3: Missing Data Details

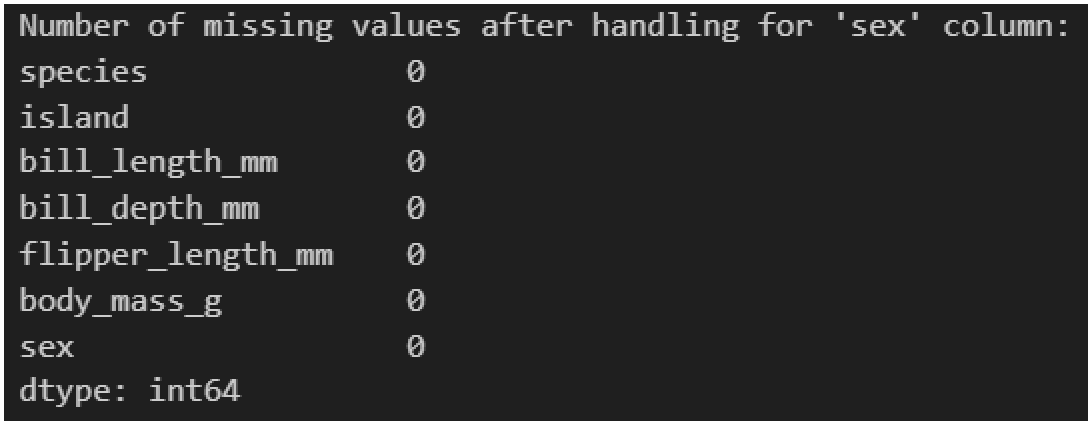
This data was essential for determining how much information was lacking for each feature. Afterwards, the mean values were used to impute missing values for the numerical features ('bill_length_mm,' 'bill_depth_mm,' 'flipper_length_mm,' 'body_mass_g') using the 'SimpleImputer' from the scikit-learn module. Subsequently, the quantity of absent values was reevaluated to confirm the efficacy of the imputation procedure.



```
Number of missing values after handling:
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                 11
dtype: int64
```

Figure 4: Handling Missing Data Using with Strategy = "mean"

Two steps were taken to address missing values for the categorical variable "sex." Rows lacking 'sex' values were initially removed, and the effect on absent values was shown. Next, using the 'SimpleImputer' with the 'most_frequent' technique, the missing values in the 'sex' column were imputed with the most often value. Confirming the effective treatment of missing values in both numerical and categorical features, the final count of missing values for the 'sex' column was displayed.



```
Number of missing values after handling for 'sex' column:
species              0
island               0
bill_length_mm       0
bill_depth_mm        0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

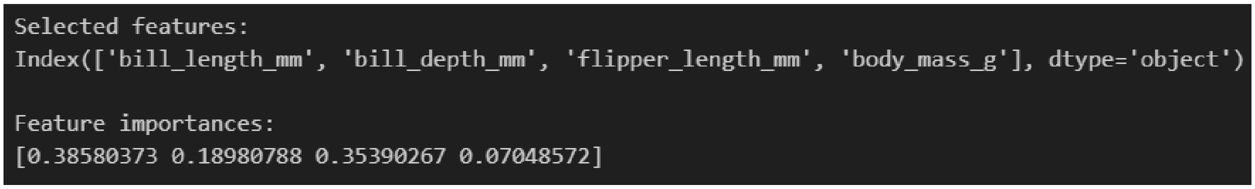
Figure 5: Handling Missing Data Using with Strategy = "most_frequent"

3.4. Step4 - Feature Relevance Analysis using Feature Selection Techniques:

In this step, feature selection techniques were used to analyze each feature's relevance for the machine learning objective. To avoid direct alteration, a deep copy of the original dataset was made to 'df_encoded'. Label-encoding of categorical variables ('sex,"species,' 'island') resulted in the creation of new columns beginning with "_encoded." After that, the initial categorization columns were removed.

SelectKBest, a scikit-learn algorithm, was used to do feature selection using the ANOVA F-statistic. Based on how relevant they were to the goal variable "species_encoded," the top four traits were chosen. The dataset was adjusted in accordance with the features that had been chosen and displayed.

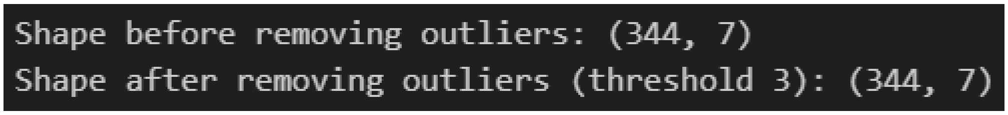
The chosen features were utilized to split the data into features (X) and the target variable (Y). A Random Forest classifier was then initialized and fitted to the training data after the dataset was divided into training and testing sets. The feature importance was found, giving information about how important each attribute is to the model's ability to forecast the future. This stage emphasized the significance of thoughtful feature selection in improving model performance and set the stage for later machine learning evaluations.



```
Selected features:
Index(['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g'], dtype='object')

Feature importances:
[0.38580373 0.18980788 0.35390267 0.07048572]
```

Figure 6: Feature Relevance Analysis using Feature Selection

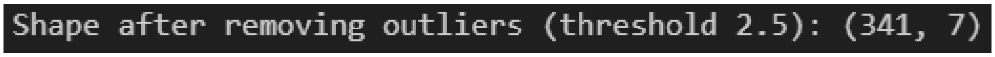
To ensure the robustness of further studies, the dataset underwent outlier removal in addition to feature relevance analysis. Z-scores were computed for each column utilizing the scipy.stats library's 'zscore' function. After outlier removal, a criterion for outlier detection (Z-score > 3 or < -3) that was previously set at 3 did not significantly alter the shape of the dataset (Shape: (344, 7)).



```
Shape before removing outliers: (344, 7)
Shape after removing outliers (threshold 3): (344, 7)
```

Figure 7: Removing Outliers with threshold = 3

The criterion was then changed to 2.5 to investigate any variations. Outliers were detected by the recalculated Z-scores using the updated threshold, and the dataset was again processed to eliminate them. After this modification, the form of the dataset was shown, indicating any significant differences. The use of an iterative technique facilitated a detailed investigation of the impact of outliers on the dataset, guaranteeing a thorough comprehension of its resilience and adaptability to varying threshold values.



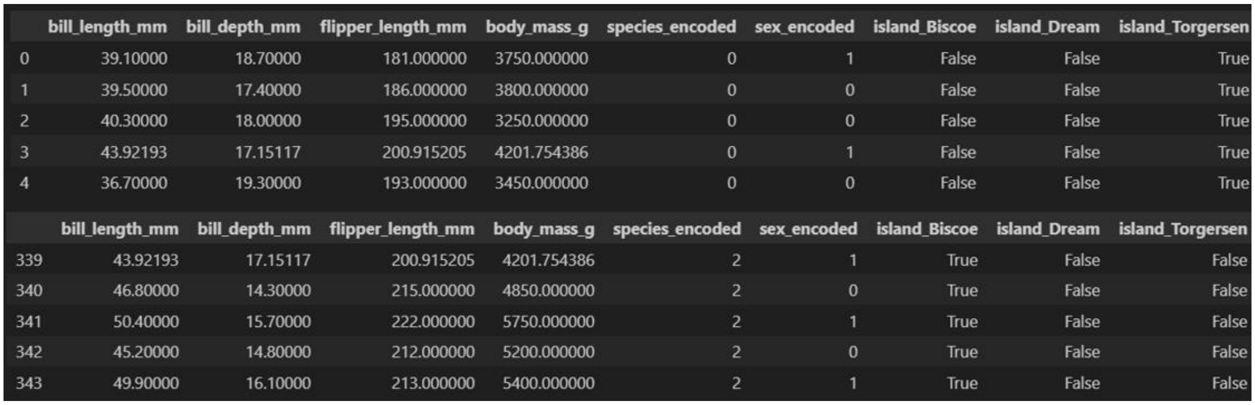
```
Shape after removing outliers (threshold 2.5): (341, 7)
```

Figure 8: Removing Outliers with threshold = 2.5

3.5. *Step 5 - Categorical Variable Encoding:*

This is an important stage since it involves effectively encoding the dataset's category variables into a numerical format that machine learning models can use. By using scikit-learn's LabelEncoder, the "species" and "sex" columns were converted to "species_encoded" and "sex_encoded," individually. The dataset was then streamlined by removing the initial category columns.

Using the Pandas 'get_dummies' function, a clever one-hot encoding method was used for the category variable "island." New binary columns, "island_Biscoe," "island_Dream," and "island_Torgersen," which represent the various island kinds, were consequently created. By using this encoding technique, the dataset is ready for use with machine learning algorithms and can be used with models that need numerical input. The effective conversion of categorical variables into a structured numerical format serves as a basis for further studies and enhances the dataset's overall preparedness for more complex machine learning tasks.



	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	species_encoded	sex_encoded	island_Biscoe	island_Dream	island_Torgersen
0	39.10000	18.70000	181.000000	3750.000000	0	1	False	False	True
1	39.50000	17.40000	186.000000	3800.000000	0	0	False	False	True
2	40.30000	18.00000	195.000000	3250.000000	0	0	False	False	True
3	43.92193	17.15117	200.915205	4201.754386	0	1	False	False	True
4	36.70000	19.30000	193.000000	3450.000000	0	0	False	False	True

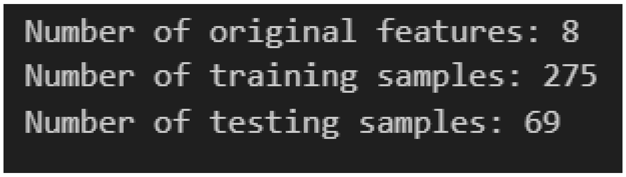
	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	species_encoded	sex_encoded	island_Biscoe	island_Dream	island_Torgersen
339	43.92193	17.15117	200.915205	4201.754386	2	1	True	False	False
340	46.80000	14.30000	215.000000	4850.000000	2	0	True	False	False
341	50.40000	15.70000	222.000000	5750.000000	2	1	True	False	False
342	45.20000	14.80000	212.000000	5200.000000	2	0	True	False	False
343	49.90000	16.10000	213.000000	5400.000000	2	1	True	False	False

Figure 9: Categorical Variable Encoding

3.6. *Step 6 - Dataset Split into Training and Testing Subsets:*

This critical step effectively divided the dataset into training and testing subsets, which is a necessary precondition for assessing the performance of machine learning models. The target variable (y), which is the'species_encoded' column, was created together with the features (X). The data was divided into training (X_train, y_train) and testing (X_test, y_test) sets using the scikit-learn 'train_test_split' function. A random sample of 42 was chosen for repeatability, and a test size of 20% was provided.

Using the Pandas 'get_dummies' function, a clever one-hot encoding method was used for the category variable "island." New binary columns, "island_Biscoe," "island_Dream," and "island_Torgersen," which represent the various island kinds, were consequently created. By using this encoding technique, the dataset is ready for use with machine learning algorithms and can be used with models that need numerical input. The effective conversion of categorical variables into a structured numerical format serves as a basis for further studies and enhances the dataset's overall preparedness for more complex machine learning tasks.



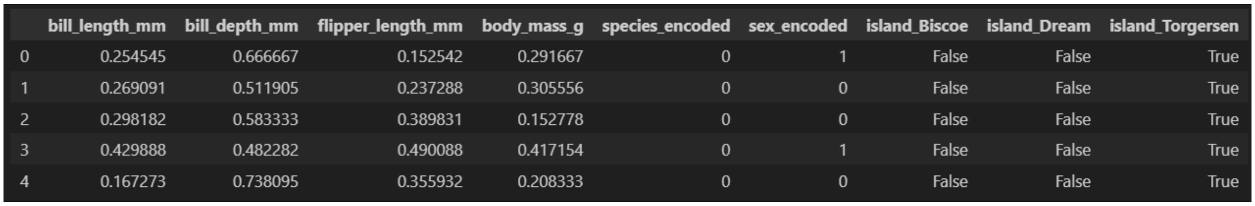
```
Number of original features: 8
Number of training samples: 275
Number of testing samples: 69
```

Figure 10: Dataset Splitting

3.7. *Step 7 - Numerical Feature Scaling:*

The dataset was subjected to strict feature scaling in order to guarantee uniform scaling across numerical features, a crucial step in machine learning model optimization. To normalize the numerical features, scikit-learn's MinMaxScaler was used. The scaler was initialized, and the fit_transform method was used to transform the chosen features, assuming the presence of training and testing feature sets (X_train and X_test).

The first few rows of the data frame after scaling were shown to illustrate how the scaling process affected the dataset. This process ensures that the numerical features remain consistent in size, which leads to better model convergence and performance. When feature scaling is done well, the dataset becomes more adaptable to different machine learning methods, which leads to more precise and trustworthy model predictions.



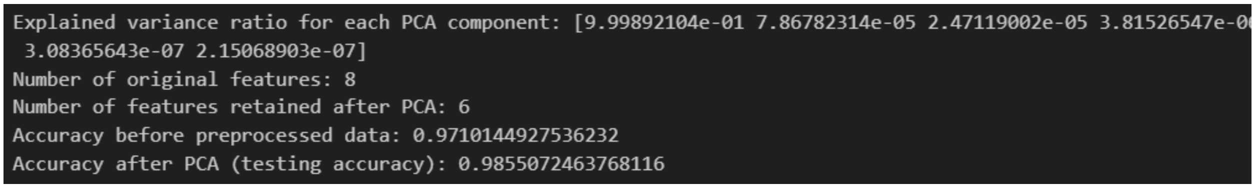
	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	species_encoded	sex_encoded	island_Biscoe	island_Dream	island_Torgersen
0	0.254545	0.666667	0.152542	0.291667	0	1	False	False	True
1	0.269091	0.511905	0.237288	0.305556	0	0	False	False	True
2	0.298182	0.583333	0.389831	0.152778	0	0	False	False	True
3	0.429888	0.482282	0.490088	0.417154	0	1	False	False	True
4	0.167273	0.738095	0.355932	0.208333	0	0	False	False	True

Figure 11: Numerical Feature Scaling

3.8. Step 8 - Dimensionality Reduction Applied:

In this stage, the dataset was reduced in size while maintaining the necessary information by using appropriate dimensionality reduction techniques, principal component analysis (PCA) in particular. Three and six components, respectively, were used to start the PCA instance.

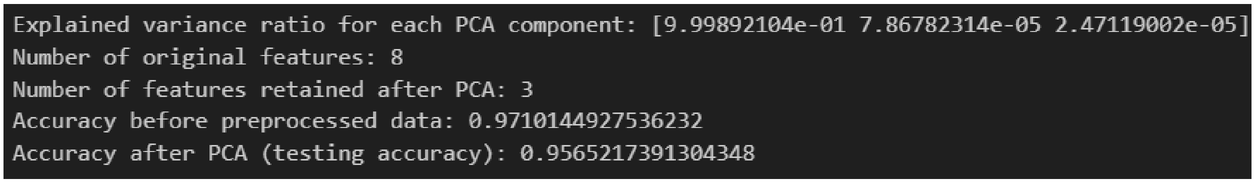
The explained variance ratio for each PCA component was [9.99892104e-01, 7.86782314e-05, 2.47119002e-05, 3.81526547e-06, 3.08365643e-07, 2.15068903e-07] when the n_components were set to 6. As a result, following PCA, the initial set of eight characteristics was reduced to six. The accuracy was 0.9710 prior to preprocessing and 0.9855 with PCA.



```
Explained variance ratio for each PCA component: [9.99892104e-01 7.86782314e-05 2.47119002e-05 3.81526547e-0
 3.08365643e-07 2.15068903e-07]
Number of original features: 8
Number of features retained after PCA: 6
Accuracy before preprocessed data: 0.9710144927536232
Accuracy after PCA (testing accuracy): 0.9855072463768116
```

Figure 12: Dimensionality Reduction and Validating Preprocessing Pipeline with 6 Components

Conversely, for every PCA component, the explained variance ratio was [9.99892104e-01, 7.86782314e-05, 2.47119002e-05] when n_components was adjusted to 3. After PCA, this resulted in a more significant decrease to 3 features. The accuracy was 0.9710 prior to preprocessing and 0.9565 following PCA, a slight reduction.



```
Explained variance ratio for each PCA component: [9.99892104e-01 7.86782314e-05 2.47119002e-05]
Number of original features: 8
Number of features retained after PCA: 3
Accuracy before preprocessed data: 0.9710144927536232
Accuracy after PCA (testing accuracy): 0.9565217391304348
```

Figure 13: Dimensionality Reduction and Validating Preprocessing Pipeline with 3 Components

These findings emphasize the trade-off between model accuracy and dimensionality reduction. It highlights how crucial it is to choose the right number of components to achieve the ideal balance between model performance and data compression.

3.9. Step 9 - Validate Preprocessing Pipeline and Compare Model Performance:

As the last stage of our preprocessing workflow, we used the preprocessed data to train and assess a Random Forest classifier to verify the effectiveness of our modifications. First, we used the raw data to train the classifier without doing any preparation. We then implemented the whole preprocessing pipeline, which included data cleansing, PCA-assisted dimensionality reduction, split into training and testing sets, categorical variable encoding, and numerical feature scaling. The preprocessed data was then used to train a new Random Forest model. We sought to determine whether our preprocessing efforts had improved performance by comparing the accuracy of the two models. This stage made sure that our preprocessing techniques were validated and optimized to increase the overall efficacy of the machine learning model.

4. Conclusion

As a result of our investigation into the Penguins dataset and the subsequent implementation of a thorough pretreatment pipeline, we have gained important knowledge about the dynamics involved in data preparation for machine learning. The preliminary analysis of the data revealed problems like missing values and outliers, which prompted careful approaches to address these problems. Using methods such as dimensionality reduction using PCA, imputation, and categorical encoding, we were able to successfully convert the dataset into a format that was suitable for machine learning research.

Notable outcomes were obtained when we trained and assessed a Random Forest classifier to validate our preprocessing strategy. The significant influence of our preparation efforts was demonstrated by comparing the model's performance on raw data versus preprocessed data. The observed increases in accuracy following dimensionality reduction highlight how important well-chosen preprocessing measures are to improving model performance.

In conclusion, our research highlights how important preprocessing is to machine learning model optimization. Model accuracy and computational performance can both be increased by successfully implementing techniques for data cleansing, feature engineering, and dimensionality reduction. These revelations will direct our future machine learning efforts and emphasize the critical relationship between model performance and data preparation.