FYS-STK4155 – Project 3 on Machine Learning

Algorithms for penguin species classification

Viggo Wetteland

December 14, 2020

Abstract

In this project, we have studied in detail a classification problem involving the Palmer penguins. We tested various supervised machine learning algorithms and found that the Logistic Regression algorithm gave us the highest accuracy score of 0.990 when predicting the three penguin species, followed by the Decision Trees algorithm score of 0.974. We also found that the best way to fill out the missing genders in the dataset was to train machine learning algorithms and try to predict them instead of just assigning them the most frequent value, and was 0.4 – 1.7% better than assigning the missing datapoints with the most frequent values. For these gender predictions, the Decision Trees algorithm performed best with an accuracy score of 0.940, followed by the Logistic Regression algorithm with a score of 0.806.

1 Introduction

The goal of this project is to develop a model for successfully classifying penguin species based on their attributes. The dataset used is called the Palmer Archipelago penguin dataset and was collected and made available by Dr. Kirsten Gorman and the Palmer Station, Antarctica LTER, a member of the Long Term Ecological Research Network **[1]**. The dataset contains data from measures made of 344 penguins. There are three different species of penguins in this dataset, collected from three islands in the Palmer Archipelago, Antarctica. Every penguin has five main attributes which are all displayed in the dataset: culmen length, culmen depth, flipper length, body mass, and gender. Some of these values are missing for a small number of the penguins, which is going to be part of this project's challenge too. We cannot work with a dataset with missing values so we need to find a way to work around them. Two solutions on how to handle missing values within a dataset will be presented and compared. The first one is to simply assign each of the missing values with the most frequent value for that attribute found elsewhere in the dataset, and the second being predicting the missing values with the best suitable machine learning algorithm.

Four different supervised machine learning algorithms will be tested to try and discover which one performs the best on this kind of classification problem. These algorithms are Logistic Regression, Feedforward Artificial Neural Network, Decision Trees, and Support Vector Machines. The methods section will include an explanation of the basics of these algorithms, and their performance will be

measured by accuracy score, confusion matrices, and classification reports. For visualizing the dataset and getting a good overview of the problem, we're going to use the functionalities of pandas and seaborn in Python.

This report is structured with an abstract, introduction, method section, results and discussion section, conclusion, appendix, and references.

2 Method

2.1 Structuring and understanding our dataset

2.1.1 Reading the data

The penguin data can be collected from Kaggle in the downloadable file "penguins_size.csv" **[3]**. For reading the data, I used pandas' "read_csv" to structure the data into a dataframe to take advantage of the very practical functionalities of pandas. For example "print(dataset.head(20))" will give us this clean overview of the first 20 rows of the dataset:

	species	island	culmen_length_mm	culmen_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
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE
6	Adelie	Torgersen	38.9	17.8	181.0	3625.0	FEMALE
7	Adelie	Torgersen	39.2	19.6	195.0	4675.0	MALE
8	Adelie	Torgersen	34.1	18.1	193.0	3475.0	NaN
9	Adelie	Torgersen	42.0	20.2	190.0	4250.0	NaN
10	Adelie	Torgersen	37.8	17.1	186.0	3300.0	NaN
11	Adelie	Torgersen	37.8	17.3	180.0	3700.0	NaN
12	Adelie	Torgersen	41.1	17.6	182.0	3200.0	FEMALE
13	Adelie	Torgersen	38.6	21.2	191.0	3800.0	MALE
14	Adelie	Torgersen	34.6	21.1	198.0	4400.0	MALE
15	Adelie	Torgersen	36.6	17.8	185.0	3700.0	FEMALE
16	Adelie	Torgersen	38.7	19.0	195.0	3450.0	FEMALE
17	Adelie	Torgersen	42.5	20.7	197.0	4500.0	MALE
18	Adelie	Torgersen	34.4	18.4	184.0	3325.0	FEMALE
19	Adelie	Torgersen	46.0	21.5	194.0	4200.0	MALE

It becomes evident that penguin number 3 has missing values on everything but species and island. Penguin number 339 is also missing the same values, and there are 9 additional penguins with missing values for their gender in the dataset. In section 2.3 we will look at two different solutions for dealing with these missing values with help from some of the other functionalities of pandas.

2.1.2 Visualizing the data

It's hard to visualize the dataset by looking at the dataframe alone, so we're going to include some seaborn plots to get a better picture of what the dataset contains. The pairplot is especially effective for visualizing the dataset, but other plots like FacetGrid and violinplot help zoom in at specific areas.

1. seaborn.pairplot (figure 1)

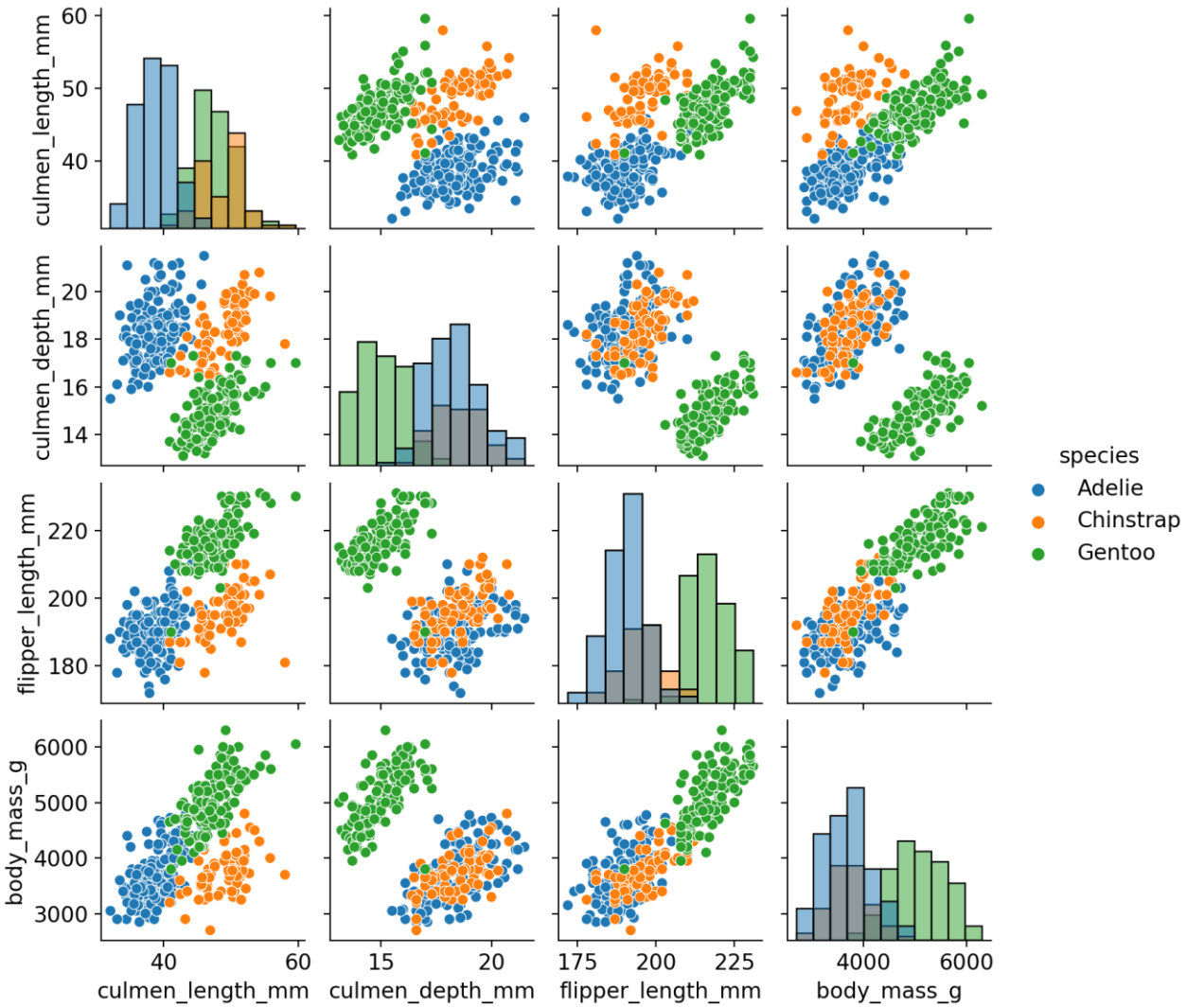

Figure 1: The pairplot visualizes the whole dataset, giving an idea of what to expect from the classification results. From this plot we can see that the Gentoo penguins (green dots) should be easiest to classify out of the three.

2. seaborn.FacetGrid (figures 2 and 3)

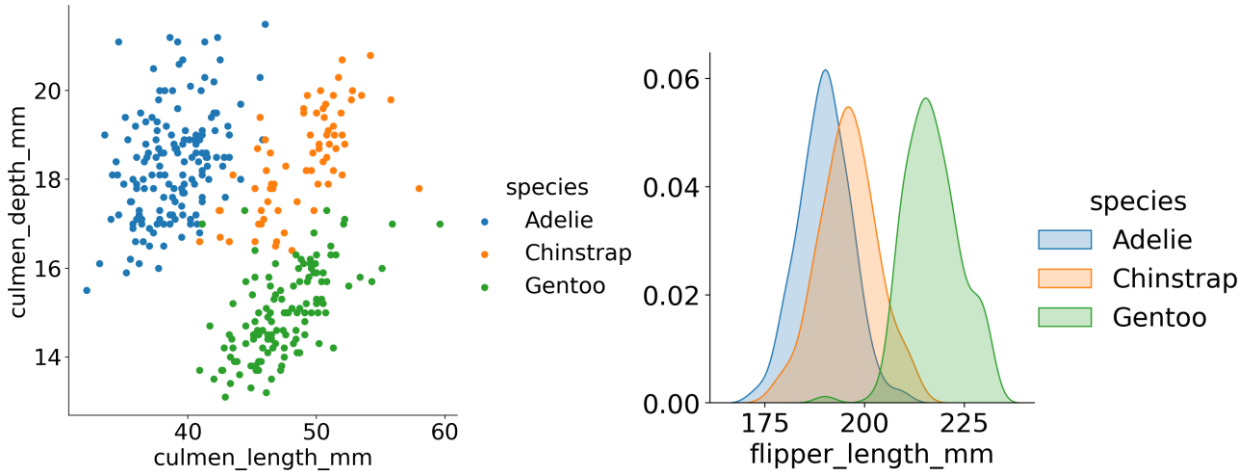

Figure 2 & 3: FacetGrid plots of culmen depths versus culmen length on figure 2 on the left, and flipper length on figure 3 on the right. This is for zooming in on the areas in figure 1 to see more detail.

3. seaborn.violinplot (figure 4)

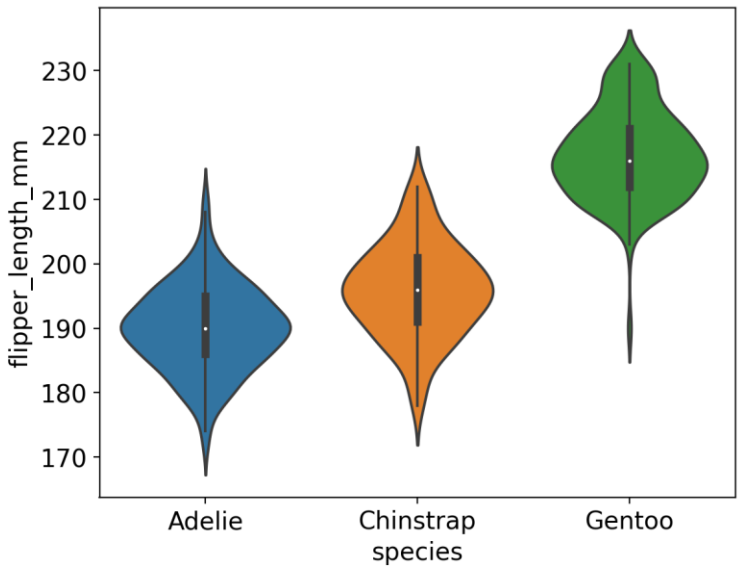

Figure 4: A violinplot of the flipper length giving a different vision on the same data representation displayed in figure 3. The width of the "leaves" corresponds to the proportion of the penguin species having that flipper length. It reveals a Gauss distribution which is very often found in nature.

From the pairplot displayed in figure 1 in particular, we see that we can expect Gentoo penguins to be the easiest to classify. Adelie and Chinstrap penguins have attributes that mostly blend together so this is going to be the hardest data to classify.

2.2 Various classification algorithms

2.2.1 Logistic Regression

Linear Regression is used for solving Regression problems, but when solving classification problems we need to transition over to Logistic Regression. We then need to introduce the logistic function

$$ln\left(\frac{P}{1-P}\right) = w_0 + w_1 x \quad (1)$$

where P is the probability that an event will occur, x is the predictor variable and the b values are the weights (linear parameters). Logistic regression makes use of the Sigmoid function, giving outputs between 0 and 1, but by adding additional expressions $w_2 x + w_3 x + \cdots + w_n x$, the algorithm can handle multi-class problems as well **[2]**. Below in figure 5 is an illustration of the differences between linear and logistic regression which shows the logarithmic nature of the logistic function.

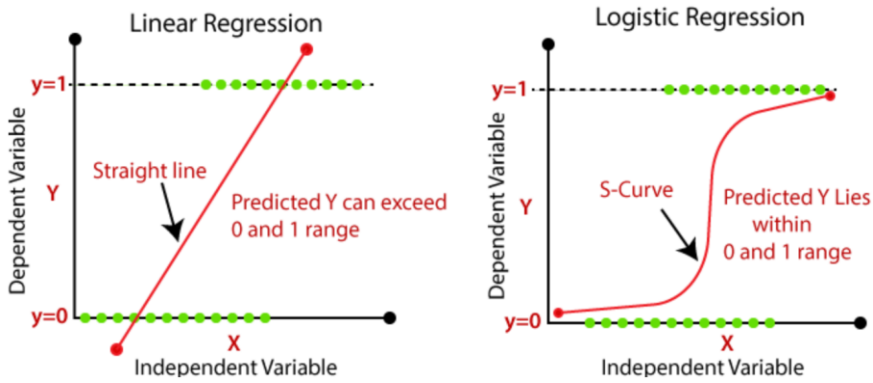

Figure 5: Graphs of linear and logistic regression to illustrate the differences. Notice how in logistic regression the Y values are constricted to being between 0 and 1.

Instead of a continuous value, the output of Logistic Regression becomes a categorical value such as 0 or 1 and can be used for classification problems.

We will use sklearn.linear_model's "LogisticRegression" algorithm, and we're going to use the One-to-Rest approach by setting the "multi_class" option to "ovr" (or auto). A brief explanation of the One-to-Rest approach is found in section 2.2.4.

2.2.2 Decision Trees

A decision tree algorithm is a supervised machine learning algorithm most used for classification problems, but can also be used for regression problems. A decision tree follows a set of if-else conditions to visualize the data and classify it according to the conditions. An example of a decision tree is the animal tree in figure 6 below where the algorithm classifies the data into four different classes based on three conditions.

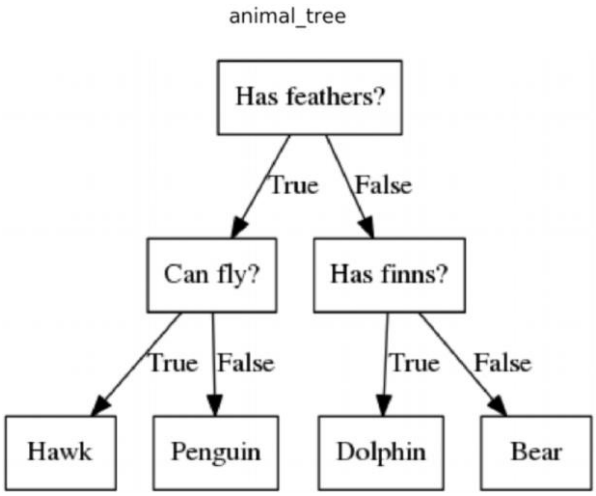

Figure 6: A simple illustration to show how a decision tree could distinguish between four different types of animals. Here there are three if-else statements that classify the animals.

The *root* of the upside-down tree is usually the most important metric. We then continue down through the *internal nodes* of the tree until we reach the *leaf nodes* with predicted class values **[6]**. Libraries of various programming languages use different methods of building these decision trees. Below is a brief explanation of each of the algorithms that can be used.

1. ID3

ID3 generates a tree by considering the whole set S as the root node. It then iterates on every attribute and splits the data into fragments known as subsets to calculate the entropy of the information gain of that attribute. After splitting, the algorithm recourses on every subset by taking those attributes which were not taken before into the iterated ones. It is not an ideal algorithm as it generally overfits the data and on continuous variables, splitting the data can be time-consuming **[7]**.

2. C4.5

It is quite advanced compared to ID3 as it considers the data which are classified samples. The splitting is done based on the normalized information gain and the feature having the highest information gain makes the decision. Unlike ID3, it can handle both continuous and discrete attributes very efficiently and after building a tree, it undergoes *pruning* by removing all the branches having low importance **[7]**.

3. CART

CART can perform both classification and regression tasks and they create decision points by considering Gini index, unlike ID3 or C4.5 which uses information gain and gain ratio for splitting. For splitting, CART follows a greedy algorithm that aims only to reduce the cost function (the accuracy score in our case). For classification, cost function such as Gini index is used to indicate the purity of the leaf nodes **[7]**.

4. CHAID

CHAID or Chi-square Automatic Interaction Detector is a process that can deal with any type of variables be it nominal, ordinal, or continuous. In classification trees, it uses the Chi-Square test. In this analysis, continuous predictors are separated into an equal number of observations until an outcome is achieved. Compared to the other algorithms it is not very much used **[7]**.

This project uses the decision tree algorithm in sklearn.tree's "DecisionTreeClassifier", which is a Classification And Regression Tree (CART) algorithm.

2.2.3 Feedforward Artificial Neural Network

For our problem, we're going to test a Multilayer Perceptron (MLP) algorithm, which is a feedforward artificial neural network classifier. MLP is a deep learning method that is characterized by several layers of input nodes connected as a directed graph between the input and output layers, which means that the signal path through the nodes only goes one way **[8]**. Each node, apart from the input nodes, has a nonlinear activation function, which will be RELU in our case. An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. The backpropagation method is used for training the network **[8]**. Sklearn.neural_network's "MLPClassifier" is the specific MLP algorithm we're going to use. With stochastic gradient descent as the solver type, and since we're using backpropagation, the calculation of the gradient proceeds backward through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last. Partial computations of the gradient from one layer are reused in the computation of the gradient for the previous layer. This backward flow of information allows for efficient computation of the gradient at each layer versus the native approach of calculating the gradient of each layer separately **[10]**.

2.2.4 Support Vector Machines

A Support Vector Machine (SVM) is a supervised machine learning algorithm that is based on the idea of finding a hyperplane that best divides a dataset into two classes. In its base form, linear separation, SVM tries to find a line that maximizes the separation between a two-class dataset of 2-dimensional space points. To generalize, the objective is to find a hyperplane that maximizes the separation of the datapoints to their potential classes in an n-dimensional space **[4]**. The datapoints with the minimum distance to the hyperplane are called *Support Vectors* and are the dotted lines in figure 7 below.

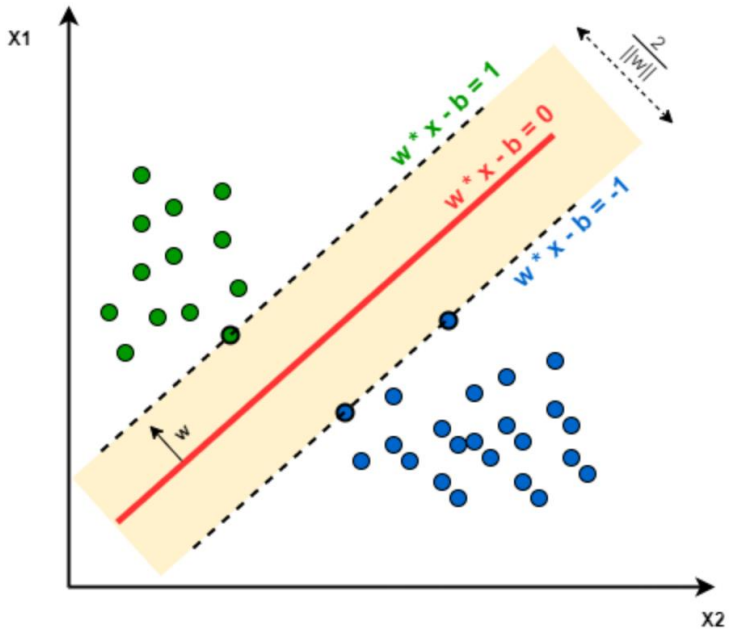

Figure 7: An illustration of a hyperplane (solid red line) and two support vectors (dotted lines) that divides a dataset into a green class and a blue class.

Since SVM only separates data points into two classes, we have two options when using this method for multiclass classification. The first option is the One-to-One approach, which breaks down the multiclass problem into multiple binary classification problems, and adds up to $\frac{m(m-1)}{2}$ SVMs. The second option is the One-to-Rest approach, which looks at each class versus all of the rest, which will then also become a set of binary classification problems and will add up to ***m*** SVMs **[5]**. In this project, we'll be using the One-to-One approach in sklearn.svm's "SVC" because our data set isn't particularly large and we don't have that many classes either.

2.3 Dealing with the NaN values

We're going to present two different ways of handling the missing values in the dataset. The first solution is very simple, while the second solution is more complicated.

Solution 1

With sklearn.impute's "SimpleImputer" we will assign the most frequent values that occurred in the rest of the dataset to these missing datapoints. This is basically just blind guessing, but it's the guess that has the highest probability of being correct. This solution only requires two lines of code:

```
imputer = SimpleImputer(strategy='most_frequent')
dataset.iloc[:,:] = imputer.fit_transform(dataset)
```

Then after replacing the genders with numbers, for example MALE = 2 and FEMALE = 1 with the help of sklearn.preprocessing's "LabelEncoder", we have a complete dataset without any missing values and a way for the algorithm to recognize the genders based on numbers instead of words.

Solution 2

With this next solution, we're going to try and predict the actual values instead of just assigning the most frequent ones. However, for the penguins that have all of their attributes missing there is nothing to base the predictions on, so the two rows with all attributes missing are going to be removed from the dataset. The 9 remaining missing values are all gender values, and these can be predicted based on information from the other attributes.

To start off, we're going to make use of pandas.DataFrame's "fillna" to replace all the NaN values with zeros. There is also a "." value that has to be manually changed to zero. Next up is removing the penguins with missing genders from the dataset and placing them in a temporary list. We will perform a train-test-split on the rest of the dataset which now only contains penguins with no missing attributes and then train the models to predict genders based on the other attributes. The model with the best accuracy score will then be used to predict the missing values within our temporary list from earlier. The code implementation for this solution will follow the same steps as in section 2.4. Penguins in the temporary list will then also have genders assigned to them and can be rejoined with the rest of the dataset. LabelEncoder can be used here as well to get numbers instead of words for the genders to get a complete dataset that can be interpreted by our algorithms.

2.4 The code implementation

Our main goal is to find the model that best predicts the different penguin species. The code implementation, therefore, revolves around executing these three basic steps:

Step 1

First of all, we need to do a train-test-split on our now complete penguin dataset. The X-array will consist of the attributes. Including the name of the island is optional and won't affect the results much. The y-array will consist of the name of the species. The code itself would look something like this:

```
array = dataframe.values (convert dataframe to array)
X = array[:, 1:7]
y = array[:, 0]
```

Followed by a standard train-test-split with a test size of 0.20.

Step 2

The next task is to evaluate our models on the training set to find which model(s) we will use later in the last step where we start to make predictions. An effective way of doing this is by looping over a list containing the models. In the loop, we can set up a test harness to use k-fold cross-validation for example **[11]**. We're going to use 10 folds and the algorithm is called "StratifiedKFold" from the sklearn.model_selection library. The accuracy score can then be tested for each model using sklearn.model_selection's "cross_val_score" which evaluates a score by cross-validation. The code could look like this:

```
models = [insert all four models here]
for model in models:
        kfold = StratifiedKFold(n_splits = 10)
        results = cross_val_score(model, X_train, Y_train, cv=kfold)
```

Step 3

The models with the best results from the evaluation can now be tested on the validation set which is the 20% we held back earlier in step 1. This can very easily be accomplished with help from sklearn.metrics' "accuracy score", which is an accuracy classification score that computes *subset accuracy* in multilabel classification. Subset accuracy means that the set of labels predicted must exactly match the corresponding set of labels in y_true. If we say that for example the decision tree model was evaluated to be the best one, we can implement the code below to get the final result:

```
model = DecisionTreeClassifier()
model.fit(X_train, Y_train)
predictions = model.predict(X_test)
score = accuracy_score(Y_test, predictions)
```

2.5 Additional accuracy evaluation methods

In addition to accuracy_score, we're using two other methods for evaluating the accuracy of our classifications. These two methods are also in the sklearn.metrics library and are called "confusion_matrix" and "classification_report". A confusion matrix simply puts the correct predictions on the diagonal of a matrix and puts the wrong predictions elsewhere. This is to get a good visual overview of how many predictions failed and for which class. The classification report includes *precision*, *recall*, *f1-score,* and *support*.

Precision is the ability of a classifier not to label an instance positive that is actually negative. For each class, it is defined as the ratio of true positives to the sum of a true positive and false positive. In other

words it's the accuracy of positive predictions. Precision = TP / (TP + FP), where TP is short for True Positive and FP is short for False Positive **[9]**.

Recall is the ability of a classifier to find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives. Recall = TP / (TP + FN) **[9]**.

The **F1 score** is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy. F1 score = 2 * (Recall * Precision) / (Recall + Precision) **[9]**.

Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing. Support doesn't change between models but instead diagnoses the evaluation process itself **[9]**.

3 Results and discussion

We have two sets of results we can look at and compare. The results from when we dealt with the missing dataset values with solution 1, and when we fixed the dataset using solution 2. Within these two sets of results, we will find which model performed the best and which solution for fixing the dataset provided the highest accuracy scores for the models.

3.1 The SimpleImputer solution

First, we're going to take a look at the results from the algorithm evaluation process (Step 2) with the SimpleImputer solution for filling out the missing values in the dataset.

Algorithm	Cross_val_score	Std Dev
LR	0.986	0.0177
FFNN	0.346	0.0707
CART	0.956	0.0331
SVM	0.571	0.0349

Table 1: The score for each algorithm evaluated by 10-fold cross-validation for the SimpleImputer solution. The corresponding standard deviation for each score is on the right.

A visual representation of the same results can be seen in figure 8 below.

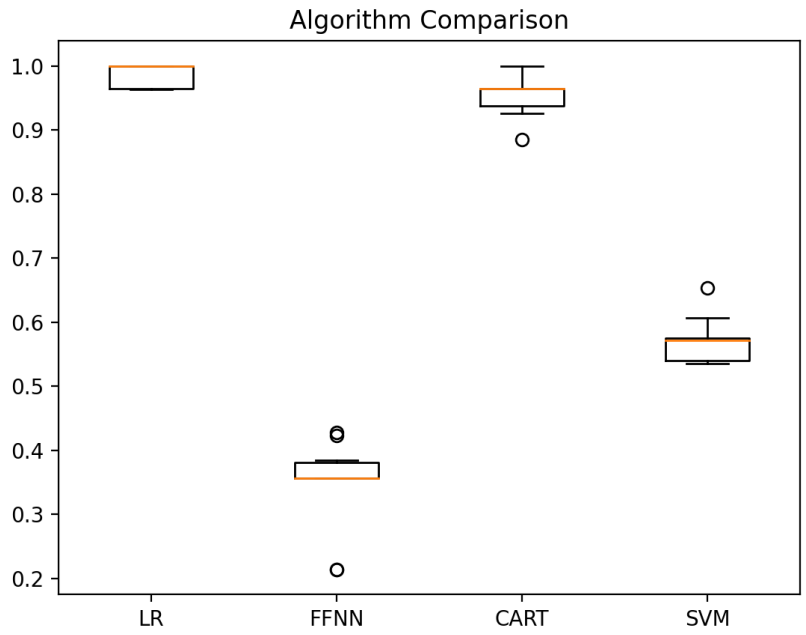

Figure 8: A boxplot comparing the various algorithms' cross_val_score using 10-fold cross-validation for the SimpleImputer solution. Logistic Regression comes out on top, with Decision Trees performing very good too.

What is interesting to note here is that the best performing algorithm also has the smallest standard deviation in the evaluation process. This gives us a strong suspicion that the Logistic Regression algorithm also will give us the best predictions on the validation set, but the Decision Trees algorithm is a solid contender too. The other two algorithms performed too poorly in the evaluation and we will scrap those for now.

3.1.1 Logistic Regression (SI)

The Logistic Regression algorithm gets an accuracy score of 0.986 on the validation set as well. This result remains the same even if we look at the mean of over 1000 runs. The reason why it gets the exact same score as on the training set becomes evident when we take a look at the confusion matrix $\begin{bmatrix} 33 & 1 & 0 \\ 0 & 11 & 0 \\ 0 & 0 & 24 \end{bmatrix}$. Only one prediction was wrong here, and the same thing must have happened during the evaluation with the 10-fold cross-validation method earlier. This is a very good result nonetheless, which we can confirm with the classification report in table 2 below.

Species	Precision	Recall	F1-score	Support
Adelie	1.00	0.97	0.99	34
Chinstrap	0.92	1.00	0.96	11
Gentoo	1.00	1.00	1.00	24

Table 2: The classification report for the Logistic Regression algorithm performed on the penguin dataset that was fixed with SimpleImputer.

We can see from the classification report in table 2 that the failed prediction of a Chinstrap penguin was actually a member of the Adelie species. This is a false positive. We see that the LR algorithm predicted the Gentoo penguins perfectly, which confirms what we expected from looking at figure 1 where the Gentoo data separates itself nicely from the other two classes in all of the graphs, making this species easier for the algorithms to classify correctly.

3.1.2 Decision Trees (SI)

The Decision Trees algorithm falls a bit behind with an accuracy score of 0.926 on the validation set. The corresponding confusion matrix $\begin{bmatrix} 29 & 5 & 0 \\ 0 & 11 & 0 \\ 0 & 0 & 24 \end{bmatrix}$ reveals five failed predictions. The mean score over 1000 runs is a slightly higher 0.957, but still not as good as LR. It seems very likely that the challenge here is to correctly classify Adelie penguins, as all of the members in the other two groups were classified correctly again. The classification report in table 3 points in the same direction.

Species	Precision	Recall	F1-score	Support
Adelie	1.00	0.85	0.92	34
Chinstrap	0.69	1.00	0.81	11
Gentoo	1.00	1.00	1.00	24

Table 3: The classification report for the Decision Trees algorithm performed on the penguin dataset that was fixed with SimpleImputer.

In table 3 we see that the Gentoo penguin predictions receive a perfect score, and it is the precision of the Chinstrap predictions with five false positives that belong in the Adelie category that pull the score down.

3.2 The predicted genders solution

Using the same method as for when we evaluated which algorithms best predicted the penguin species, we get these results from evaluating the algorithms based on predicting the missing genders:

Algorithm	Cross_val_score	Std Dev
LR	0.829	0.0823
FFNN	0.496	0.00741
CART	0.848	0.0787
SVM	0.624	0.0397

Table 4: The score for each algorithm evaluated by 10-fold cross-validation for the predicted genders solution. This is for finding which algorithm to use for best predicting the missing genders.

Again it's the same algorithms that performed the best, although these results were not as high as one could have hoped for. Testing the Logistic Regression and Decision Trees algorithms on the validation set

we get LR: 0.806 and CART: 0.940, so here we're going to use Decision Trees to predict the missing genders as it performed surprisingly well. CART predicted that six of the penguins with missing genders are females and three of them are males. Statistically, one of these is incorrect because of an accuracy score in the 0.90 region based on the model training, but it's close enough. Taking a look at the data in the table below gives us an idea of how accurate the predictions could be.

Species	Culmen length	Culmen depth	Flipper length	Body mass	Gender
Adelie	34.1	18.1	193	3475	FEMALE
Adelie	42.0	20.2	190	4250	MALE
Adelie	37.8	17.1	186	3300	FEMALE
Adelie	37.8	17.3	180	3700	FEMALE
Gentoo	44.5	14.3	216	4100	FEMALE
Gentoo	46.2	14.4	214	4650	FEMALE
Gentoo	47.3	13.8	216	4725	FEMALE
Gentoo	44.5	15.7	217	4875	MALE

Table 5: An overview of the predicted genders in comparison with the other attributes. The one prediction for the Chinstrap penguin is not included as there is nothing to compare it against.

The prediction for the Adelie male penguin can be said to be well-founded. Male penguins are generally larger than females. The predicted male has larger values for culmen length, culmen depth, and body mass. The prediction for the Gentoo male penguin can also be said to be well-founded. Here we see that the predicted male has larger values for culmen depth, flipper length, and body mass.

Now that we have the predicted genders in order, we're going to take a look at the results from the algorithm evaluation process (Step 2), now with the predicted genders solution for filling out the missing values in the dataset.

Algorithm	Cross_val_score	Std Dev
LR	0.986	0.0238
FFNN	0.331	0.108
CART	0.975	0.0365
SVM	0.576	0.0581

Table 6: The score for each algorithm evaluated by 10-fold cross-validation for the predicted genders solution. The corresponding standard deviation for each score is on the right.

A seen in table 6, the evaluation yet again reveals our top two contenders to be the Logistic Regression algorithm and the Decision Trees algorithm, both having very high scores from the cross-validation. We will now see how these two algorithms perform on the validation data with the predicted genders solution.

3.2.1 Logistic Regression (PG)

The Logistic Regression algorithm pulls off an impressive accuracy score of 1.00 on the validation set. The confusion matrix $\begin{bmatrix} 29 & 0 & 0 \\ 0 & 11 & 0 \\ 0 & 0 & 29 \end{bmatrix}$ shows a slightly larger sample of Gentoo penguins in the validation set relative to the total population, which can be part of the reason why the predictions are perfectly accurate because Gentoo penguins have been the easiest to predict so far. Next is the classification report.

Species	Precision	Recall	F1-score	Support
Adelie	1.00	1.00	1.00	29
Chinstrap	1.00	1.00	1.00	11
Gentoo	1.00	1.00	1.00	29

Table 7: The classification report for the Logistic Regression algorithm performed on the penguin dataset that was completed with predicted genders.

As expected, all measures in the classification report (table 7) also get a 1.00 value since every penguin species was predicted correctly. However, by running the algorithm with different random states while train-test-splitting the dataset we see that we get one failed prediction when the Adelie penguin sample is larger: $\begin{bmatrix} 36 & 1 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 19 \end{bmatrix}$. This will still be hard to match with the Decision Trees algorithm in the next section.

3.2.2 Decision Trees (PG)

The Decision Trees algorithm does indeed fall a bit behind again with an accuracy score of 0.957 on the validation set this time. The corresponding confusion matrix $\begin{bmatrix} 26 & 3 & 0 \\ 0 & 11 & 0 \\ 0 & 0 & 29 \end{bmatrix}$ reveals three failed predictions. This further confirms the suspicion that the most difficult challenge here is to correctly classify Adelie penguins, as all of the members in the other two groups keep being correctly classified. The classification report is as follows.

Species	Precision	Recall	F1-score	Support
Adelie	1.00	0.90	0.95	29
Chinstrap	0.79	1.00	0.88	11
Gentoo	1.00	1.00	1.00	29

Table 8: The classification report for the Decision Trees algorithm performed on the penguin dataset that was completed with predicted genders.

In table 8 we see that the Gentoo penguin predictions receive a perfect score as usual, and it is the precision of the Chinstrap predictions that keep getting false positives that belong in the Adelie category.

But surprisingly, after simply double-checking with the validation set that had a higher Adelie sample, expecting to see even more failed predictions, we discover something strange instead. The confusion matrix from the Decision Trees algorithm ends up looking like this: $\begin{bmatrix} 37 & 0 & 0 \\ 0 & 13 & 0 \\ 0 & 0 & 19 \end{bmatrix}$. This goes against all assumptions previously made. The Decision Trees algorithm actually performs better with a larger sample of the more difficult to predict Adelie penguins. The winner out of our top two algorithms is not as clear cut anymore and requires further testing.

After 1000 runs with different random states while train-test-splitting and getting different validation sets we get the following results:

- LR accuracy score: 0.990
- CART accuracy score: 0.974

So in the end it is Logistic Regression that performs best, with an impressive 99% accuracy. The test run where the Decision Trees algorithm got a perfect score was very likely just an outlier.

4 Conclusion

In this project, we successfully developed a model for classifying species in the Palmer penguins dataset. We started off by discovering missing values in the dataset and implementing the SimpleImputer solution and the predicted genders solution. With SimpleImputer and using the most frequent values to complete the dataset, we found that the Logistic Regression algorithm ended up with an accuracy score of 0.986 followed closely by the Decision Trees algorithm's accuracy score of 0.957. When we predicted the missing values in the dataset with machine learning classification, we found that The Logistic Regression algorithm again performed best with an accuracy score of 0.990, followed by the Decision Trees algorithm score of 0.974. From these results, we can conclude that the best solution for filling out the missing values in the dataset was to predict the values with machine learning classification, as the best algorithms' accuracy scores were 0.4 – 1.7% better with this solution. The best model for classifying the Palmer penguins was the one that included the Logistic Regression algorithm for the species classification and the Decision Trees algorithm for the missing genders classification. CART was best at predicting the missing genders in the dataset with an accuracy score of 0.940 versus LR's accuracy score of 0.806.

The other two algorithms, Feedforward Artificial Neural Network and Support Vector Machines never got an evaluation accuracy score higher than 0.650 in any of the multiple tests we did and were therefore not used for this classification problem.

Future work:

Find out why there was such a huge difference in performance among the various machine learning algorithms used in this project. Particularly why FFNN performed so poorly on this dataset. It would also be interesting to test our model on a larger and perhaps more complicated dataset with more attributes and classes to see if it can maintain the accuracy score of 0.990.

5 Appendix

Here you can find the python codes used in addition to a test of input and output for each code:

https://github.com/gery2/FYS-STK4155---Project-3

6 References

1. Allison Horst, *Palmer penguins* 2020, accessed 14 December 2020: https://allisonhorst.github.io/palmerpenguins/

2. Wetteland, V. *Classification and Regression*, University of Oslo, Norway, November 8, 2020: https://github.com/gery2/FYS-STK-4155---Project-2/blob/main/Report/FYS-STK4155%20Project%202.pdf

3. Parul Pandey, *Palmer Archipelago (Antarctica) penguin data* 2020, accessed 14 December 2020: https://www.kaggle.com/parulpandey/palmer-archipelago-antarctica-penguin-data?select=penguins_size.csv

4. Noel Bambrick, *Support Vector Machines: A Simple Explanation* 2016, accessed 14 December 2020: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

5. Eugen Baeldung, *Multiclass Classification Using Support Vector Machines* 2020, accessed 14 December 2020: https://www.baeldung.com/cs/svm-multiclass-classification

6. Davuluri Hemanth Chowdary, *Decision Trees Explained With a Practical Example* 2020, accessed 14 December 2020: https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example

7. Great Learning Team, *Decision Tree Algorithm Explained with Examples* 2020, accessed 14 December 2020: https://www.mygreatlearning.com/blog/decision-tree-algorithm/

8. Todd D. Lyle, *Multilayer Perceptron (MLP)* 2020, accessed 14 December 2020: https://www.techopedia.com/definition/20879/multilayer-perceptron-mlp

9. Shivam Kohli, *Understanding a Classification Report For Your Machine Learning Model* 2019, accessed 14 December 2020: https://medium.com/@kohlishivam5522/understanding-a-classification-report-for-your-machine-learning-model-88815e2ce397

10. John McGonagle, *Backpropagation* 2020, accessed 14 December 2020: https://brilliant.org/wiki/backpropagation/

11. Wetteland, V. *Regression analysis and resampling methods*, University of Oslo, Norway, October 10, 2020: https://github.com/gery2/FYS-STK4155---Project-1/blob/main/Report/1602331789273_FYS-STK4155%20-%20Project%201.pdf