University of Economics in Prague

Faculty of Informatics and Statistics

DATA-X - Classification of Penguins species

Subject: Data-X – Applied Data Analytics Models in Real World Tasks (4IT439)

Team 3

Bc.Hana Marková, Ing. Ekaterina Pronevich, Bc.Jakub Sokol, Bc.Karolína Pěstová

April 18,2023

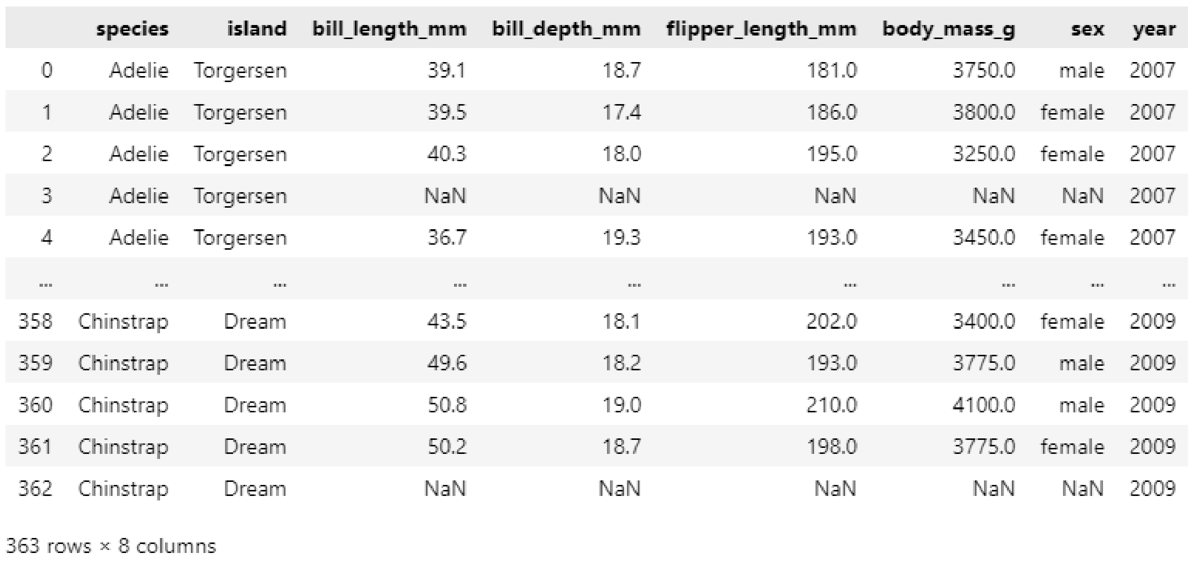
- 1 Content

- 1 Problem definition

We were presented with a dataset of penguin characteristics. The aim of our work was to predict the species of the penguin based on its characteristics. The species were Adélie, Chinstrap and Gentoo. The goal of our work was to implement models that classify penguins by species. We utilized libraries pandas, matplotlib, seaborn, sklearn, and skopt for training the models. Our measure of success for the models is based on the accuracy, precision, recall, F1 score, and support metrics.

- 2 Data Understanding

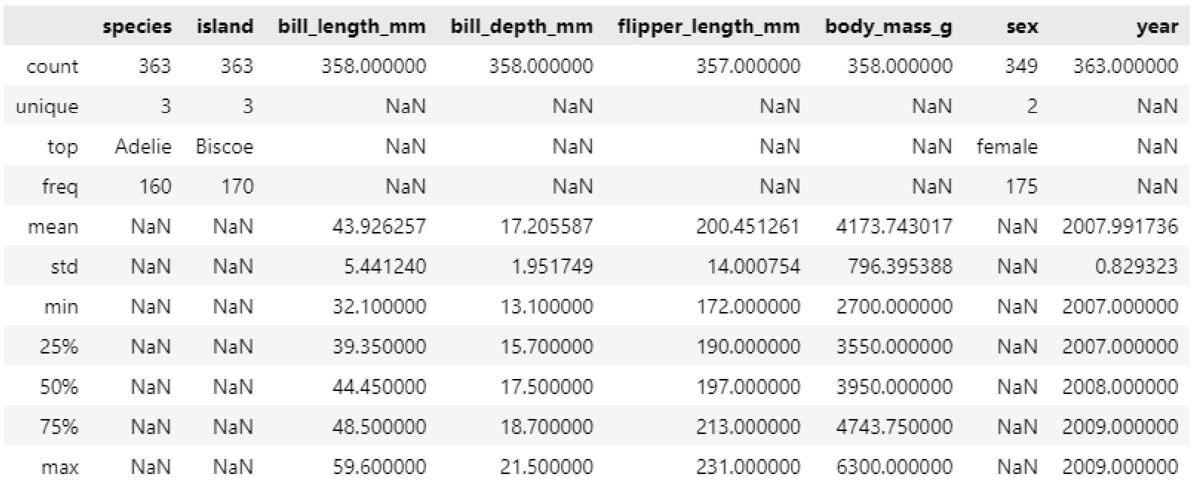
The dataset is presented in csv format with 8 columns (species, island, bill length, bill depth, flipper length, body mass, sex, year) and 363 rows. A preview of the data is shown below.



	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	female	2007
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN	2007
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	female	2007
...	...	...	...	...	...	...	...	...
358	Chinstrap	Dream	43.5	18.1	202.0	3400.0	female	2009
359	Chinstrap	Dream	49.6	18.2	193.0	3775.0	male	2009
360	Chinstrap	Dream	50.8	19.0	210.0	4100.0	male	2009
361	Chinstrap	Dream	50.2	18.7	198.0	3775.0	female	2009
362	Chinstrap	Dream	NaN	NaN	NaN	NaN	NaN	2009

363 rows × 8 columns

The species variable takes on Adelie, and Gentoo, Chinstrap. The island variable takes on Biscoe, Dream, and Torgersen. The main descriptive statistics and original data types of these criteria are presented below. Among others, we can also see how many unique variables are in the dataset for each variable.

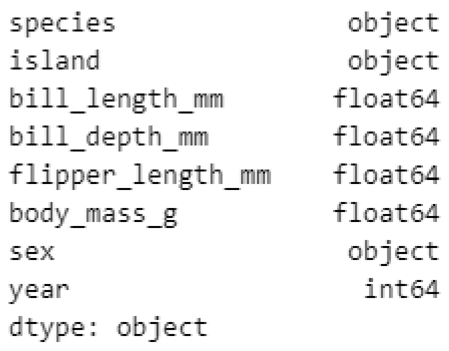


	species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
count	363	363	358.000000	358.000000	357.000000	358.000000	349	363.000000
unique	3	3	NaN	NaN	NaN	NaN	2	NaN
top	Adelie	Biscoe	NaN	NaN	NaN	NaN	female	NaN
freq	160	170	NaN	NaN	NaN	NaN	175	NaN
mean	NaN	NaN	43.926257	17.205587	200.451261	4173.743017	NaN	2007.991736
std	NaN	NaN	5.441240	1.951749	14.000754	796.395388	NaN	0.829323
min	NaN	NaN	32.100000	13.100000	172.000000	2700.000000	NaN	2007.000000
25%	NaN	NaN	39.350000	15.700000	190.000000	3550.000000	NaN	2007.000000
50%	NaN	NaN	44.450000	17.500000	197.000000	3950.000000	NaN	2008.000000
75%	NaN	NaN	48.500000	18.700000	213.000000	4743.750000	NaN	2009.000000
max	NaN	NaN	59.600000	21.500000	231.000000	6300.000000	NaN	2009.000000

Figure 1 Descriptive statistics

- 3 Data Preparation

For more efficient work with data, we needed to do the preprocessing. We found out from the output below that 4 variables are either categorical or discontinuous and their type needs to be converted to category type. This step was the first due to the fact that some variables had a data type that cannot be used in the modeling.



```
species               object
island                object
bill_length_mm       float64
bill_depth_mm        float64
flipper_length_mm    float64
body_mass_g          float64
sex                   object
year                   int64
dtype: object
```

Figure 2 Original data types

We converted these variables (species, island, sex, and year) to the category type. The result of the change is displayed below:

```
species                 category
island                  category
bill_length_mm           Float64
bill_depth_mm            Float64
flipper_length_mm        Float64
body_mass_g                Int64
sex                     category
year                    category
dtype: object
```

Then we created dummy variables for categorical variables from the dataset. We didn't create a dummy for the species category as it is a target variable.

Getting dummies

```
df_penguins = pd.get_dummies(df_penguins, columns=["island","sex","year"], drop_first=True)
```

The next equally important change was excluding empty and repeating values. This helped us get unique and not missing values. We dropped all duplicated rows. We discovered that when some data is missing in a row, it is always missing from other columns of the given record as well. Such a record is irrelevant to us, so we have deleted all these records. After cleaning up, the number of rows in the dataset was reduced to 337.

```
Number of dupliated rows:  13

Number of missing values in each column:
species               0
island                0
bill_length_mm        5
bill_depth_mm         5
flipper_length_mm     6
body_mass_g           5
sex                  14
year                  0
dtype: int64
```

- 4 Data Visualization

The first need of the visual was to find out the number of penguins depending on the species. As we can see in the graph, the number of Adelie penguins is twice as high as the number of Chinstap penguins. The number of Gentoo penguins is also above the average, 120 penguins to be exact.

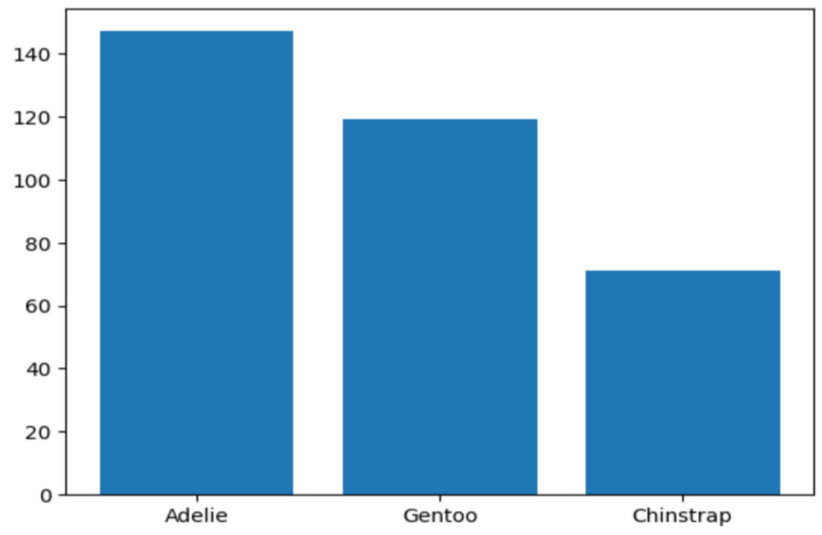


Figure 5 Number of penguins of a particular species

Next, we were interested in how many penguins live on a definite island. From the graph below we see that the largest number of penguins inhabit the island of Biscoe. The smallest number of penguins live on Torgensen Island. Then from this graph we are interested in how many of certain species live on each of these islands.

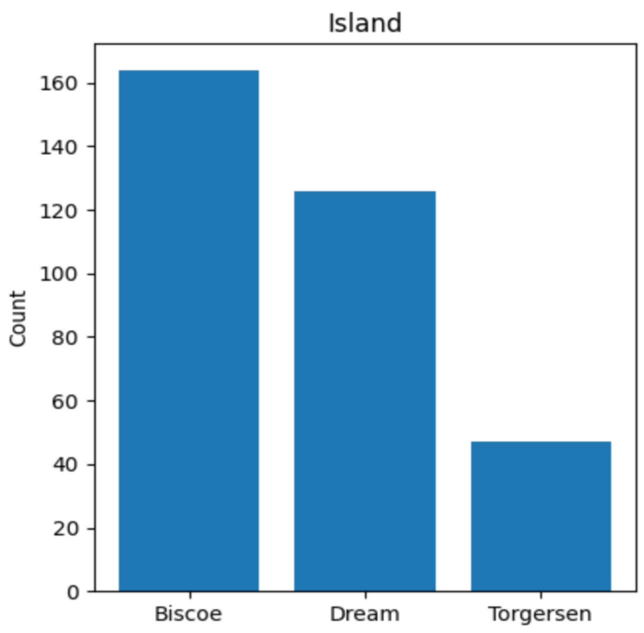


Figure 6 Number of penguins on the island

Based on our query above, a visualization of the number of penguins of a particular species living on a particular island was created. As we can see on the graph, Biscoe Island is dominated by Gentoo penguins, but in spite of this, the island is also inhabited by Adelie penguins. Dream Island is the only island with the species Chinstrap. In contrast to Biscoe and Dream Island, Torgersen Island has only Adelie penguins in almost the same numbers as the other islands.

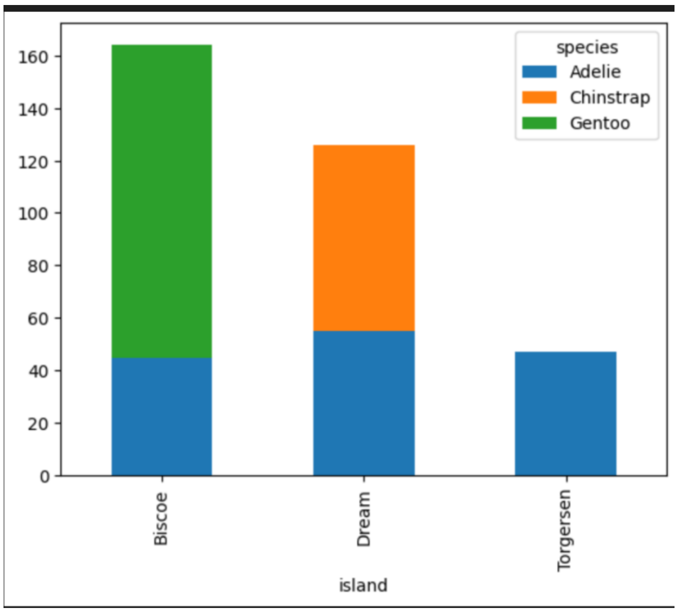


Figure 7 Occurrence of penguin species on individual islands

We were also interested in the sex ratio of the total number of penguins. As we can see in our graph, the number of females and males is almost 1 to 1.

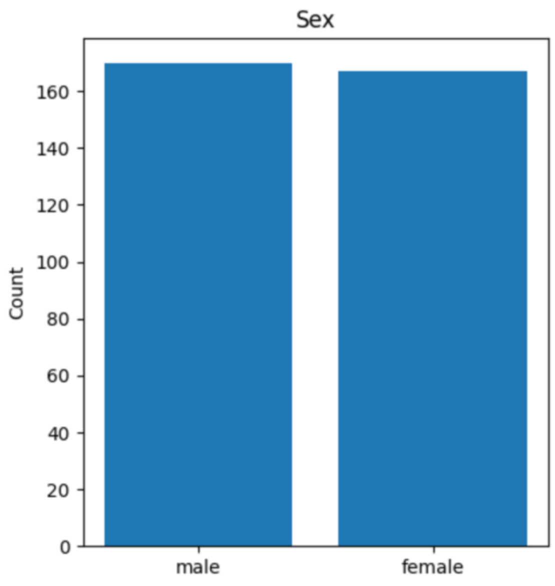
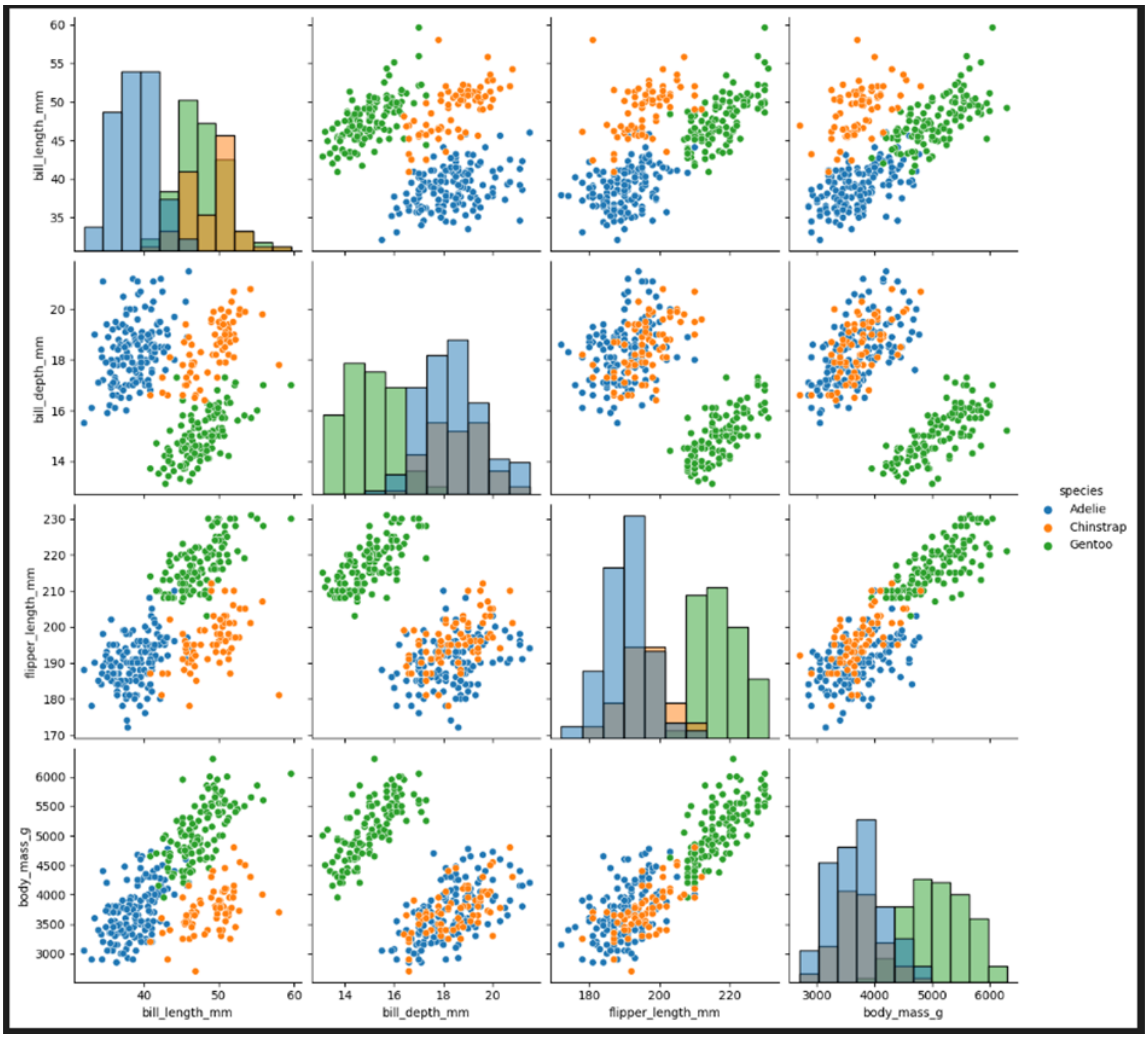


Figure 8 The number of penguins of a gender

We also displayed histograms and scatter plots of the continuous variable, all colored by species type. It is possible to see on the dot charts that some combinations of variables can clearly distinguish penguin species. And it is also possible to detect specific characteristics of individual penguin species.



A correlation analysis was also carried out to get the relationships between all the attributes. A heat map from 0 to 1 was created containing a correlation table. Based on the theory that the attributes with the highest correlation coefficient best explain the class variable we can see in a practical example a very high positive correlation between the variables flipper length and body mass. Thus we get that the greater the length of the flipper, the greater the mass of the body.

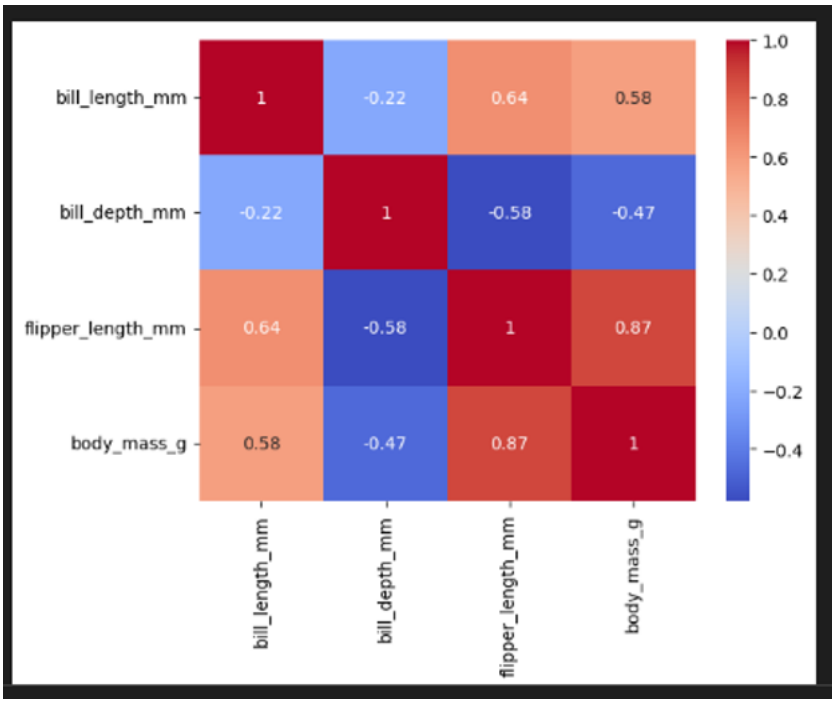


Figure 9 Correlation matrix

- 5 Modeling

For predicting penguins species we used these models: Random Forest, Logistic Regression, and Gradient Boosting. Before building the models, we divided our dataset into 2 parts. The division was made into the following percentage parts: training (70%), and validation (30%) set. For each model, we tuned its hyperparameters based on Bayesian optimization. We also use 5-fold cross-validation.

- 5.1. Random Forest

- 5.1.1 About Random Forest Model

The first model we chose is Random Forest. Random Forest is one of the most popular machine-learning algorithms for classification and regression tasks. It is based on an ensemble of decision trees, where each tree is trained on a random subset of the data and a random subset of features. There are several reasons why we choose to use the Random Forest model:

- High accuracy: Random Forest demonstrates high accuracy in prediction, especially when working with a large number of features and complex interactions between them. This is supported by various studies, such as the research conducted [1,2].
- Resistance to overfitting: Random Forest has built-in protection against overfitting, as each tree is built on a random subset of data and features. This allows the model to maintain high accuracy on new data. This is supported by research conducted [3].
- Interpretability: The decision trees that make up the Random Forest model can be easily visualized, allowing for easy interpretation of results. This is supported by various studies, such as the research conducted [4].
- Speed: Unlike other models, Random Forest can handle large datasets while maintaining fast speed. This is supported by various studies, such as the research conducted [1].

- 5.1.2 Building Random Forest Model

We build a Random forest model for penguin classification:

- Definition of Random Forest Model

```
rf_clf = RandomForestClassifier(random_state=seed)
rf_param_grid = {
    'n_estimators': Integer(1, 1000),
    'criterion': Categorical(['gini', 'entropy']),
    'max_depth': Integer(1, 15),
    'max_features': Integer(3, X_train.shape[1]),
    'min_samples_leaf': Integer(5, 500)
}
```

- Bayesian optimization for tuning hyperparameters with 5-fold cross-validation.

```
rf_search = BayesSearchCV(
    estimator=rf_clf,
    search_spaces=rf_param_grid,
    n_iter=100,
    cv=5,
    random_state=seed,
    n_jobs=-1
)
```

- Fitting the model

```
rf_search.fit(X_train, y_train)
```

- Descovering best model with best parameters

```
print('Random Forest:')
print('Best model:', rf_search.best_estimator_)
print('Best score:', rf_search.best_score_)
Random Forest:
Best  model:  RandomForestClassifier(criterion='entropy',  max_depth=14,
max_features=3,min_samples_leaf=6, n_estimators=1000, random_state=777)
Best score: 0.9829787234042552
```

As can be seen from the results above, among the best parameters selected was e.g. max tree depth of 14, samples per tree leaf of 6 or the selection criterion being entropy. The Random Forest model is very accurate, the model is performing with an accuracy of 0.982 and it is very close to a perfect model.

- 5. 2 Logistic Regression

- 5.2.1 About Logical Regression Model

Also in our work, we used the Logistic Regression model a popular machine-learning algorithm used for both classification and regression tasks. It is an ensemble method that combines multiple decision trees to make predictions. There are several reasons why we choose to use the Random Forest model:

- Simplicity: Logistic Regression is a simple algorithm that is easy to understand and interpret [5].
- Efficiency: Logistic Regression is a relatively fast algorithm that can be efficiently applied to large volumes of data [6].
- Interpretability: The results of Logistic Regression can be easily interpreted, as it outputs coefficients for each feature, making it possible to understand the importance of each feature for classification [7].

5.2.2 Building Logical Regression Model

We build a Logical Regression model for penguin classification:

- Definition of Random Forest Model

```
lr_clf = LogisticRegression(random_state=seed)
```

- Bayesian optimization for tuning hyperparameters with 5-fold cross-validation.

```
lr_param_grid = {
     'fit_intercept': Categorical([True, False]),
     'C': Real(0.001, 1000),
     'penalty': Categorical(['l2', 'none'])
}
```

- Fitting the model

```
lr_search.fit(X_train, y_train)
```

- Discovering best model with best parameters

```
print('Logistic Regression:')
print('Best model:', lr_search.best_estimator_)
print('Best score:', lr_search.best_score_)
Logistic Regression:
Best model: LogisticRegression(C=848.4509333096364, random_state=777)
Best score: 1.0
```

As can be seen from the results above, the best parameters selected was c of approximately 848 and random_state of 777. The Logical Regression model is very accurate, the model is performing with an accuracy of 1 and it seems to be a very efficient model for penguin classification.

- 5. 3 Gradient Boosting

- 5.3.1 About Gradient Boosting Model

The last model for our project was the Gradient Boosting model. This model is a powerful machine-learning algorithm used for both classification and regression tasks. There are several reasons why we choose to use the Gradient Boosting model:

- Accuracy: Gradient Boosting is known for its high accuracy and ability to capture complex non-linear relationships between features. It achieves this by iteratively improving the model by minimizing the loss function with respect to the predictions[9].
- Robustness: Gradient Boosting is a robust algorithm that can handle noisy and missing data[8].
- Flexibility: Gradient Boosting is a flexible algorithm that can be customized with various hyperparameters to optimize performance, such as the learning rate, number of trees, and tree depth[10].

5.2.3 Building Gradient Boosting Model

We build a Logical Regression model for penguin classification:

Definition of Gradient Boosting Model:

- Definition of Gradient Boosting Model

```
gb_clf = GradientBoostingClassifier(random_state=seed)
```

- Bayesian optimization for tuning hyperparameters with 5-fold cross-validation.

```
gb_param_grid = {
    'n_estimators': Integer(1, 1000),
    'max_depth': Integer(1, 15),
    'learning_rate': Real(0.001, 100),
    'min_samples_leaf': Integer(5, 500),
    'max_features': Integer(3, X_train.shape[1])
}
```

- Fitting the model

```
gb_search.fit(X_train, y_train)
```

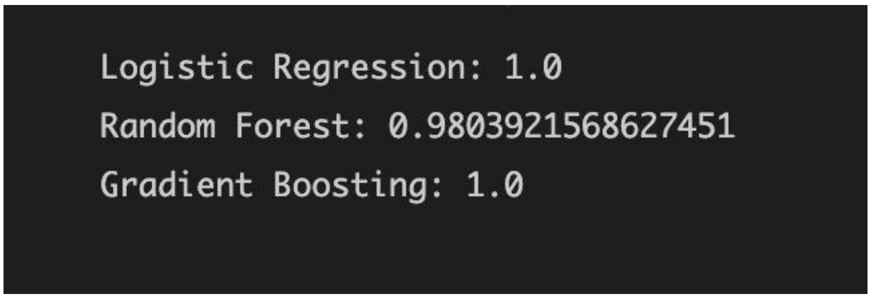
- Discovering best model with best parameters

```
print('Gradient Boosting:')
print('Best model:', gb_search.best_estimator_)
print('Best score:', gb_search.best_score_)
Gradient Boosting:
Best model: GradientBoostingClassifier(learning_rate=0.001,
max_features=3,min_samples_leaf=5, n_estimators=1000,random_state=777)
Best score: 0.9914893617021276
```

As can be seen from the results above, among the best parameters selected was e.g. features of 3 or min samples per tree leaf of 5. The Random Forest model is very accurate, the model is performing with an accuracy of 0.992 and it is very close to a perfect model.

- 6 Evaluation

Based on the conducted research using the Random Forest, Logistic Regression, and Gradient Boosting models, the following accuracy values were obtained:



```
Logistic Regression: 1.0
Random Forest: 0.9803921568627451
Gradient Boosting: 1.0
```

The Random Forest model showed an accuracy of 98.04%

The Logistic Regression model showed an accuracy of 100%

The Gradient Boosting model showed an accuracy of 100%

These results indicate that all three models performed the classification task with high accuracy. The Logistic Regression and Gradient Boosting models showed the highest accuracy results of 100%, which demonstrates the effectiveness of these models for the given task. Overall, these results suggest that the models used in this research could be effective for solving classification tasks in this field.

- Development Environment Characteristics

Python 3.8.10

Packages: pandas,metplotlib,seaborn,sklearn and skopt.

- Sources:

[1] Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

[2] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[3] Cutler, D. R., et al. (2007). Random forests for classification in ecology. Ecology, 88(11), 2783-2792.

[4] Strobl, C., et al. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics, 8(1), 25.

[5] Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 4). Springer.

[6]Hosmer Jr, D. W., & Lemeshow, S. (2004). Applied logistic regression (Vol. 398). John Wiley & Sons.

[7]James, G., et al. (2013). An introduction to statistical learning. Springer.

[8]Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of statistics, 29(5), 1189-1232.

[9]Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

[10]Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 785-794).

-