



Cardiff University Computer Science and Informatics

CM3203 – One Semester Individual Project

Final report – 40 Credits

Species Distribution Modelling with Machine Learning

Author - Laura Edwards

Supervisor - Christopher Jones

Moderator - Jing Wu

Abstract

This paper studies environmental features and social media for predicting the presence of wildlife species in the UK using supervised machine learning classifiers. Ground truth for the eleven species distributions used was obtained from the National Biodiversity Network Atlas. The environmental features recognised include land cover, human population, emissions, mean temp, wind speed and rainfall, along with the social media platform Flickr. This data was obtained, processed and used as input into the classifiers. Six different machine learning algorithms were implemented and tested in this project. One of the main goals was to identify evaluation metrics to get an overall view of the performance of each model, making them easily comparable, thus identifying the most effective algorithm at correctly predicting the labels. A second goal was to compare three different grid cell sizes to test how the granularity affects the results. The final goal was to experiment with the impact the environmental features had on performance and identifying the most effective. The results deemed Random Forest to be the optimum performer consistently with Neural Networks and Support Vector Machines a close second. The grid cell sizes appear to follow a correlation that as the size increases, so does its accuracy. Lastly, the environmental features which proved to give the best performance individually, therefore showing how they consistently had a direct impact on species occurrence, was land cover. However, a combination of climate data proved to outperform other features.

Acknowledgements

I would like to firstly thank my supervisor Christopher Jones for his engagement and support during this project. Donating both his time and knowledge, without which I wouldn't have been able to successfully complete this work within the given time.

I am also thankful to my family and friends for their unconditional support and encouragement throughout the completion of this project in these unprecedented times.

Table of Contents

ABSTRACT	2
ACKNOWLEDGEMENTS.....	2
CHAPTER 1: INTRODUCTION	7
1.1. MOTIVATION	7
1.2. PROJECT AIM AND SCOPE.....	7
1.3. INTENDED AUDIENCE.....	8
1.4. ASSUMPTIONS	8
1.5. METHODOLOGY	8
1.6. REPORT STRUCTURE	9
CHAPTER 2: BACKGROUND.....	10
2.1. COORDINATE SYSTEM.....	10
<i>2.1.1. Latitude and Longitude.....</i>	<i>10</i>
<i>2.1.2. Easting and Northing</i>	<i>10</i>
2.2. NATIONAL BIODIVERSITY NETWORK (NBN) ATLAS.....	11
2.3. UK BIODIVERSITY ACTION PLAN (BAP).....	12
2.4. FLICKR API.....	12
2.5. MACHINE LEARNING ALGORITHMS.....	13
<i>2.5.1. Supervised vs Unsupervised.....</i>	<i>15</i>
<i>2.5.2 Fitting</i>	<i>16</i>
<i>2.5.3 Decision Trees.....</i>	<i>17</i>
<i>2.5.4. Random Forest Classifier.....</i>	<i>17</i>
<i>2.5.5. Support Vector Machines (SVM)</i>	<i>18</i>
<i>2.5.6. K- Nearest Neighbors.....</i>	<i>19</i>
<i>2.5.7. Naïve Bayes.....</i>	<i>19</i>
<i>2.5.8. Neural Network Classifier</i>	<i>20</i>
<i>2.5.9. Feature Scaling</i>	<i>21</i>
2.6. EVALUATING THE MODEL	21
<i>2.6.1. Confusion Matrix.....</i>	<i>21</i>
<i>2.6.2. Accuracy.....</i>	<i>22</i>
<i>2.6.3. Precision.....</i>	<i>22</i>
<i>2.6.4. Recall.....</i>	<i>23</i>
<i>2.6.5. F1-Score</i>	<i>23</i>
2.7. PYTHON LIBRARIES	23
2.8. RELATED WORK	24

CHAPTER 3: APPROACH	25
3.1. PROJECT REQUIREMENTS	25
3.1.1. <i>Functional Requirements</i>	<i>26</i>
3.1.2. <i>Non-functional Requirements</i>	<i>26</i>
3.2. PROJECT STRUCTURE	27
3.3. "PROCESSDATA" – DATA COLLECTION + DATA PROCESSING.....	27
3.3.1. <i>Wildlife Selection and NBN Data</i>	<i>27</i>
3.3.2. <i>Flickr API.....</i>	<i>28</i>
3.3.3. <i>Environmental Features.....</i>	<i>30</i>
3.3.4. <i>Process Data using Grid.....</i>	<i>34</i>
3.4. "CLASSIFIERS" – MACHINE LEARNING	35
3.4.1. <i>"Classifiers" Folder Overview.....</i>	<i>35</i>
3.4.2. <i>Machine Learning Process Overview</i>	<i>36</i>
3.5. DEVELOPMENT STRATEGY AND METHODOLOGY	37
CHAPTER 4: IMPLEMENTATION	37
4.1. OVERVIEW	37
4.2. DATA COLLECTION AND PRE-PROCESSING	38
4.2.1. <i>Flickr Data Collection</i>	<i>38</i>
4.2.2. <i>Process NBN Data</i>	<i>39</i>
4.2.3. <i>Process Environmental Data</i>	<i>42</i>
4.3. MACHINE LEARNING CLASSIFICATION MODELS	45
4.3. ENVIRONMENTAL FEATURE IMPORTANCE	52
CHAPTER 5: TESTING	52
CHAPTER 5: RESULTS AND EVALUATION	57
5.1. CLASSIFICATION MODELS PERFORMANCE.....	57
5.1.1. <i>Accuracy.....</i>	<i>58</i>
5.1.2. <i>Precision.....</i>	<i>59</i>
5.1.3. <i>Recall.....</i>	<i>59</i>
5.1.4. <i>F1- Score</i>	<i>60</i>
5.1.5. <i>Grid Cell Sizes.....</i>	<i>61</i>
5.2. ENVIRONMENTAL FEATURES.....	62
5.1.5. <i>Species Environment- Further Analysis</i>	<i>63</i>
CHAPTER 6: FUTURE WORKS	64
CHAPTER 7: CONCLUSION	65
CHAPTER 8: REFLECTION ON LEARNING.....	66

Table of Figures

Figure 1: Agile Methodology	9
Figure 2: Longitude and Longitude	10
Figure 3: Easting and Northing	11
Figure 4: NBN Atlas map.....	11
Figure 5: NBN CSV Output File.....	11
Figure 6: BAP Example Priority List	12
Figure 7: Flickr API XML Output.....	13
Figure 8: Machine Learning Process.....	14
Figure 9: Process of Supervised Learning	15
Figure 10: Classification and Regression Graphs	15
Figure 11: Process Unsupervised Learning	16
Figure 12: Fitting Graphs	16
Figure 13: Decision Tree Example	17
Figure 14: Random Forest Diagram	18
Figure 15: Support Vector Machine Diagram	18
Figure 16: K-Nearest Neighbors Diagram	19
Figure 17: Conditional Probability Equation	19
Figure 18: Naive Bayes Diagram.....	20
Figure 19: Multilayer Perception Diagram	21
Figure 20: Standard Scaling equation.....	21
Figure 21: Confusion Matrix.....	22
Figure 22: Accuracy Equation.....	22
Figure 23: Precision Equation.....	22
Figure 24: Recall Equation	23
Figure 25: F1-Score Equation	23
Figure 26: Project Split Diagram	27
Figure 27: Species Data Flow.....	28
Figure 28: Species Flickr data flow	29
Figure 29: Temperature data flow.....	30
Figure 30: Rainfall data flow.....	31
Figure 31: Wind Speed data flow	31
Figure 32: Land Cover Dominant Class data flow	32
Figure 33: Land Cover Subclass data flow	33
Figure 34: Emissions data flow	33
Figure 35: Human population data flow.....	33
Figure 36: Grid supporting diagram	35
Figure 37: "Classifiers" folder overview.....	35
Figure 38: Machine Learning Solution.....	36
Figure 39: Flickr Connection.....	38
Figure 40: Flickr Script Diagram.....	38
Figure 41: Flickr Data Collection	39
Figure 42: Create file of Easting and Northing.....	40
Figure 43: Convert Long and Lat to Easting and Northing	40

Figure 44: Create 20km Grid	40
Figure 45: Call cell by ID.....	41
Figure 46: Get Cell Count.....	41
Figure 47: Create new grid CSV file	42
Figure 48: Plot ground truth data.....	42
Figure 49: Convert ASCII Format	43
Figure 50: Add values to CSV diagram.....	43
Figure 51: Most Frequent.....	43
Figure 52: Get Cell Value	44
Figure 53: Loading Datasets.....	45
Figure 54: Join data frames and drop columns.....	46
Figure 55: Drop rows with a -9999 value	46
Figure 56: Describe data frame.....	46
Figure 57: Split features and labels	46
Figure 58: Experiment train and test split.....	47
Figure 59: train-test split	47
Figure 60: Initialise data frame.....	47
Figure 61: Feature Scaling.....	47
Figure 62: Naive Bayes Classifier train and predict	48
Figure 63: Confusion Matrix implementation.....	48
Figure 64: Evaluation metrics.....	48
Figure 65: Experiment with SVM Kernels.....	49
Figure 66: Support Vector Machine Classifier train and predict.....	49
Figure 67: Decision Tree Classifier train and predict.....	49
Figure 68: Experimenting with K value.....	50
Figure 69: K-Nearest Neighbor Classifier train and predict	50
Figure 70: Random Forest Classifier train and predict.....	50
Figure 71: Neural Network Classifier train and test	51
Figure 72: Save data frame to pickle	51
Figure 73: Read in pickle for each species	51
Figure 74: Join data frames	51
Figure 75: Mean for classifier	52
Figure 76: Individual Environmental Feature	52
Figure 77: 30km Classifier Results	57
Figure 78: 20km Classifier Results	58
Figure 79: 10km Classifier Results	58
Figure 80: Bard Chart of accuracy for 10, 20, 30km for each classifier	58
Figure 81: Bar Chart of Precision for 10, 20, 30km for each classifier	59
Figure 82: Bar Chart of Recall for 10, 20, 30km for each classifier	60
Figure 83: Bar Chart of F1-Score for 10, 20, 30km for each classifier	61
Figure 84: Environmental Features Results.....	62
Figure 85: Harvest Mouse Environment Features Results	63
Figure 86: Red Grouse Environmental Feature Results.....	64

Chapter 1: Introduction

1.1. Motivation

I have always had a keen interest in the outdoors and a fascination with wildlife programs which meant I was enthusiastic about undertaking a project such as this which incorporates my interests along with my degree. Over the past few years, there has been rapid development in the area of species distribution modelling, which is hugely due to new techniques used for collating the data. Without these advancements, I would have been unable to obtain the data necessary to take on this project. The presence of wildlife is regarded to be heavily impacted by the environment it belongs in. My motivation was to test how well I could predict their occurrences with what I deemed as important environmental information, along with the use of available social media data. The use of social media data is interesting as the photos are posted by the general public, therefore reliant on them accurately being able to identify a species.

1.2. Project Aim and Scope

The projects' scope focuses on how machine learning can be utilised for providing an accurate as possible predictions on the presence of a given species based on the data extracted for environmental features and from social media. Based on this, there are several aims for this project which were previously mentioned in my initial plan. Firstly, these include implementing five different machine learning algorithms and one deep learning algorithm with appropriate evaluation techniques to make them easily comparable. A second aim is to compare three different grid cell sizes to test how the granularity affects its accuracy at predicting a species' presence. Thirdly, I aim to experiment with the impact the environmental features have on performance and different combinations to identify the most practical and effective features. From this project, the results found can be used to illustrate environmental features that have greater impact on the presence of species as well as identifying which classifier was proven to achieve a prediction to the highest degree of accuracy given the data inputted. The reasoning behind applying machine learning for predicting a species occurrence is that the accuracy of the model will improve when given more environmental features to learn from.

1.3. Intended Audience

The intended audience for this project are individuals or organisations who are interested in learning how to implement machine learning algorithms as species occurrence predictors and which model proves to be the most successful along with what grid cell size. Also, anyone interested in wildlife species and what environmental features have the greatest impact on predicting their presence. I imagine wildlife enthusiasts will be the most intrigued.

1.4. Assumptions

There were a handful of assumptions made through the completion of this project. Firstly, the data extracted from the National Biodiversity Network Atlas is factual and suitable to use as ground truth data. Secondly, the photos collated using the Flickr API are in fact accurate images of the said species passed. Thirdly, the environmental data collated is a correct representation.

1.5. Methodology

I used an agile methodology to complete this project as opposed to a traditional waterfall approach. Due to this being my first machine learning project, I found agile to be a more appropriate choice as it offers a flexible nature, allowing me to have an adaptive rather than a predictive approach. Agile is defined as an iterative development model. Therefore, for this project, I broke it into several phases with different requirements and solutions. Thus when new information came to light that needed investigating, it was easily incorporated into the subsequent or future iterations. For this project, I found it suitable to work in fortnightly sprints. The initial stage of an agile methodology is to define developments that need to be undertaken, then to develop a solution that meets the requirements that have been previously defined and utilise testing to certify they work as expected. By holding meetings to discuss and evaluate the sprints results I was able to define the new developments for the next sprint. This approach resulted in avoiding any unnecessary work and allowed me to focus on priority tasks to meet the projects main objectives. [32].



Figure 1: Agile Methodology

1.6. Report Structure

I have structured my report into the recommended sections which are as follows:

Chapter 1: Introduction - An introduction giving basic insight on what the project entails.

Chapter 2: Background - Descriptive information on the background knowledge needed in order to fully understand my report.

Chapter 3: Approach - Provide a clear picture of the system I have created, including what the software system does (specification) and how it does so (design).

Chapter 4: Implementation - A finer analysis of how I developed the system including code examples from my implementation.

Chapter 5: Test Cases - Testing of the main components to certify they work as expected and their output is correct.

Chapter 6: Results + Evaluation - Demonstration that the project meets the requirements which were aimed for and results of the tests executed.

Chapter 7: Future Work - These are ideas I had for future work I would like to accomplish on the same basis of this project but was unable to complete given the available timeframe.

Chapter 8: Conclusion - A synopsis of my aims, objectives and a reiteration of the results achieved.

Chapter 9: Reflection - Reflect on the project and what I have learnt throughout the process which will be invaluable to carry forward. [33].

Chapter 2: Background

This chapter will give you an overview of some important concepts that are essential to understanding my solution to this project. This includes the coordinate systems utilised and species specific data retrieval. Also, a detailed explanation of the Machine Learning algorithms used and the metrics to evaluate their performance. Penultimately, a list of the Python libraries implemented with a brief description of their purpose. Lastly, existing research papers that I have read relating to this project and have given me insight.

2.1. Coordinate System

A major requirement for this project was to split the UK into grid cell sizes and gather data for all points within that cell. Thus, knowledge and understanding of the coordinate system is required.

2.1.1. Latitude and Longitude

Latitude and longitude is a coordinate system that can be used to calculate and define the direction and orientation of any point on the Earth's Surface. On a globe or map, latitude refers to the distance north or south of the equator and is specified by degrees between 0 and 90. The equator is where latitude equals 0 degrees. Longitude is an angle between 0 and 180 degrees pointing east or west from the Greenwich Meridian. [1].

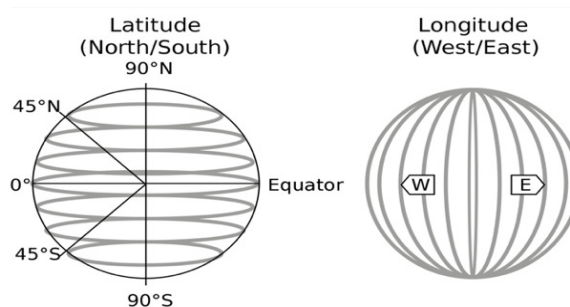


Figure 2: Longitude and Longitude

2.1.2. Easting and Northing

Easting and Northing are a geographic Cartesian coordinate system for any given point. Easting is the eastward-measured distance, also known as the x-coordinate. Northing is the northward-measured distance known as the y-coordinate. These coordinates are commonly measured in meters from the axis of the horizontal datum. For this project, I used the Ordnance Survey National Grid reference (OSGB36) system, which is used for Great Britain. [2]

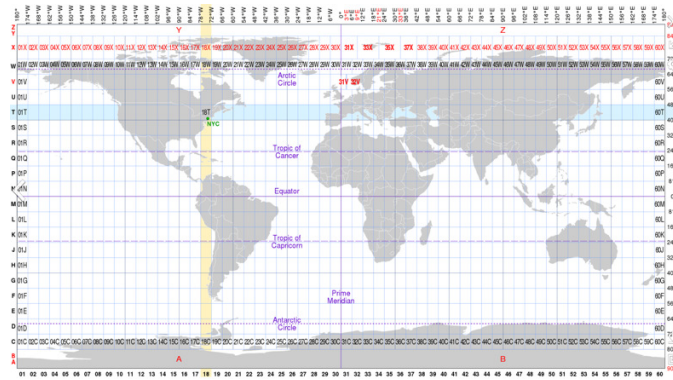


Figure 3: Easting and Northing

2.2. National Biodiversity Network (NBN) Atlas

The NBN is a registered charity that advocates biological data being shared since 2000. Their aims are to improve data availability and quality to provide evidence for a base on environmental decisions. The NBN Atlas provides an online tool to engage, educate and inform people about the natural world. It aids the improvement of biodiversity awareness, the expansion of research opportunities and the transformation of environmental conservation in the UK. The NBN Atlas is innovative because it is the first time various sources of knowledge about UK species and habitats, as well as the opportunity to interrogate, integrate, and analyse the data, all in one place. Its aim is to make the learning and understanding of wildlife in the UK less complicated. [3]

NBN has a feature that plots each of their specific species' occurrences on a map, and from there, you can download a CSV file. The CSV file provides detail on each point which was used in this project as my ground truth as to whether a species is absent or present in a given grid cell.

64,326 records (64,326 in total)

This map contains both point- and grid-based occurrences at different resolutions

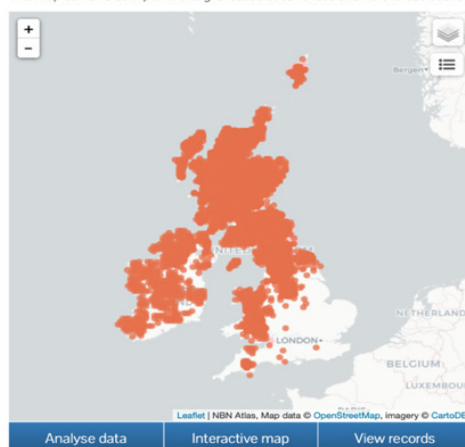


Figure 4: NBN Atlas map

Occurrence ID	Licence	Rightsholder	Scientific name	Taxon author	Name qualifier	Common name	Species ID (TVK)	Taxon Rank	Occurrence st
15301911	OGL	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
125497859	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
123394879	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
102812580	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
57859331	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
72635170	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
57474926	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
66161876	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
80712211	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
114781848	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
72924836	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
116882132	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
123394880	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
59402895	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
51200352	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present
248701162	CC-BY-NC	BTO	Lagopus lagopus	(Linnaeus, 1758)		Red Grouse	NHMSYS0000530420	species	present

Figure 5: NBN CSV Output File

2.3. UK Biodiversity Action Plan (BAP)

For this project, the species used were selected from the UK Biodiversity Action Plan (BAP). The UK BAP priority is a list of 1,150 species deemed as the most threatened and requiring protective action. The amended list was produced in 2007 and split into several groups. The following groups include birds, terrestrial mammals, herptiles and fish etc. [4]. In this project, the groups used are birds, mammals, reptiles and amphibians. Birds are warm-blooded vertebrates with a feather-covered body, wings, scaly legs, beak, no teeth, and bear their young in a hard-shelled egg. [26]. Mammals are warm-blooded, breathe air, have hair, give birth to live young and secrete milk to feed them. [27]. Reptiles are cold-blooded, air-breathing vertebrates covered in scales or bony plates. [28]. Finally, amphibians are small cold-blooded vertebrates that need water and a moist environment to survive. [29]. This UK BAP priority species list provides the scientific name, common name and if they occur in the four UK countries, as seen in the figure below.

Scientific name	Common name	Taxon	England	Scotland	Wales	Northern Ireland	Original UK BAP species?
<i>Acrocephalus paludicola</i>	Aquatic Warbler	bird	Y	N	Y	N	Yes – SAP
<i>Acrocephalus palustris</i>	Marsh Warbler	bird	Y	Y	Y	N	Yes – SAP
<i>Alauda arvensis</i> subsp. <i>arvensis/scotica</i>	Sky Lark	bird	Y	Y	Y	Y	Yes – SAP
<i>Anser albifrons</i> subsp. <i>albifrons</i>	European Greater White-fronted Goose	bird	Y	N	N	N	
<i>Anser albifrons</i> subsp. <i>flavirostris</i>	Greenland Greater White-fronted Goose	bird	N	Y	Y	Y	
<i>Anthus trivialis</i>	Tree Pipit	bird	Y	Y	Y	N	

Figure 6: BAP Example Priority List

2.4. Flickr API

Flickr is a self-proclaimed best photo sharing online application in the world. Its initial launch date was in 2004. The services they provide to their customers is a platform where they can post and share their photos to friends and family and offer them a way to organise their photos. It is unique to other sites with similar concepts by using tags to describe images along with the option of geotags. [5]

The benefits of these geotags are being able to search by location. In 2009 it was reported that “100 million photos posted by users contained geotags, which equates to around 33% of the total photos.” [6]

The Flickr API was used to build a dataset of photos that will be pre-processed for machine learning. To gain access to the API, you need to obtain a key from Flickr. This involves creating an account and signing up with an explanation of how you intend to use this information. The key can be either commercial or non-commercial, referring to whether you are making money from your application. Hence non-commercial is most suitable for this project. Once the accounts are set up an API Key and secret will be provided. Available methods are very well documented.

Flickrapi is a python library designed to access the Flickr API to establish a connection with Flickr and stream available data. Photos gathered using flickrapi are in an XML format, as shown below.

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
  <photos page="1" pages="1" perpage="250" total="227">
    <photo id="51124190465" owner="23882583EN08" secret="2ab8cf6ba2" server="65535" farm="66" title="I
    <photo id="51115584962" owner="66190370EN05" secret="d4c14f6f08" server="65535" farm="66" title="I
    <photo id="51115584932" owner="66190370EN05" secret="8aa77fd547" server="65535" farm="66" title="I
    <photo id="51115584847" owner="66190370EN05" secret="d2ec9273e6" server="65535" farm="66" title="I
    <photo id="51068002383" owner="31693460EN06" secret="b65f135809" server="65535" farm="66" title="I
    <photo id="51065203018" owner="31693460EN06" secret="ca0d090527" server="65535" farm="66" title="I
    <photo id="51062386326" owner="31693460EN06" secret="e5fee554e6" server="65535" farm="66" title="I
    <photo id="51041605278" owner="95795254EN02" secret="1eef9f6e34" server="65535" farm="66" title="I
    <photo id="50776084711" owner="67356167EN05" secret="6a578615ef" server="65535" farm="66" title="I
    <photo id="50599259528" owner="95795254EN02" secret="81b7a88ee0" server="65535" farm="66" title="I
    <photo id="50568313283" owner="95795254EN02" secret="4becb37061" server="65535" farm="66" title="I
    <photo id="50381334568" owner="78465328EN04" secret="7d6160cf08" server="65535" farm="66" title="I
    <photo id="50353914223" owner="35884794EN04" secret="5434386181" server="65535" farm="66" title="I
    <photo id="50179935207" owner="46262145EN08" secret="80b8d96582" server="65535" farm="66" title="I
    <photo id="50171207006" owner="34518162EN08" secret="0bb8ac2840" server="65535" farm="66" title="I
    <photo id="50102443247" owner="90555326EN05" secret="6015a95f84" server="65535" farm="66" title="I
    <photo id="48701066333" owner="67356167EN05" secret="5d2dfe4fd4" server="65535" farm="66" title="I
    <photo id="47945411406" owner="63774863EN05" secret="8ab638045d" server="65535" farm="66" title="I
    <photo id="32552259367" owner="34518162EN08" secret="95f3b7ca07" server="7902" farm="8" title="Ma
    <photo id="46969052471" owner="67356167EN05" secret="7f5633d48b" server="4897" farm="5" title="Ma
    <photo id="42174595980" owner="34705819EN05" secret="32e3c1e530" server="65535" farm="66" title="Rec
    <photo id="43340062442" owner="58519338EN08" secret="3d757403c0" server="1788" farm="2" title="Rec
    <photo id="41932884641" owner="67356167EN05" secret="16976180ea" server="962" farm="1" title="Red
```

Figure 7: Flickr API XML Output

2.5. Machine Learning Algorithms

Machine Learning (ML) is a branch of Artificial Intelligence (AI) and results from the expeditious development of data mining techniques and methods. The last decade has seen increasing popularity in ML, enabling both organisations and individuals to have insight and understanding of their datasets. [7]. There are many ML algorithms which in brief are a sequence of statistical processing steps where they are trained to discover patterns in vast volumes of data. The data can be in the format of numbers, words or images as long as it is digitally stored. The better the algorithm, the more accurate the results. The key disparity between ML models and computer programs is the lack of

intervention needed by developers to instruct the system. From this, Arthur Samuels coined the definition of machine learning as the “computers ability to learn without being explicitly programmed”. [8]. Therefore, it is worth looking through the general workflow of the machine learning process to get a more profound understanding.

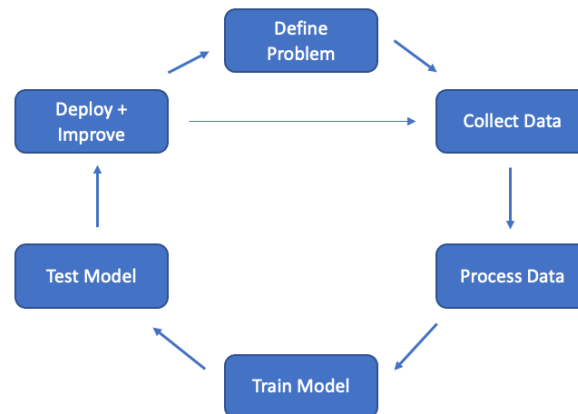


Figure 8: Machine Learning Process

Define Problem – Before collecting data or implementing, start by clearly identifying the problem and goal.

Collect Data – The dataset can be populated by data from databases, spreadsheets or text files. The more diverse and extensive the data used is, the higher accuracy the machine learning model's findings would be.

Process Data – This process entails analysing, cleaning and reformatting the data gathered into a much more desirable form that the classifiers can utilise. After that, it can be divided into two sets: Training and Testing. The training set is used to build the machine learning model, while the testing set is used to assess it.

Train Model – Using the training data, a model will be constructed using a defined algorithm. Identifying trends and similarities, as well as making assumptions.

Test Model – Using the testing data, the model will make predictions and its accuracy assessed using chosen metrics. It is vital to validate the model using an unseen dataset in order to ensure an unbiased evaluation.

Deploy and Improve – This process entails experimenting with various algorithms and collecting a wider variety of data in order to refine predictions before a suitable model is found. To summarise, the vast majority of machine learning processes are never ending since there is still space for progress in the future as new evidence becomes available or circumstances change.

2.5.1. Supervised vs Unsupervised

Supervised machine learning algorithms is where it takes the desired input variables 'x' and a desired output variable 'y'. A selected algorithm is used to approximate the mapping by discovering how to arrive at those variables by identifying patterns. Ideally, this mapping function will be used for new input data 'x' to predict the output variable 'y' accurately. Supervised learning obtained its name through the process of an algorithm learning from the training data, which is comparable to a teacher supervising the students learning process. We know the accurate answers, so the algorithm iterates over the training data, making assumptions that are then corrected by the teacher. When the algorithm reaches a satisfactory level of accuracy, learning comes to an end. Supervised machine learning algorithms are grouped into either classification or regression, the difference being the nature of their output variables. [9].

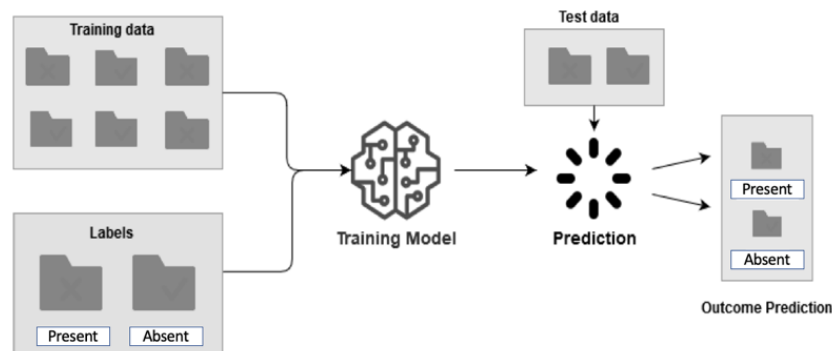


Figure 9: Process of Supervised Learning

Classification Models are utilised when the output is a category e.g. "Present" or "Absent". On the other hand, Regression Models are utilised when the output is a real number e.g. size, weight. [9].

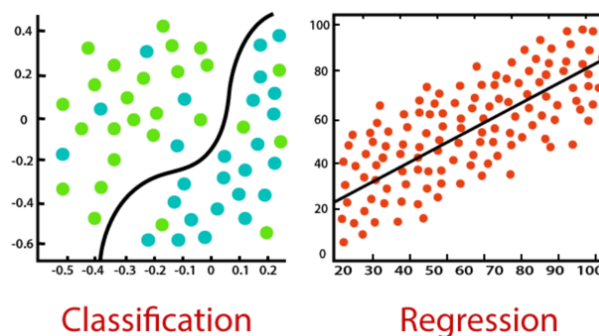


Figure 10: Classification and Regression Graphs

Unsupervised machine learning algorithms is where it only takes input variables 'x' and has no corresponding output variable. Thus, the target variable 'y' is absent from the

data, so prediction is not the goal. Therefore, we are trying to find structure and patterns in the data. In conclusion, supervised and unsupervised machine learning algorithms differ mainly in the way the data is labelled. Unsupervised machine learning algorithms are further grouped into clustering. [9].

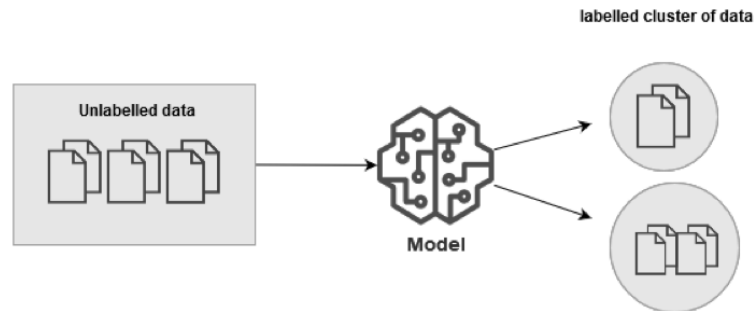


Figure 11: Process Unsupervised Learning

Clustering models try to find patterns in the unlabelled data and group the data into clusters according to similarity. e.g. grouping customers by purchasing behaviour. [9].

2.5.2 Fitting

Model “fitting” refers to the chosen machine learning algorithms effectiveness at generalising to similar data on which it was trained. To get the desired outcome, it is imperative to be neither overfitted nor underfitted otherwise, we run the risk of poor performance. Two essential terms, variance and bias, should both be considered when examining the fit of the model. When model fitting, bias indicates any assumptions made by the model to make a target. If the model has made a high number of assumptions about the target function, it will have a high bias and vice versa. Variance refers to the models’ dependence on the data it is trained with. If a model pays high attention to the training data, it will have a high variance. A good/robust fitted model will have a balance between bias and variance. An overfitted model will have a low bias and high variance, meaning the model performs well on training data but does not perform well on evaluation data. An underfitted model will have a high bias and low variance, meaning the model performs poorly on training data. [10]. [30].

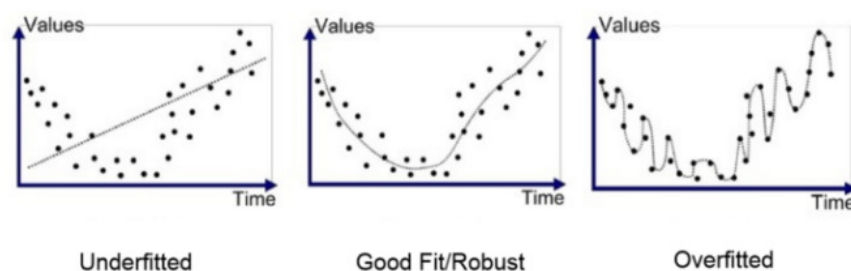


Figure 12: Fitting Graphs

2.5.3 Decision Trees

Decision Trees (DT) is a popular algorithm capable of performing both regression and classification tasks. DT applies the learning concept of divide and conquer. It has a tree-like structure where the internal nodes correspond to features, branches represent the decision rules and leaf nodes for the class label. When predicting the class for the given data, it starts at the root node, which is situated at the top of the tree. It compares the inputted data with values of the root and follows the corresponding branch and so on until a leaf node is met with the predicted class value. A basic example to illustrate my explanation is a question of where someone is fit? Firstly they check the “age” at the root and if it is less than 30. If “Yes” it follows the branch to “Eat’s a lot of pizzas?” or if “No” it corresponds to “Exercises in the morning?”. This routine continues until the leaf node is met and the class is found. [11].



Figure 13: Decision Tree Example

2.5.4. Random Forest Classifier

Random Forest (RF) is one of the most popular classifiers due to its simplicity and high accuracy. RF consists of an ensemble of several decision trees hence, the name “forest”. Also, “random” as training data points are split into random samples when building trees. Therefore, these trees are learning from unique samples of data and trained in parallel, this is referred to as “bootstrapping”. The plan is that by training each tree on slightly different observation samples it will lead to a lower variance without increasing bias. Once tested, the predictions are aggregated from each decision tree which is referred to as “bagging”. Advantages of RF include its efficiency in performing on large datasets and accuracy. [12].

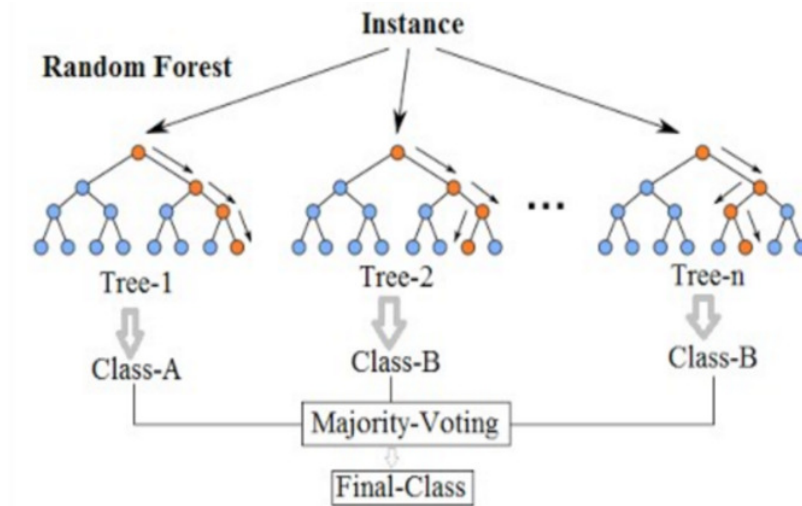


Figure 14: Random Forest Diagram

2.5.5. Support Vector Machines (SVM)

Support Vector Machine (SVM) algorithm is commonly used for solving classification problems, although it can be used for regression. The aim of this algorithm is to successfully get a hyperplane, which is like a boundary to effectively separate the two groups of data points that are plotted in an n-dimensional space, where n is the number of features. There will be an assortment of different hyperplanes that can be used to separate the data points, but a well-performing hyperplane will minimise the distance between the points and itself. This distance is called margin. The one with the largest margin for both classes should be selected. Support vectors are the points that are located closer to the hyperplane. Thus those are the ones chosen to define its position and orientation to maximise margin. There is a selection of different SVM mathematical algorithms, which are called kernels. A kernels purpose is to take the given inputted data and convert it into the required form. For this project, I used a Linear Kernel. [13].

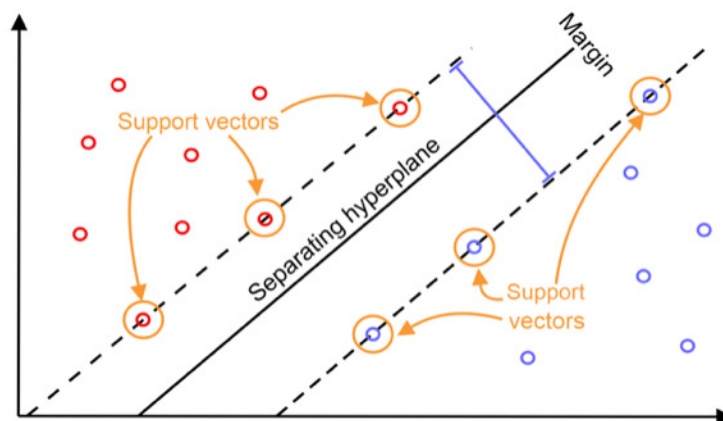


Figure 15: Support Vector Machine Diagram

2.5.6. K- Nearest Neighbors

Although K-Nearest Neighbors (KNN) is one of the most straightforward algorithms and still achieves high accuracy. KNN makes no assumptions about the underlying data structure, making it a non-parametric algorithm. This model may also be considered as a lazy learner algorithm due to its nature in handling training data. It does not learn from it immediately. Instead it is stored until needed for classification, and only then performs an action on the data. The “K” is an integer that indicates the number of nearest neighbouring data points that will be used. The neighbours hold a majority vote for which class to be predicted. For instance, if the given “K” value is 5, then the five nearest points to that point will be used to make the decision. So it is imperative to choose the correct value for “K “, which can be achieved through trial and error whilst selecting the best fit. However, calculating the distance between the data points for all the training data comes with the disadvantage of computation costs. [14].

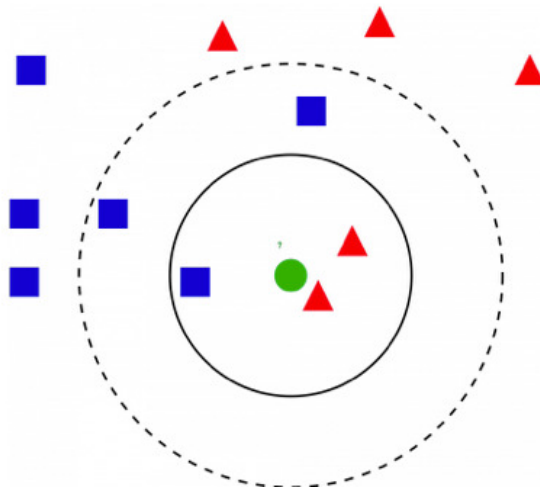


Figure 16: K-Nearest Neighbors Diagram

2.5.7. Naïve Bayes

Naïve Bayes is a classifier based on Bayes Theorem, which works on conditional probability. The name “naïve” comes from its assumption that all the features are independent of each other. Conditional probability is the probability of an event occurring given that something has already happened. [15]. The mathematical equation for this is as follows:

$$P(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

Figure 17: Conditional Probability Equation

As the figure below will help illustrate, the classifier will be using the training data inputted, and the algorithm will learn the conditional probability of each feature on the class label 'C'. Bayes Theorem will then be applied when testing the algorithm to calculate the probability of a class using features A_1, A_2, \dots, A_n . The class is decided by the highest probability. A disadvantage to this classifier is the requirement of the features to be independent as it may overlook important correlations within the data. Positives are its fast performance which aids real-time predictions and how it is highly scalable with both predictors and data points. [15].

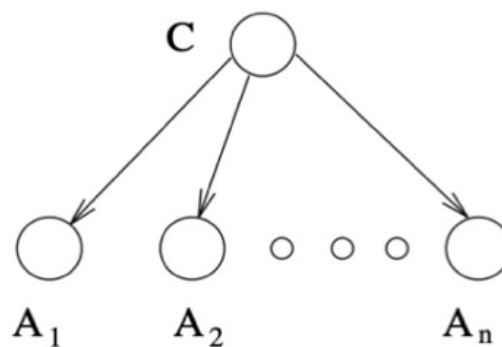


Figure 18: Naive Bayes Diagram

2.5.8. Neural Network Classifier

In this project, I used a basic algorithm of Deep Learning, Multilayer Perception (MLP). Deep learning is a subgroup of machine learning. This model is categorised as an artificial neural network. As you can see from the figure below, there are three main layers that are fully connected, input, hidden and output. At the input layer, each node correlates to one feature of the given dataset leading to the output layer where nodes represent the classes. MLP trains its data by carrying out a method called backpropagation. To explain further, learning happens after processing data and changing connection weights between neurons. This is based on a comparison between the degree of error and the expected result in the output layer. MLP may use an algorithm such as gradient descent in order to alter the weight, which will improve accuracy. Artificial neural networks behave similarly to a biological one however, it takes probabilistic data as inputs and converts it to output classes. [16].

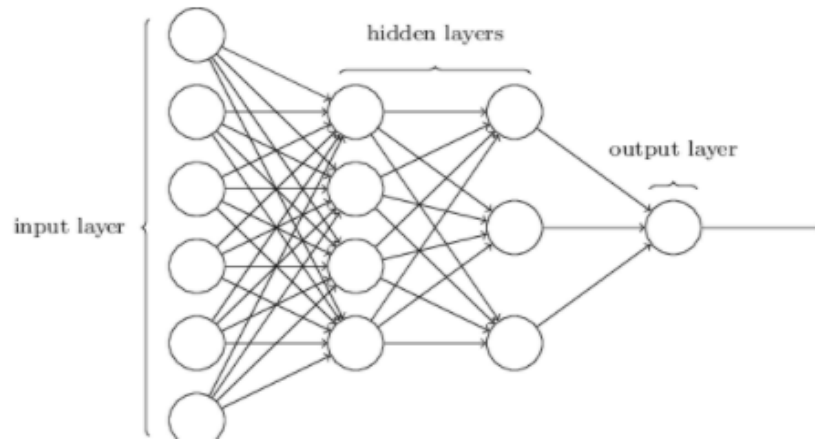


Figure 19: Multilayer Perception Diagram

2.5.9. Feature Scaling

Some features' raw data values can span varying degrees of magnitude and range which leads to a few algorithms not achieving their best performance, as they are sensitive to this data. The classifiers most affected by this raw data are algorithms based on distance and gradient descent, therefore a feature scaling technique is needed. This step is usually performed during the pre-processing stage. There are several available methods for scaling the data, this project utilises standard scaler. The end result of the scaling process will be attributes with a mean of 0 and a standard deviation of 1. Below is the equation used. [40].

$$X' = \frac{X - \mu}{\sigma}$$

Figure 20: Standard Scaling equation

2.6. Evaluating the Model

This section includes the evaluation techniques used in this project to assess and compare the performance of the implemented machine learning classifiers.

2.6.1. Confusion Matrix

A Confusion Matrix is a table that is frequently utilised to visualise the performance of a given classifier model by summarising prediction results on the test data. The possible Confusion Matrix results include:

- True Positive (TP) – Correctly predicted present
- False Positive (FP) – Predicted present, but it's actually absent
- True Negative (TN) – Correctly predicted absent

- False Negative (FN) – Predicted absent, but it's actually present

These four measures are used to show the effectiveness of the classifier. One axis of the table is labelled Actual, and the other axis is Prediction. A Confusion Matrix is of size $N \times N$, where N correlates to the number of classes. In this instance, it is a binary classification problem; thus, $N=2$.

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Figure 21: Confusion Matrix

2.6.2. Accuracy

Accuracy is defined as the number of accurate predictions divided by the total number of input samples, and it can also be expressed as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figure 22: Accuracy Equation

Accuracy alone is not perfect for giving the whole picture and can provide a false sense of accomplishment when there is a class-imbalanced data set like this project where there is a disparity between the number of present and absent labels. [17].

2.6.3. Precision

Precision is defined as the number of accurately predicted positive results divided by the number of positive results predicted by the classifier. [17]. It can be expressed as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Figure 23: Precision Equation

2.6.4. Recall

Recall is defined as the number of accurately predicted positive results divided by the total number of positive results that should have been detected. [17]. It can be expressed as follows:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Figure 24: Recall Equation

2.6.5. F1-Score

F1-Score defines the number of instances the classifier correctly classified, also robustness. This is a comprehensive assessment of the models accuracy, with the primary goal of striking a balance between precision and recall. [17].

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Figure 25: F1-Score Equation

2.7. Python Libraries

My chosen programming language was Python for the implementation of this particular project as it is the most suitable for data science. Additionally, Python is widely popular with a large community of developers making support easier to come by. Python has many well-documented libraries, which are prewritten functions that solve common programming tasks thus, helping to reduce programming time. A number of Python libraries have been used to assist my project, and a brief description will be provided of the most significant tools used.

Scikit-learn

It is also referred to as “sklearn”. This is a popular machine learning library that provides supervised and unsupervised algorithms and is useful for a project such as this as it offers metrics to evaluate the classifiers.

Pandas

Pandas stands for “Python Data Analysis” library. It takes data files e.g. CSV and creates an object with rows and columns called a Data Frame. It is useful for pre-processing and analysing data.

Numpy

It is a library for scientific computing that deals with powerful n-dimensional array objects, as well as several routines for processing those arrays, e.g. mathematical functions. [18].

Matplotlib

This is a plotting library for producing quality static, animated and interactive visualizations for given datasets. Forms include bar charts, scatter plots and histograms.

Basemap

Basemap is a toolkit for creating maps and plotting longitude and latitude data. It is an extension of matplotlib. Matplotlib plots the data, which is projected onto a matplotlib figure. [19].

Seaborn

Seaborn is a Python library built on the matplotlib library that is mainly used for data visualization. It has a simple high-level interface and useful statistical graphics for plotting the outcomes of the classifiers. [20].

Convertbng

This library converts decimal longitude and latitude coordinates to Easting and Northings (OSGB36) and vice versa. This conversion uses “Rust binary” and is particularly quick. [39].

CSV

CSV allows you to read, process and parse CSV files from text files.

Pickle

This is used to both serialize and deserialize objects. An object is “pickled” and converted into a byte stream saved on a disk. It can then be “unpickled” to inverse operation and return the object. [21].

2.8. Related Work

“Species distribution model transferability and model grain size – finer may not always be better” – Syed Amir Manzoor, Geoffrey Griffiths & Martin Lukac

This project uses species distribution modelling to predict the distribution of invasive species. They focussed on grain size’s effect on accuracy and model transferability using varying grain sizes of 50m, 300m and 1km in Wales UK. Their results showed that the models’ accuracy increased as grain size increased along with models transferability. In brief, the finer grain size variables led to less accurate models. [22].

“A Review of Machine Learning Based Species’ Distribution Modelling” – Jian Zhang & Sen Li

This paper looked at four different machine learning methods Random Forest, MaxEnt, Support Vector Machine(SVM) and Artificial Neural Networks (ANN) and their performance at predicting species distribution. They looked at their application, benefits and flaws in detail. It was deemed that the deep learning approach was the most promising with increasing environmental data. [23].

“Using Flickr for characterizing the environment: an exploratory analysis” – Shelan S. Jeawak, Christopher B. Jones & Steven Schockaert

Flickr data is a valuable informal information source for the disciplines of geography and ecology. They characterise places based on Flickr tags and compare it to structured data sources that had mixed results. Sometimes better and sometimes worse. To conclude, it was found that the Flickr data was complementary to traditional sources for characterising the environment. [24].

“Mapping Wildlife Species Distribution With Social Media: Augmenting Text Classification With Species Names” - Shelan S. Jeawak, Christopher B. Jones & Steven Schockaert

This project also uses social media data. This time to predict species distribution by utilising posts that explicitly mention the species, which led to high precision and low recall. They also tried to use a text classifier to get visual models of the location, that achieved a better performance. However, the optimal results came from combining the two strategies. [25].

Chapter 3: Approach

3.1. Project Requirements

As stated prior in this report in section 1.2, the aims of the project are to identify environmental features and use social media data to accurately predict the presence of a given species. To achieve this, six different machine learning solutions will be implemented with a comprehensive evaluation to compare the algorithms. In addition, three variations of grid cell sizes will be utilised to find the granularity that achieves the optimum performance. Another aim is to identify the most important environmental features that achieve the highest accuracy when predicting species occurrence. A set of functional and non-functional requirements were set prior to starting the

implementation of the project. These requirements were used to navigate the development of this project to achieve a fully functioning model.

3.1.1. Functional Requirements

The most important functional requirements can be categorised into two main sections as follows:

Data collection and pre-processing:

- Collected data is required to be processed to get value/count for each grid cell and saved to a new CSV file.
- The system can load in the relevant datasets required for training and testing the classifier.
- The data inputted into the classifier must be formatted in a certain way. This involves merging the loaded data to form one dataset, dropping unnecessary columns, deleting rows with missing data and splitting the data into training and testing.

Machine Learning:

- Develop Machine Learning Algorithms for Random Forest, Naïve Bayes, Support Vector Machine, Decision Trees, K Nearest Neighbor and Neural Networks.
- Train, test and evaluate each algorithm for eleven species and 3 different grid cell sizes.
- Visualise results for all algorithms to compare performances and find the most accurate for each grid cell size.
- Train and test a machine learning classifier using individual environmental features and combinations to evaluate their effectiveness.

3.1.2. Non-functional Requirements

- Reusability – The implementation can be reusable for other people in their systems.
- Reliability – The program will run with no failures or errors.
- Scalability – If the program needed to incorporate more grid cell sizes, species or environmental features it is easily achievable without affecting performance.

3.2. Project Structure

As you can see from the figure below, the project is split into two. The main reason for dividing the projects implementation is for simplicity as it allows for modifications to be made easier and to avoid over cluttering. Moreover, given the projects time frame this was considered the most suitable solution. Data collection and processing methods will be developed in the “ProcessData” folder. Further discussion on the collated data and their data flow together with functions used will be in section 3.3. Whereas the folder “Classifiers” has all 11 species with their notebooks containing the machine learning algorithms trained and tested using data from “ProcessData” along with their corresponding results. Discussion on this process can be found in section 3.4.



Figure 26: Project Split Diagram

3.3. “ProcessData” – Data Collection + Data Processing

3.3.1. Wildlife Selection and NBN Data

The aim of this project is to build a classifier which is able to determine the presence of a species based on selected environmental features. It is essential when collecting data for it to be directed at the relevant species. Eleven species were selected for training and testing the model. The species selected are chosen from the UK Biodiversity Action Plan (BAP) and represents a variety of different taxonomic groups. It’s imperative they differ in number of occurrences and distributions to give a broader overview of how the classifiers perform. This occurrence data was downloaded as a CSV file from the National Biodiversity Network and used as “ground truth”.

Species Name	Latin Name	Taxonomic Group	Wales	England	Scotland	NBN Count
Red Grouse	Lagopus Lagopus	Bird	Y	Y	Y	64,326

Tundra Swan	Cygnus Columbianus	Bird	Y	Y	Y	21,298
Willow Tit	Poecile Montanus	Bird	Y	Y	Y	93,353
Hawfinch	Coccothraustes Coccothraustes	Bird	Y	Y	Y	22,323
Yellow Wagtail	Motacilla Flava	Bird	Y	Y	Y	154,646
Twite	Linaria Flavivirostris	Bird	Y	Y	Y	46,512
Polecat	Mustela Putorius	Terrestrial Mammal	Y	Y	Y	8,523
Harvest Mouse	Micromys Minutus	Terrestrial Mammal	Y	Y	N	4,258
Adder	Vipera Berus	Reptile	Y	Y	Y	24,731
Grass Snake	Natrix Natrix	Reptile	Y	Y	Y	26,876
Great Crested Newt	Triturus Crstatus	Amphibian	Y	Y	Y	82,897

Data Flow

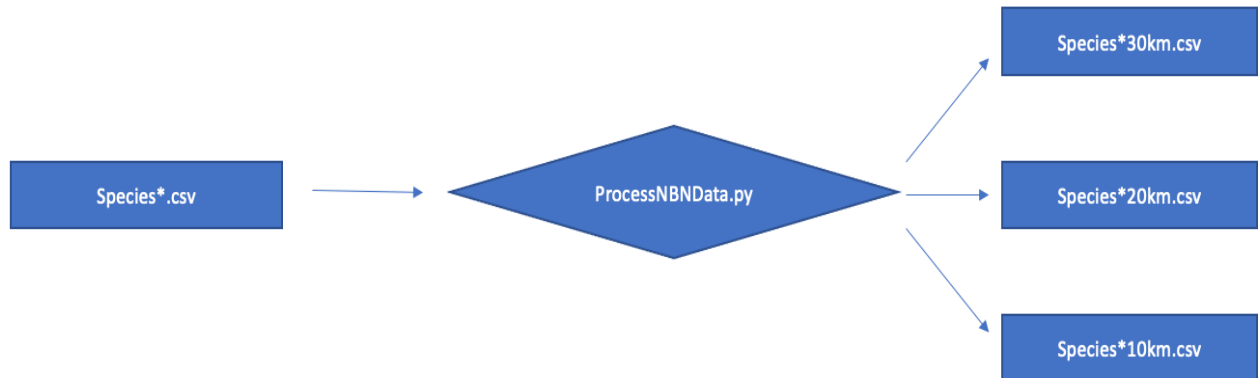


Figure 27: Species Data Flow

3.3.2. Flickr API

Once eleven species were chosen, all their possible data was collated. The Python library 'flickrapi' was used to stream photos that contain the tag of the species name and save them to a CSV file. 'Flickrapi' is a popular library that permits the Flickr API to be accessed using a valid authorisation key and secret. When collecting the data,

parameters can be set to narrow the search to specific photos. The filters used for this project include:

- **Tags** – Only photos containing the given word in their tags would be retrieved. Multiple tags can be set at the same time. In this case, the species' common name or their Latin name.
- **Bbox** – A list of 4 values that represent the boundary box, and only photos within this box will be searched. Hence, the bottom right corner and top left corner of the UK was used.

Species Name	Flickr Count
Red Grouse	227
Tundra Swan	91
Willow Tit	118
Hawfinch	117
Yellow Wagtail	166
Twite	64
Pole Cat	187
Harvest Mouse	192
Adder	471
Grass Snake	223
Great Crested Newt	41

Data Flow

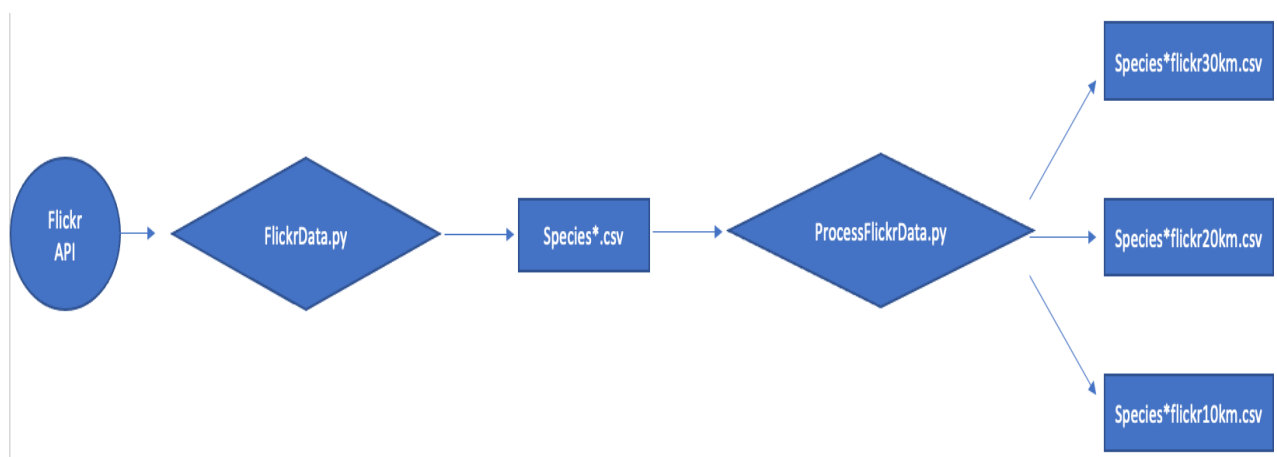



Figure 28: Species Flickr data flow

3.3.3. Environmental Features

These are the environmental features used for this project. Features were researched that may affect a species' presence, however, sourcing this data was reliant on the data being available to the public. These are the seven found in an Ascii Grid format that could easily be converted using excel to CSV.

Environmental Features	Description	Format
Mean Temperature	The dataset contains mean temperature for each month in a 5km resolution for the year 2016-2017.	Ascii Grid
<p style="text-align: center;">Data Flow</p>  <pre> graph LR subgraph Inputs direction TB J1[JanMeanTemp.asc] --> J2[JanMeanTemp.csv] J3[FebMeanTemp.asc] --> J4[FebMeanTemp.csv] J5[MarMeanTemp.asc] --> J6[MarMeanTemp.csv] J7[AprMeanTemp.asc] --> J8[AprMeanTemp.csv] J9[MayMeanTemp.asc] --> J10[MayMeanTemp.csv] J11[JunMeanTemp.asc] --> J12[JunMeanTemp.csv] J13[JulMeanTemp.asc] --> J14[JulMeanTemp.csv] J15[AugMeanTemp.asc] --> J16[AugMeanTemp.csv] J17[SepMeanTemp.asc] --> J18[SepMeanTemp.csv] J19[OctMeanTemp.asc] --> J20[OctMeanTemp.csv] J21[NovMeanTemp.asc] --> J22[NovMeanTemp.csv] J23[DecMeanTemp.asc] --> J24[DecMeanTemp.csv] end J2 --> P{ProcessTempData.py} J4 --> P J6 --> P J8 --> P J10 --> P J12 --> P J14 --> P J16 --> P J18 --> P J20 --> P J22 --> P J24 --> P P --> O1[TempMean30km.csv] P --> O2[TempMean20km.csv] P --> O3[TempMean10km.csv] </pre> <p style="text-align: center;"><i>Figure 29: Temperature data flow</i></p>		
Mean Wind Speed	The dataset contains mean wind speed for each month in a 5km resolution for the year 2016-2017.	Ascii Grid
<p style="text-align: center;">Data Flow</p>		

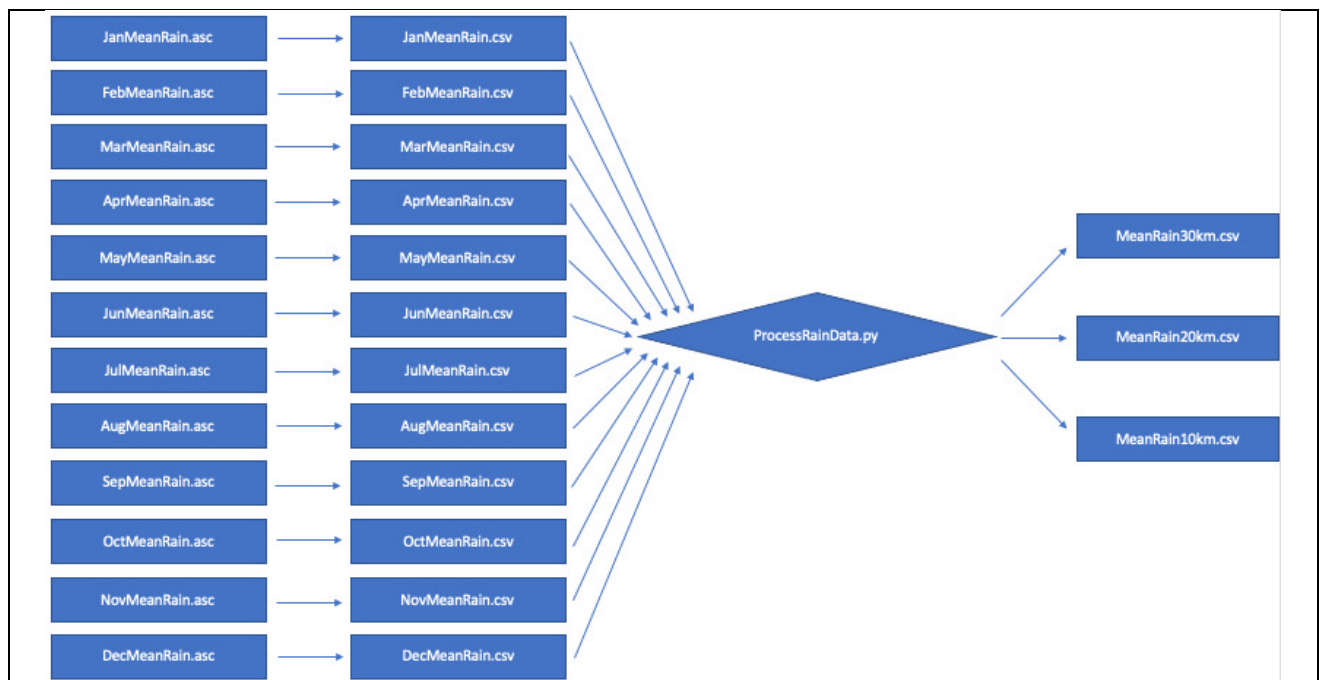


Figure 30: Rainfall data flow

Mean Rainfall	The dataset contains mean rainfall for each month in a 5km resolution for the year 2016-2017.	Ascii Grid
---------------	---	------------

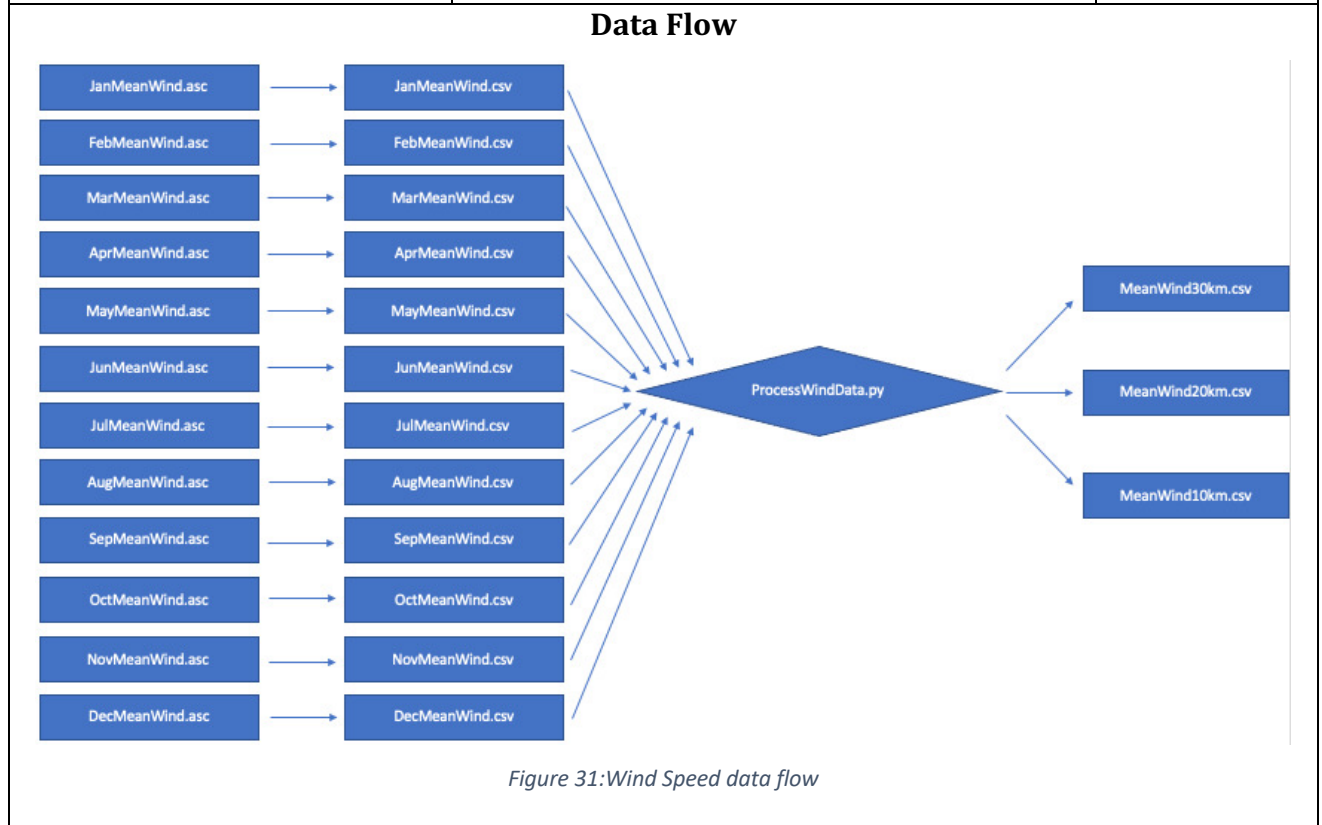


Figure 31: Wind Speed data flow

Land Cover Dominant Aggregate Class	<p>The dataset contains 10 simplified aggregate classification of land cover in a 1km resolution for the year 2010.</p> <p>"1-Broad-leaved / Mixed woodland 2-Coniferous Woodland 3-Arable and horticulture 4-Improved Grassland 5-Semi-natural grassland 6-Mountain, heath, bog 7-Built up areas and gardens 8-Standing open water 9-Coastal 10-Oceanic Sea"</p> <p>[31]</p>	Ascii Grid
<p style="text-align: center;">Data Flow</p> <pre> graph LR A[LandCoverAggregateClass.asc] --> B[LandCoverAggregateClass.csv] B --> C{ProcessLCDData.py} C --> D[LandCover30km.csv] C --> E[LandCover20km.csv] C --> F[LandCover10km.csv] </pre> <p style="text-align: center;"><i>Figure 32: Land Cover Dominant Class data flow</i></p>		
Land Cover Dominant Subclass	<p>The dataset contains 26 different classification of land cover in a 1km resolution for the year 2010.</p> <p>"1-Sea / Estuary 2-Water (inland) 3-Littoral rock 4-Littoral Sediment 5-Saltmarsh 6-Supra-littoral rock 7-Supra-littoral sediment 8-Bog (deep peat) 9-Dense dwarf shrub heath 10-Open dwarf shrub heath 11-Montane habitats 12-Broad-leaved / mixed woodland 13-Coniferous woodland 14-Improved Grassland 15-Neutral grassland 16-Setaside grassland 17-Bracken 18-Calcareous grassland 19-Acid grassland 20-Fen, marsh, swamp"</p>	Ascii Grid

	21-Arable cereals 22-Arable horticulture 23-Arable non-rotational 24-Suburban / rural developed 25-Continuous urban 26-Inland bare ground”	[31]
<p style="text-align: center;">Data Flow</p> <p style="text-align: center;"><i>Figure 33: Land Cover Subclass data flow</i></p>		
Emissions	The dataset contains the sum of pollutant-specific emissions data in 1km resolution for the year 2018	Ascii Grid
<p style="text-align: center;">Data Flow</p> <p style="text-align: center;"><i>Figure 34: Emissions data flow</i></p>		
Population	The dataset contains gridded human population based on Census 2011 in 1km resolution for the year 2011.	Ascii Grid
<p style="text-align: center;">Data Flow</p> <p style="text-align: center;"><i>Figure 35: Human population data flow</i></p>		

3.3.4. Process Data using Grid

As you can see from the data flow figures above, the environmental features and species data obtained needs to be processed in a specific way to acquire the information in 10km, 20km and 30km grid files. Therefore, for this project, it is necessary to split the UK into grid cells and calculate its respective values by processing the data to create new CSV files which will be inputted into machine learning algorithms to predict the presence of a species. The functions implemented to achieve this are as follows, and the boundary box for the UK was set using Easting and Northing:

`grid10km()`- Returns an array of x values (Easting) and y values (Northing), which if plotted form a grid with 10x10km cells. Also, the number of columns and rows used to fill the given boundary box.

`grid20km()`- Returns an array of x values (Easting) and y values (Northing), which if plotted form a grid with 20x20km cells. Also, the number of columns and rows used to fill the given boundary box.

`grid30km()`- Returns an array of x values (Easting) and y values (Northing), which if plotted form a grid with 30x30km cells. Also, the number of columns and rows used to fill the given boundary box.

`getCellByID()` – Using an integer for ID, this function will return an array of the x and y coordinates associated with that cell.

`getCellValue()`- Using the array of x and y coordinates, it will find the records that lie in that cell and return either a sum, mean or most frequent value from those records. (This is dependent on the feature as they have different specifications)

Using the figure below to illustrate my explanation, the algorithm for creating the grid starts by getting the boundary box which are 4 corner coordinates using Easting and Northing of the UK. Firstly, it calculates the coordinates of the bottom left point of the top left grid cell. Then using the given grid cell size (10, 20 or 30km), it loops through that row, adding the respective meters each time to the x coordinate and storing both x and y values every time. Until the right boundary is met and you move on to the next row, this continues until both the low y value and right x value is met.

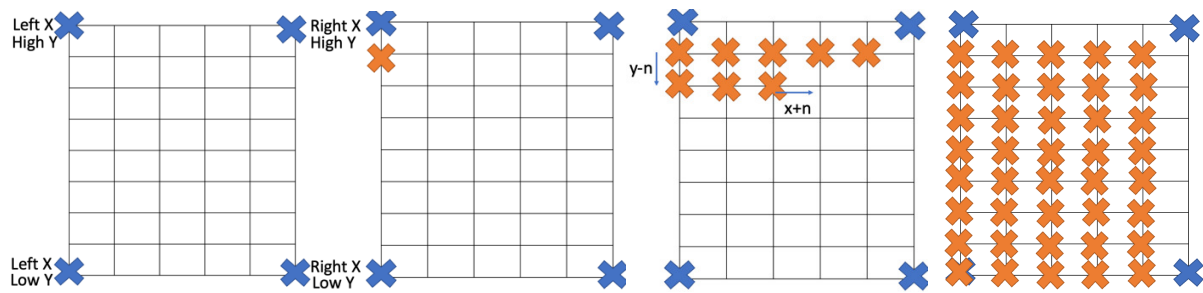


Figure 36: Grid supporting diagram

The `getCellByID()` function works by taking an integer which due to the way the array is stored for both x and y, the bottom left corner coordinates of the cell are simply the x and y values in the array at that number. So to get all four coordinates of that cell the grid size is added to both the x and y coordinates already found. `getCellValue()` then searches the given dataset to locate data that lies within the coordinates of the boundary found, and their findings are saved to a CSV file.

3.4. “Classifiers” – Machine Learning

3.4.1. “Classifiers” Folder Overview

The Classifiers folder contains a separate folder for each of the eleven species used. Inside said folders are notebooks for 10km, 20km, 30km and Environmental Features. These notebooks are where the machine learning processes discussed in section 3.4.2 takes place. Their results are saved as a pickle and the 'ClassifierResults' notebook collates them to produce a mean for the classifiers scores including all eleven species.

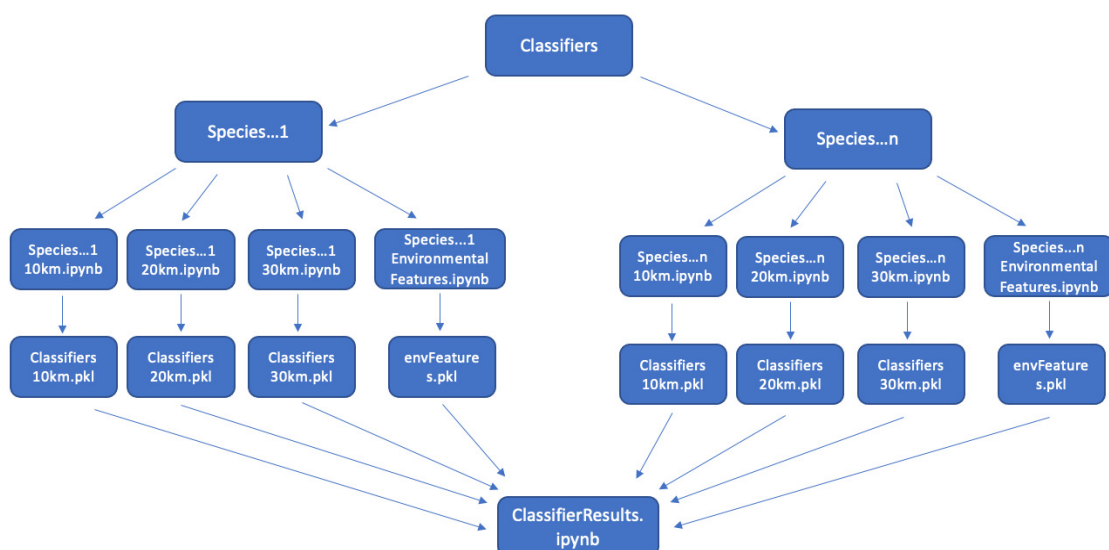


Figure 37: "Classifiers" folder overview

3.4.2. Machine Learning Process Overview

As you can see below, this flow chart will illustrate the behaviours used for each species in this folder for implementing a machine learning solution. The first stage involves reading in all the data required. For example, if species Red Grouse at a 20 km Grid Size is being implemented, the data needed for input is as follows RedGrouse20km, Population20km, LandCover20km, LandCoverSubclass20km, Emissions20km, TempMean20km, MeanRain20km, MeanWind20km and redgrouseflickr20km .These will all be combined to form one large dataset with any unnecessary columns dropped. Further processing of the data involves getting statistics to recognise if there are rows that need to be deleted due to no data values as this could affect the classifiers accuracy. Once a fully processed dataset has been formed it will be segregated into training and test sets. A crucial stage for a few classifiers used is scaling of the features, so the training and testing data will be scaled appropriately for those algorithms. Each of the classifiers selected will be trained using the training data and predict using the test data. In this instance, the classifiers using the non-scaled data are Naïve Bayes, Decision Trees, Random Forest, whereas Support Vector Machine, K Nearest Neighbor and Neural Network will use the scaled training and testing data. Their performances will be evaluated using various metrics such as accuracy, recall, precision and F1-Score along with a Confusion Matrix. To increase the effectiveness of the classifier's prediction abilities, parameter tuning will be performed until acceptable results are achieved. Once this process is completed for all species and grid cell sizes, results will be congregated and compared to find the classifier and grid cell size that outperformed the others.

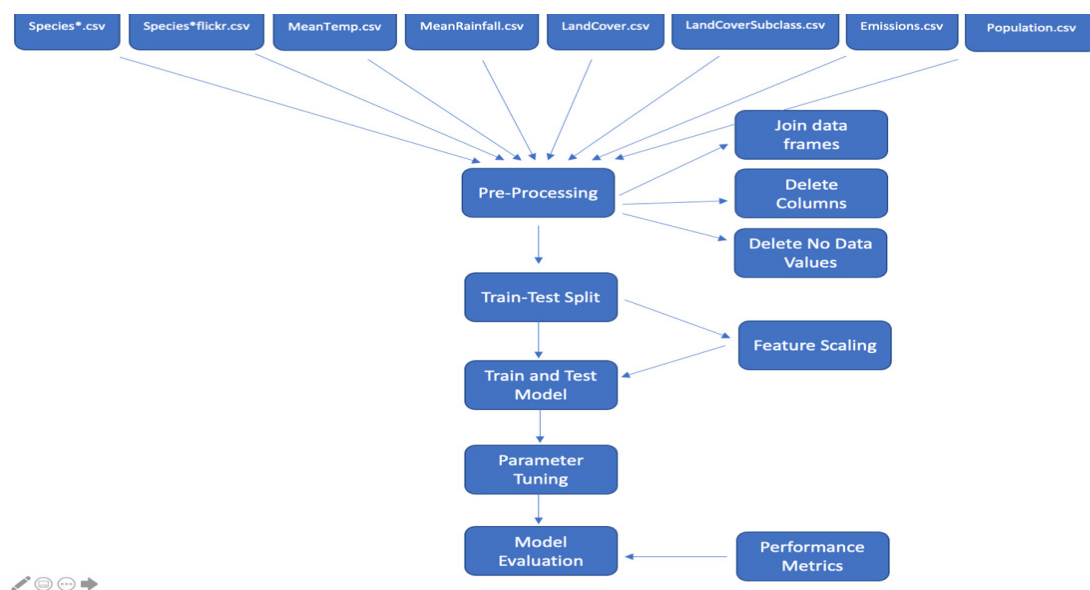


Figure 38: Machine Learning Solution

3.5. Development Strategy and Methodology

As mentioned previously in section 1.5, an agile methodology was used for this project. Fortnightly sprints were implemented to produce the optimal project results. The development process for this project was broken down into the following five iterations:

- Iteration One - Focussed on searching for appropriate datasets that could be used for environmental features that are freely available and in a useable format. Selecting a wide range of species that will be representative of wildlife and download ground truth data from NBN Atlas.
- Iteration Two – Create a grid function to split the UK into grid cell sizes. Pre-process the data to get the values associated with each grid cell and in a format that can be inputted into a classification model.
- Iteration Three – Focussed on developing the 6 selected classifiers. They will be trained on the given training data and tested on unseen data. All 6 classifiers implemented on all eleven species and for three grid sizes.
- Iteration Four – Parameters for the classifiers will need tuning to optimise performance and different evaluation techniques to calculate accuracy and compare results for each model.
- Iteration Five – Focussed on experimenting with environmental features using Random Forest classifier to test the effectiveness of the features individually and with multiple combinations.

Chapter 4: Implementation

4.1. Overview

The implementation of this project can be divided into three sections:

1. Data collection and processing.
2. Development of multiple machine learning classifier algorithms, evaluation techniques and their mean metrics for all species.
3. How different environmental features were used as classifier input.

4.2. Data Collection and Pre-processing

4.2.1. Flickr Data Collection

Below is a detailed description of the script developed to collect the necessary data from Flickr. The initial step involved setting up a Flickr Connection using the python package FlickrAPI and passing in a valid key and secret, permitting any number of methods to query the API.

```
#flickr connection
key = u'0794e7ff463ae21c212f3b0ee7123fc9'
secret = u'd512dcddceddfa43'
flickr = flickrapi.FlickrAPI(key, secret)
```

Figure 39: Flickr Connection

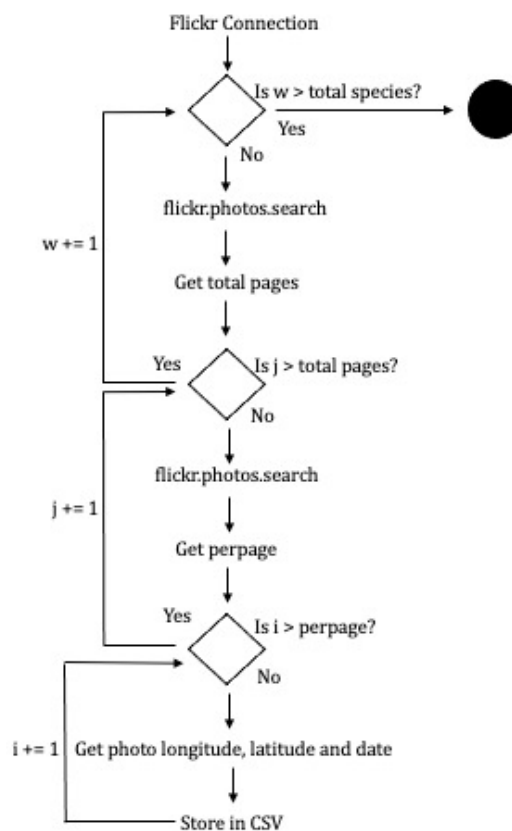


Figure 40: Flickr Script Diagram

As you can see from the figure above the script has 3 'For loops'. My initial loop iterates through the selected 11 species and opens a CSV file related to that species. Two calls are made to the API using flickr.photos.search() for each species. The first call is a generic search using parameters such as the English and Latin name of the individual species as tags and a boundary box of the UK using four coordinates. The Flickr API returns the results in a series of pages hence, the primary purpose of this call is to get the number of pages that will be searched. An XML python package is utilised to

navigate the outputted data from the calls. The second nested 'For loop' will go through the number of pages found by the initial call, this number corresponds to the second API call which is necessary to specify which page number to return. The final 'For loop' will get the number of photos per page which by default is 250, from here we can run through the photos on each page and extract the required data. In this case we need specific information such as date taken, longitude and latitude and append said information into the CSV file. However, exception handling is used if there is not 250 photos on said page.

```
#Loop through species
for w in range(0, len(englishnames)):

    with open (englishnames[w] + 'flickr.csv', 'a') as infile:
        writer = csv.writer(infile)

        header = "dateTime", "latitude", "longitude"
        writer.writerow(header)

        photos = flickr.photos.search(tags = englishnames[w] + ',' + latinnames[w], bbox="-5.9025857, 49.924471, 2.5125204, 58.898406")

        totalPages = int(photos[0].get('pages')) + 1
        print (int(photos[0].get('total')))

        #Loop through the pages of xml
        for j in range(1, totalPages):

            #Open that page of xml
            photos = flickr.photos.search(tags = englishnames[w] + ',' + latinnames[w], bbox="-5.9025857, 49.924471, 2.5125204, 58.898406", page=j)

            #Loop through the page of xml
            for i in range(0, int(photos[0].get('perpage'))):
                #print (int(photos[0].get('total')))

                #Need exception handling as quite often a page will not have 250 results

                try:
                    #Collect necessary data from Flickr API
                    photo_id = photos[0][i].get('id')

                    photoInfo = flickr.photos.getInfo(photo_id = photos[0][i].get('id') )
                    datetime = "" + photoInfo[0][4].get('taken') + ""
                    longitude = "" + photoInfo[0][12].get('longitude') + ""
                    latitude = "" + photoInfo[0][12].get('latitude') + ""

                    #Insert the data into a csv table

                    data = datetime, latitude, longitude
                    writer.writerow(data)

                #If value is out of bounds then break the loop and move to the next page
                except IndexError:
                    break
```

Figure 41: Flickr Data Collection

4.2.2. Process NBN Data

To prepare the data for the machine learning aspect of this project, the initial CSV file from NBN Atlas needed to be outputted into three different CSV files for each grid cell size. The first step to achieving this for each of the 11 species' was to read in their observation data obtained from NBN Atlas into the script below and for each row (observation) to obtain the longitude and latitude values, converting them to Easting and Northing values, adding these x and y coordinates to a new CSV file. The output of this 'For loop' is 11 new CSV files with the structure Species*EN.csv data is ['Easting', 'Northing'].

```

for i in range(0, len(species)):
    with open (species[i] + 'EN.csv', 'w') as infile:
        with open (species[i] + '.csv', 'r') as file:
            reader = csv.reader(file)
            writer = csv.writer(infile)
            writer.writerow(['Easting', 'Northing'])
            next(reader) #skip first line
            for row in reader:
                lat = float(row[21])
                lon = float(row[22])
                xy = geofunc.ll2en(lat, lon)
                eastingNorthing = str(xy[0])[1:-1], str(xy[1])[1:-1]
                writer.writerow(eastingNorthing)

```

Figure 42: Create file of Easting and Northing

My conversion of longitude and latitude coordinates to Easting and Northing (OSGB36) is a function called ll2en() which uses the convert_bng library.

```

def ll2en(self, lat, lon):
    xy = convert_bng(lon, lat)
    return(xy)

```

Figure 43: Convert Long and Lat to Easting and Northing

An extremely valuable function in my project was CreateGrid(). It allowed for the splitting of the UK into grid cells and acquire their respective information needed from each individual cell. All the environmental features and ground truth data were run using this function.

The algorithm iterates through the rows of the grid starting at the top left, whilst a nested loop iterates through the columns. The x and y coordinates of the bottom left corner of the cells are stored in separate arrays. Using this implementation process made it easier to call grid cells by their cell id.

```

def grid20km(self, leftX, rightX, lowY, highY):
    grid_size = 20000
    n_cols = (rightX - leftX)/grid_size
    n_rows = (highY - lowY)/grid_size
    xx = []
    yy = []
    highY = highY - grid_size
    rightX = rightX - (grid_size*2)
    plotY = highY

    while plotY >= lowY:
        xx.append(leftX)
        yy.append(plotY)

        plotX = leftX

        while plotX <= rightX:
            plotX = plotX + grid_size
            xx.append(plotX)
            yy.append(plotY)

        plotY = plotY - grid_size

    rightX = plotX

    return(xx, yy, grid_size, n_cols, n_rows)

```

Figure 44: Create 20km Grid

The `getCellByID()` function was called and used in conjunction with the grid arrays to obtain all 4 corners of the cell. This is achieved by adding the grid cell size to the x and y coordinates.

```
def getCellByID(self, xx, yy, squareID, grid_size):
    getCellx = []
    getCellx.append(xx[squareID])
    getCellx.append(xx[squareID] + grid_size)

    getCelly = []
    getCelly.append(yy[squareID])
    getCelly.append(yy[squareID] + grid_size)

    return (getCellx, getCelly)
```

Figure 45: Call cell by ID

Once all four coordinates for that cell are retrieved the next step involves searching a given CSV file to gather all the available data within those points. In this case, the earlier CSV files created containing the Easting and Northing coordinates are utilised, and the points located within that grid cell boundary are found and used. This function differs slightly between environmental features. In this instance, we calculated how many points were retrieved within the coordinates.

```
def getCellCount(self, getCellx, getCelly, species, grid_size):

    csvSource = open(species + 'EN.csv', 'r')
    reader = csv.reader(csvSource)

    count = 0
    iterreader = iter(reader)

    next(iterreader)
    for row in iterreader:
        if float(row[0]) >= float(getCellx[0]) and float(row[0]) < float(getCellx[1]) and float(row[1]) >= float(getCelly[0]) and float(row[1]) < float(getCelly[1]):
            count = count + 1
    return(count)
```

Figure 46: Get Cell Count

For each species a new CSV file was created called `Species*GridCellSize*km.csv` with the structure ['Cell ID', 'X', 'Y', 'Count', 'Presence']. The nested 'For loop' is in the range of 0 to the number of columns multiplied by the number of rows. This number relates to a cell in which we need its coordinates, calling the function mentioned prior `getCellByID()`. When all four coordinates of the cell are found, we call `getCellCount()`. If there is a count higher than 0, we use a '1' in the presence column otherwise, a '0'. These are the labels that will be used later for the classifier. This process was reiterated for 30km, 20km and 10km.

```

xx, yy, grid_size, n_cols, n_rows = grid.grid20km(grid.leftX, grid.rightX, grid.lowY, grid.highY)

for j in range(0, len(species)):
    spec = species[j]

    with open (species[j] + '20km.csv', 'w') as infile:
        writer = csv.writer(infile)
        writer.writerow(['Cell ID', 'X', 'Y', 'Count', 'Presence'])

        for k in range(0, (27*51)):

            getCellX, getCellY = grid.getCellByID(xx, yy, k, grid_size)
            count = countin.getCellCount(getCellX, getCellY, spec, grid_size)

            if float(count) > 0:
                presence = 1
                data = k, getCellX[0], getCellY[0], count, presence
                writer.writerow(data)

            else:
                presence = 0
                data = k, getCellX[0], getCellY[0], count, presence
                writer.writerow(data)

```

Figure 47: Create new grid CSV file

The ground truth grid data was later plotted. Using the longitude and latitude coordinates of the UK as parameters, Basemap() created an outline of the UK map using the coastlines. For each row where the species is 'present' the Easting and Northings coordinates added half the size in meters of the granularity being used to indicate the middle of the cell. These coordinates are converted to longitude and latitude before being plotted. The map can be compared to the one used in NBN Atlas to verify results and visualise distribution.

```

#Plot ground truth data
fig, ax = plt.subplots(1, figsize=(10,10))
m = Basemap(llcrnrlon=-7.5600,llcrnrlat=49.7600, urcrnrlon=2.7800,urcrnrlat=60.840, resolution='f')
m.drawcoastlines()
for i, row in df.iterrows():
    if row['Presence'] == 1:
        easting = float(row['X']) + 15000
        northing = float(row['Y']) + 15000
        lonlat = convert_lonlat(easting, northing)
        lat, lon = str(lonlat[1])[1:-1], str(lonlat[0])[1:-1]
        x, y = float(lon), float(lat)
        xl, yl = m(x, y)
        m.scatter(xl, yl, marker = '.', c = 'red', zorder=5)
plt.show()

```

Figure 48: Plot ground truth data

4.2.3. Process Environmental Data

All of the environmental data gathered was processed similarly to the method just mentioned with minor changes. Environmental data was originally downloaded in an ASCII grid format, for convenience excel was used to export the data as a CSV file. The script below was applied to create a new CSV file with the data in this structure ['X', 'Y',

'LandCoverSub']. In an ASCII grid file '-9999' was commonly used to signify no data recorded, therefore these cells were skipped. The maths behind this algorithm differs between environmental features as it is dependent on the first six lines of the ASCII file, these indicate the number of rows, columns, grid cell size and no value number etc,

```
with open ('LandCoverSubclass1km.csv', 'w') as infile:
    with open ('LandCoverSubclassData.csv', 'r') as file:
        reader = csv.reader(file)
        writer = csv.writer(infile)
        writer.writerow(['X', 'Y', 'LandCoverSub'])
        y = 1300000 - 1000
        for i in range(6):
            next(reader)
        for row in reader:
            x = 0
            for value in row:
                if value != '-9999':
                    data = x, y, value
                    writer.writerow(data)
                    x = x + 1000
            y = y - 1000
```

Figure 49: Convert ASCII Format

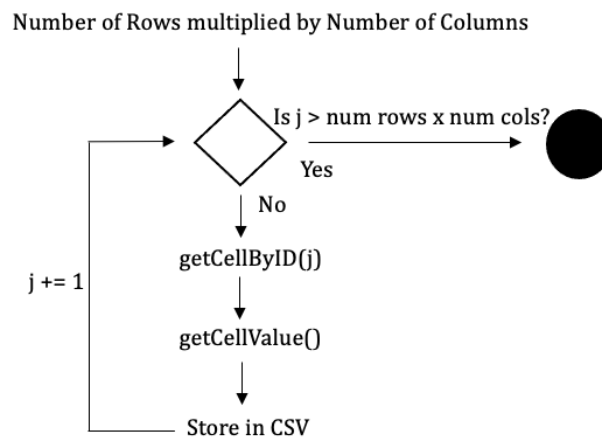


Figure 50: Add values to CSV diagram

The same method described previously was used to create a grid and acquire coordinates for the cell by ID (see figure above). However, in this case, we wanted the most common Land Cover classification in each grid cell. This function differs between features as in some cases, the mean of a grid cell is needed e.g. mean temp or the total e.g. human population. If no data was found for a given cell, '-9999' would be inputted as a default value as when joining the files all the rows would need to coincide.

```
def most_frequent(List):
    return max(set(List), key = List.count)
```

Figure 51: Most Frequent

```

def getCellValue(self, getCellX, getCellY):
    csvSource = open('LandCover1km.csv', 'r')
    reader = csv.reader(csvSource)

    count = 0
    iterreader = iter(reader)
    tempValue = []

    next(iterreader)
    for row in iterreader:
        if float(row[0]) >= float(getCellX[0]) and float(row[0]) <= float(getCellX[1]) and float(row[1]) >= float(getCellY[0]) and float(row[1]) <= float(getCellY[1]):
            tempValue.append(float(row[2]))
            count = count + 1

    if count > 0:
        value = most_frequent(tempValue)
    else:
        value = -9999

    return(value)

```

Figure 52: Get Cell Value

Completion of processing the collated data in this way resulted in the following files:

- MeanWindSpeed*GridSize.csv – ['Cell ID', 'X', 'Y', 'MeanWind']
 - 'MeanWind' refers to the average windspeed found for 12 months for each cell.
- TempMean*GridSize.csv – ['Cell ID', 'X', 'Y', 'MeanTemp']
 - 'MeanTemp' relates to the average temperature found for 12 months for each cell.
- MeanRain*GridSize.csv – ['Cell ID', 'X', 'Y', 'MeanRain']
 - 'MeanRain' defines the average rainfall found for 12 months for each cell.
- Population*GridSize.csv – ['Cell ID', 'X', 'Y', 'Population']
 - 'Population' is the sum of the human population found for each cell.
- LandCover*GridSize.csv – ['Cell ID', 'X', 'Y', 'Land Cover']
 - 'Land Cover' indicates the most frequent aggregate land cover class for each cell.
- LandCoverSubclass*GridSize.csv – ['Cell ID', 'X', 'Y', 'LandCoverSub']
 - 'LandCoverSub' refers to the most frequent land cover subclass found for each cell.
- Emissions*GridSize.csv – ['Cell ID', 'X', 'Y', 'Emissions']
 - 'Emissions' is the sum of the emissions found for each cell.

- *SpeciesFlicker*GridSize.csv – ['Cell ID', 'X', 'Y', 'FlickerCount', 'FlickerPresence']
 - 'FlickerCount' indicates the number of photos found for each cell.
 - 'FlickerPresence' relates to whether there is at least one photo found for each cell.
- *Species*GridSize.csv - ['Cell ID', 'X', 'Y', 'Count', 'Presence']
 - 'Count' represent the number of occurrences found for each cell.
 - 'Presence' defines whether there is at least one occurrence found for each cell.

4.3. Machine Learning Classification Models

For this project, five different machine learning classifier models and a deep learning model was implemented. Each classifier will be discussed individually about why they were selected, the pros and cons of the model and their development. The Scikit-learn library was used to assist the development of all. Jupyter notebook was used for this part of the project, allowing for easy analyses of the process in a step by step manner. However, this decision was made when 3-5 species were going to be used, thus writing a script to pass all species may have reduced development time. The implementation process for the 11 species and 3 grid cell sizes follow a similar pattern.

The first step involved importing all the CSV file data, which was processed as mentioned prior in section 4.2. into the notebook using pandas read_csv(), providing the path to retrieve the relevant data. This then created a data frame with an assigned variable name.

```
#import data
redGrouse = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessNBNDData/RedGrouse20km.csv')
population = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessPopulationData/Population20km.csv')
landCover = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessLCData/LandCover20km.csv')
landCoverSub = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessLCData/LandCoverSubclass20km.csv')
emissions = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessEmissionsData/Emissions20km.csv')
temp = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessTempData/TempMean20km.csv')
rain = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessRainfallData/MeanRain20km.csv')
wind = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/ProcessWindData/MeanWind20km.csv')
flicker = pd.read_csv('/Users/laura/Desktop/newProjectML/ProcessData/FlickerData/redgrouseflicker20km.csv')
```

Figure 53: Loading Datasets

After reading in the data, it's imperative to get all this information into one data frame, therefore concatenated the tables that had been imported using pandas.concat(). Once,

joined they form one large data frame, the columns deemed unnecessary for input into the classifier were x and y coordinates, plus its corresponding ID, thus dropped.

```
#Join tables and Drop columns
data = pd.concat([redGrouse, population, landCover, landCoverSub, emissions, temp, rain, wind, flickr], axis = 1)
data = data.drop(['CellID', 'Cell ID', 'X', 'Y', 'Count', 'ID', 'FlickrPresence'], axis = 1)
data
```

Figure 54: Join data frames and drop columns

Earlier mentioned, when there was no data found for a specific cell, -9999 was inputted as a default value. If there is a no data value in a row, all the data corresponding to that specific cell will need to be removed as if inserted into the classifier it could affect the results.

```
#drop rows with no value == -9999
data = data[data.Population != -9999]
data = data[data.Emissions != -9999]
data = data[data.MeanTemp != -9999]
data
```

Figure 55: Drop rows with a -9999 value

A useful tool to check the validity of the data and that all the no data values are removed and to test for anomalies is to use `.describe()`. For example, if the minimum value for a column was '-9999' then it would be apparent all necessary rows haven't been removed.

```
data.describe()
```

Figure 56: Describe data frame

For the machine learning algorithms it's necessary to separate the features from the labels. In this instance, the 'Presence' column holds the labels of a 0 or 1 thus, indicating the presence or absence to be used as a target to classify the features against. X refers to features and Y the labels.

```
x = data.drop('Presence', axis=1)
y = data['Presence']
```

Figure 57: Split features and labels

A requirement of supervised learning entails evaluating the classifiers predictive performance on unseen data, therefore, reducing bias. A method that is commonly used is the Scikit-learn library which provides `train_test_split()`. This function will split the features and labels into training and testing subsets. The size of the training and testing data sets are based on the test size argument passed into the function. In this case, the training size equals 75% and 25% for testing. This is relatively standard for machine

learning projects as it avoids overfitting or underfitting the model. A multitude of splits was tested on the data from 10:90 train-test to 90:10 and 75:25 had the more favourable outcome.

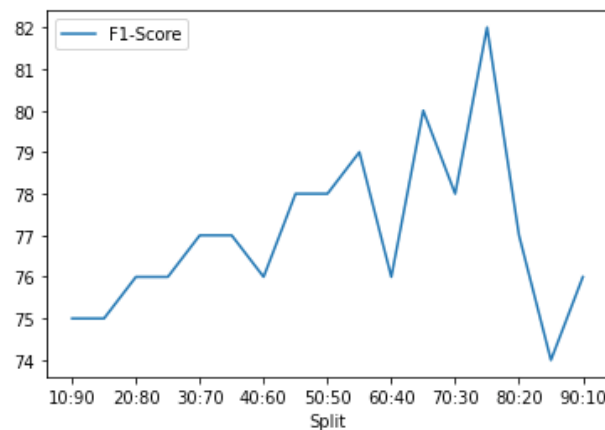


Figure 58: Experiment train and test split

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25)
```

Figure 59: train-test split

A requirement of this project is to compare the results of the classifiers, and an easy way to group the results is to initialise a data frame and add the respective information as you progress.

```
classifiers = pd.DataFrame (columns = ['Classifier', 'Accuracy', 'Precision', 'Recall', 'F1-Score'])
classifiers
```

Figure 60: Initialise data frame

Feature scaling through standardisation is an important step of the pre-processing process for many distance-based machine learning algorithms as the distance between data points is used to determine similarity. [25] Scikit-learn offered several standardisation functions and for this project StandardScaler() was chosen.

```
scaler = StandardScaler()
scaler.fit(x_train)
X_train = scaler.transform(x_train)
X_test = scaler.transform(x_test)
```

Figure 61: Feature Scaling

Naïve Bayes was the first supervised learning classifier chosen to classify the species occurrence. It is quick and efficient compared to other more complicated algorithms due to its assumption that features are independent. However, this speed comes at a cost as

less accuracy is achieved as this assumption is not usually the case. [41]. To implement the model, GaussianNB() is used along with its default parameters.

```
#Naive Bayes
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

Figure 62: Naive Bayes Classifier train and predict

Confusion_matrix() is a helpful function using the Scikit-learn library to get the false positives, false negatives, true positive and true negatives. Also, using sns.heatmap() to perceive the results in a visually pleasing manner for the classifiers evaluation.

```
#Create the confusion matrix
LABELS = ['0', '1']
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(4,4))
sns.heatmap(conf_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d", cmap="Blues");
plt.title("Confusion matrix- Naive Bayes")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```

Figure 63: Confusion Matrix implementation

Four performance metrics (1)Accuracy, (2) Precision, (3) Recall, (4) F1-Score were used. All evaluation metrics uses Scikit Learn library “metrics” to give different evaluations of performance.

- sklearn.metrics.accuracy_score() – This function calculates subset accuracy. [35].
- sklearn.metrics.precision_score() – This function calculates the classifiers ability not to label as ‘present’, when it’s actually ‘absent’. [36].
- sklearn.metrics.recall_score() - This function calculates the classifiers ability to find all positive labels. [37].
- sklearn.metrics.f1_score() - This function calculates the average of precision and recall. [38].

For each classifier these results are stored in the data frame “classifiers” created before.

```
#Add to dataframe
classifier = "Naive Bayes"
accuracy = round(accuracy_score(y_pred, y_test),2)*100
precision = round(precision_score(y_pred, y_test, average= 'binary', pos_label=1),2)*100
recall = round(recall_score(y_pred, y_test, average= 'binary', pos_label=1),2)*100
f1score = round(f1_score(y_pred, y_test, average= 'binary', pos_label=1),2)*100
classifiers = classifiers.append({'Classifier': classifier, 'Accuracy': accuracy, 'Precision': precision, 'Recall': recall, 'F1-Score': f1score})
classifiers
```

Figure 64: Evaluation metrics

A second classifier chosen to implement is Support Vector Machines. The advantage of this algorithm is that it generalizes well, leading to a lower risk of overfitting. On the other hand, it doesn't perform well when given a larger dataset as the training time increases. [42]. This algorithm required feature scaling therefore, 'X_train' and 'X_test' data are used instead of 'x_train' and 'x_test'. This algorithm was developed using Scikit-Learn's SVC() with the parameters kernel equals 'linear'. As you can see below from experimentation with other possible kernels, linear had the best performance for F1-Score.

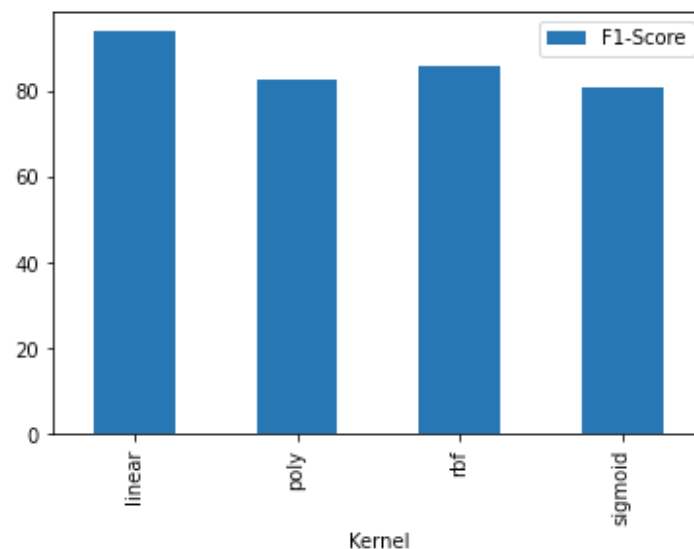


Figure 65: Experiment with SVM Kernels

```
#SVM
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train.values.ravel())
y_pred = svclassifier.predict(X_test)
```

Figure 66: Support Vector Machine Classifier train and predict

Another classifier is Decision Trees. A positive reason for using this algorithm is that feature selection occurs intuitively therefore, any features deemed unimportant will have no influence over the overall results. It does however, tend to risk overfitting. [43]. The function used to implement this classifier is DecisionTreeClassifier().

```
#Decision Tree
clf = DecisionTreeClassifier()
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

Figure 67: Decision Tree Classifier train and predict

Nearest Neighbour was chosen to be implemented in this project. Pros of the algorithm are that it doesn't assume anything about the data. Cons are it is time-consuming to find the optimal K value. [44]. As you can see below, from experimenting with the K value between 1 to 10, 7 came out on top, hence used as a parameter. KNN also requires feature scaling to avoid making wrong predictions. Scikit-Learn's `KNeighborsClassifier()` function was used.

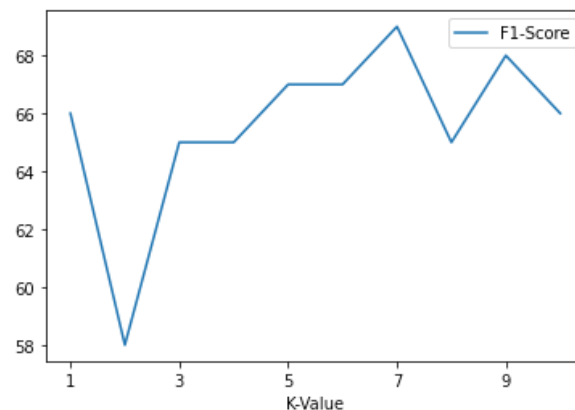


Figure 68: Experimenting with K value

```
#Nearest Neighbor
knn = KNeighborsClassifier(n_neighbors=7)
knn = knn.fit(X_train, y_train.values.ravel())
y_pred = knn.predict(X_test)
```

Figure 69: K-Nearest Neighbor Classifier train and predict

The penultimate classifier chosen for development is Random Forest. The reasons for this selection is its reduced likelihood to overfit compared to other algorithms such as Decision Trees. However, due to the algorithm being more complex it requires more computational resources and time. [45]. To implement the model `RandomForestClassifier()` is used along with its default parameters.

```
#Random Forest
rf_base_model = RandomForestClassifier()
rf_base_model = rf_base_model.fit(x_train, y_train)
y_pred = rf_base_model.predict(x_test)
```

Figure 70: Random Forest Classifier train and predict

The final algorithm which can be considered as deep learning is Neural Networks. The Neural Network developed here is called Multi-Layer Perception and is implemented using `MLPClassifier()`. Training of the model can be time-consuming, but once completed, the prediction process is fast. Scaling of the features had a significant impact

on performance when tested. Parameters were trialled and these achieved the best outcome.

```
#Neural Network Classifier
#https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp = mlp.fit(X_train, y_train.values.ravel())
y_pred = mlp.predict(X_test)
```

Figure 71: Neural Network Classifier train and test

Each classifiers results were appended to the 'classifiers' data frame which were saved to a pickle for future use when comparing results.

```
classifiers.to_pickle("classifiers20km.pkl")
```

Figure 72: Save data frame to pickle

On completion of this process for each species, thus having a data frame for the classifiers results, eleven species were saved as a pickle. They are all depickled using `read_pickle()` and passed the correct path to the directory.

```
#import pickle for 10km
adder = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/Adder/classifiers10km.pkl')
grassSnake = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/GrassSnake/classifiers10km.pkl')
greatCrestedNewt = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/GreatCrestedNewt/classifiers10km.pkl')
harvestMouse = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/HarvestMouse/classifiers10km.pkl')
hawfinch = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/Hawfinch/classifiers10km.pkl')
tundraSwan = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/TundraSwan/classifiers10km.pkl')
poleCat = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/PoleCat/classifiers10km.pkl')
redGrouse = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/RedGrouse/classifiers10km.pkl')
twite = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/Twite/classifiers10km.pkl')
willowTit = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/WillowTit/classifiers10km.pkl')
yellowWagtail = pd.read_pickle('/Users/laura/Desktop/newProjectML/Classifiers/YellowWagtail/classifiers10km.pkl')
```

Figure 73: Read in pickle for each species

The data frames containing the results for each species are combined using `pd.concat()`.

```
#Join
data = pd.concat([adder, grassSnake, greatCrestedNewt, harvestMouse, hawfinch, poleCat, redGrouse, twite, willowTit, ye
```

Figure 74: Join data frames

The mean is achieved by iterating through the data frame and every row adding the results for each metric, then dividing by 11. Result of this cell is a data frame containing the mean score for each metric using all 11 species selected. Allowing for the requirement to be met by comparing the results of the classifiers to find the most accurate.

```

#mean 20km species
df20 = pd.DataFrame(columns=['Classifier', 'Accuracy', 'Precision', 'Recall', 'F1-Score'])
for index, row in data.iterrows():
    accuracy = row[1] + row[6] + row[11] + row[16] + row[21] + row[26] + row[31] + row[36] + row[41] + row[46] + row[51]
    accuracy = accuracy/12
    precision = row[2] + row[7] + row[12] + row[17] + row[22] + row[27] + row[32] + row[37] + row[42] + row[47] + row[52]
    precision = precision/12
    recall = row[3] + row[8] + row[13] + row[18] + row[23] + row[28] + row[33] + row[38] + row[43] + row[48] + row[53]
    recall = recall/12
    flscore = row[4] + row[9] + row[14] + row[19] + row[24] + row[29] + row[34] + row[39] + row[44] + row[49] + row[54]
    flscore = flscore/12
    df20.loc[index] = row[0], accuracy, precision, recall, flscore
df20

```

Figure 75: Mean for classifier

4.3. Environmental Feature Importance

A sub experiment of this project was to evaluate the impact of the environmental features used and to find combinations that support one another. This experiment was conducted using 20km grid cell data. To achieve this, the data was processed the same way as previously explained, however, when setting the labels and features only specifically selected columns were used. When splitting the data for each feature into train and test sets a random state was applied to reduce bias.

```

#Using just mean temp data
temp = data.drop(['Land Cover', 'LandCoverSub', 'Population', 'Emissions', 'MeanWind', 'MeanRain'], axis = 1)
X = temp.drop('Presence', axis=1)
Y = temp['Presence']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 42)
temp

```

Figure 76: Individual Environmental Feature

The Random Forest classifier was implemented to test the effectiveness of the data at predicting a species occurrence and the results were saved to a data frame. This behaviour was carried out on all environmental features individually and also features that coincide. A mean using all eleven species was found using a similar method to that previously mentioned.

Chapter 5: Testing

This section concentrates on the Test Cases that were used to test the project developed met the main functional requirements mentioned in section 3.1.

Test Case ID: TC1	Process data to get values for each grid cell		
Precondition: Collected data for each feature in CSV format			
Test Case Steps: 3			
Step No	Procedure	Response	Pass/Fail

1	Read in CSV file using file name	No error message	Pass
2	Get value associated with each grid cell	No error message	Pass
3	Save to CSV file in directory	No error message	Pass
Test Case Outcomes: Processed 10km, 20km and 30km CSV files for each feature and species.			
Related Test: None			

Test Case ID: TC2	Pre-process data for classifiers		
Precondition: CSV files containing feature and species values for 10km, 20km and 30km			
Test Case Steps: 6			
Step No	Procedure	Response	Pass/Fail
1	Run cell to read in CSV files using directory and save it to data frame	No error message	Pass
2	Run cell to join together for one large dataset	No error message	Pass
3	Run cell to remove rows of cells with at least one no data value	No error message	Pass
4	Run cell to split dataset into features and labels	No error message	Pass
5	Run cell to split dataset into train and test	No error message	Pass
6	Run cell to scale features	No error message	Pass
Test Case Outcomes: Pre-processed dataset ready to be inputted to classifiers			
Related Test: TC1			

Test Case ID: TC3	Naïve Bayes
Precondition: Must have training and testing data to use as input	
Test Case Steps: 2	

Step No	Procedure	Response	Pass/Fail
1	Run cell to train Naïve Bayes and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC4	Support Vector Machines		
Precondition: Must have training and testing data to use as input			
Test Case Steps: 2			
Step No	Procedure	Response	Pass/Fail
1	Run cell to train Support Vector Machines and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC5	Decision Trees		
Precondition: Must have training and testing data to use as input			
Test Case Steps: 2			
Step No	Procedure	Response	Pass/Fail
1	Run cell to train Decision Tree and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC6	K Nearest Neighbor
Precondition: Must have training and testing data to use as input	

Test Case Steps:			
Step No	Procedure	Response	Pass/Fail
1	Run cell to train K Nearest Neighbor and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC7	Random Forest		
Precondition: Must have training and testing data to use as input			
Test Case Steps: 2			
Step No	Procedure	Response	Pass/Fail
1	Run cell to train Random Forest and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC8	Neural Networks		
Precondition: Must have training and testing data to use as input			
Test Case Steps: 2			
Step No	Procedure	Response	Pass/Fail
1	Run cell to train Neural Network and predict labels	No error messages	Pass
2	Run cell to get evaluation scores and save them to data frame	No error messages	Pass
Test Case Outcomes: Classifiers results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Test Case ID: TC9	Calculate and visualise results
--------------------------	--

Precondition: All machine learning algorithms were run successfully for every species and grid cell size along with evaluation metric results saved			
Test Case Steps: 7			
Step No	Procedure	Response	Pass/Fail
1	Run cell to read in data frame for 11 species	No error messages	Pass
2	Run cell to merge data frames	No error messages	Pass
3	Run cell to calculate mean score for each classifier and metric	No error messages	Pass
4	Run cell to create graph of accuracy for all classifiers	No error messages	Pass
5	Run cell to create graph of precision for all classifiers	No error messages	Pass
6	Run cell to create graph of recall for all classifiers	No error messages	Pass
7	Run cell to create graph of f1-score for all classifiers	No error messages	Pass
Test Case Outcomes: Tables and graphs comparing results to use in report			
Related Test: TC1, TC2, TC3, TC4, TC5, TC6, TC7 and TC8			

Test Case ID: TC10	Environmental Feature importance		
Precondition: Must have pre-processed data			
Test Case Steps: 5			
Step No	Procedure	Response	Pass/Fail
1	Run cells to split data frame using specific columns and train-test Random Forest Classifier	No error messages	Pass
2	Run cells to get evaluation scores and save them to data frame	No error messages	Pass

3	Run cell to read in data frame for 11 species	No error messages	Pass
4	Run cell to merge data frames	No error messages	Pass
5	Run cell to calculate mean metric score for each environmental features used	No error messages	Pass
Test Case Outcomes: Environmental Features results for accuracy, precision, recall and F1-Score			
Related Test: TC1, TC2			

Chapter 5: Results and Evaluation

5.1. Classification Models Performance

Successful implementation of the machine learning classifiers allowed for a critical evaluation of the models' performance. The goal for the algorithms was to successfully predict the presence or absence of a given species. Thus, the evaluation metrics results are compared between the classifiers. Using the main classification metrics for evaluating the results of a classifier. Therefore the comparison tables consist of the classifiers name, Accuracy, Precision, Recall and F1-Score. In addition, to comparing the classifiers performance, the effect of different granularity grid cell sizes will also be compared. The tables below show the models' evaluation for 10km, 20km, and 30km. These are a mean of all 11 species. Every species' individual evaluation can be found in the appendix figures 1,2+3.

	Classifier	Accuracy	Precision	Recall	F1-Score
0	Naive Bayes	75.090909	70.000000	93.909091	79.090909
1	SVC	86.272727	93.818182	88.636364	91.000000
2	Decision Tree	82.363636	88.545455	87.909091	88.090909
3	K Nearest Neighbor	86.363636	93.090909	88.818182	90.727273
4	Random Forest	87.000000	93.363636	89.363636	91.181818
5	Neural Network	86.181818	91.545455	89.818182	90.545455

Figure 77: 30km Classifier Results

	Classifier	Accuracy	Precision	Recall	F1-Score
0	Naive Bayes	70.272727	63.818182	88.636364	73.090909
1	SVC	81.454545	90.363636	83.636364	86.545455
2	Decision Tree	79.181818	84.818182	84.272727	84.363636
3	Nearest Neighbor	82.727273	89.000000	85.454545	87.000000
4	Random Forest	83.909091	91.000000	85.818182	88.181818
5	Neural Network	83.454545	88.909091	86.545455	87.545455

Figure 78: 20km Classifier Results

	Classifier	Accuracy	Precision	Recall	F1-Score
0	Naive Bayes	66.545455	65.000000	72.000000	65.454545
1	SVC	76.818182	79.727273	75.454545	77.363636
2	Decision Tree	73.636364	73.363636	72.363636	73.090909
3	Nearest Neighbor	78.272727	76.000000	78.454545	77.181818
4	Random Forest	79.454545	78.454545	79.363636	78.818182
5	Neural Network	78.909091	77.909091	79.000000	78.545455

Figure 79: 10km Classifier Results

5.1.1. Accuracy

Accuracy refers to the total number of correct predictions divided by the number of cases. The scores for accuracy for the classifiers for each of the cell sizes used are shown below.

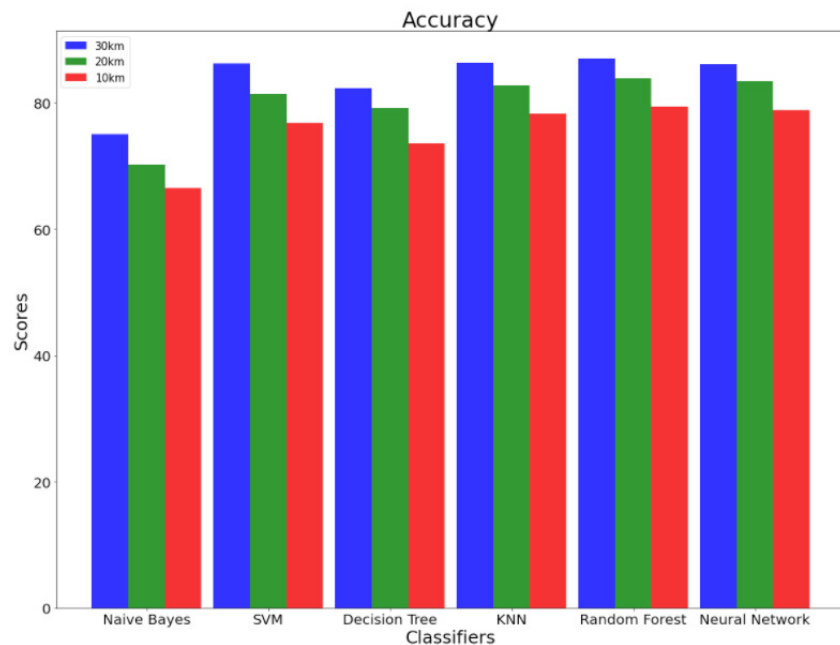


Figure 80: Bard Chart of accuracy for 10, 20, 30km for each classifier

Analysis of the results shows that aside from Naïve Bayes, they all achieved similar results between 82 - 87 for 30 km, 79 – 84 for 20km and 73 – 80 for 10km. Random Forest performed slightly better on all sizes, with Neural Network and K-Nearest Neighbour a close second and third. Whilst, Naïve Bayes consistently had an accuracy score of around 10% less.

5.1.2. Precision

Precision defines true positives divided by true positives and false positives. The scores for precision of the classifiers for each cell size is illustrated below. The results are between 0 and 100. Where 100 indicates optimal precision.

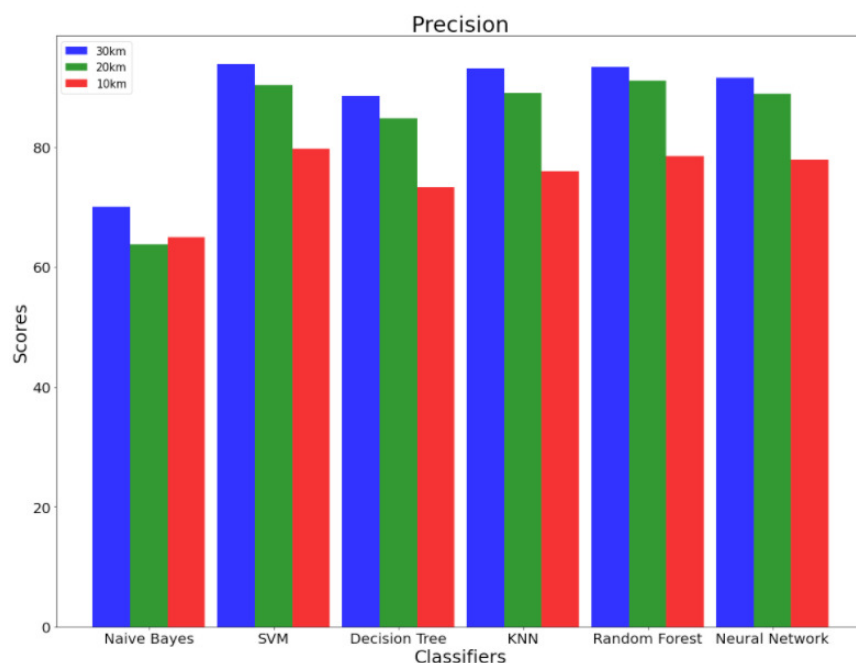


Figure 81: Bar Chart of Precision for 10, 20, 30km for each classifier

As you can see, Support Vector Machines, K-Nearest Neighbour, Random Forest and Neural Network achieved a precision of larger than 91 for 30km, >88 for 20km and >76 for 10km. Indicating that these classifiers were more successful at not labelling a 'present' cell as 'absent'.

5.1.3. Recall

Recall signifies the classifiers ability to predict positive samples in a dataset. The scores for recall of the classifiers for each of the cell sizes are as demonstrated.

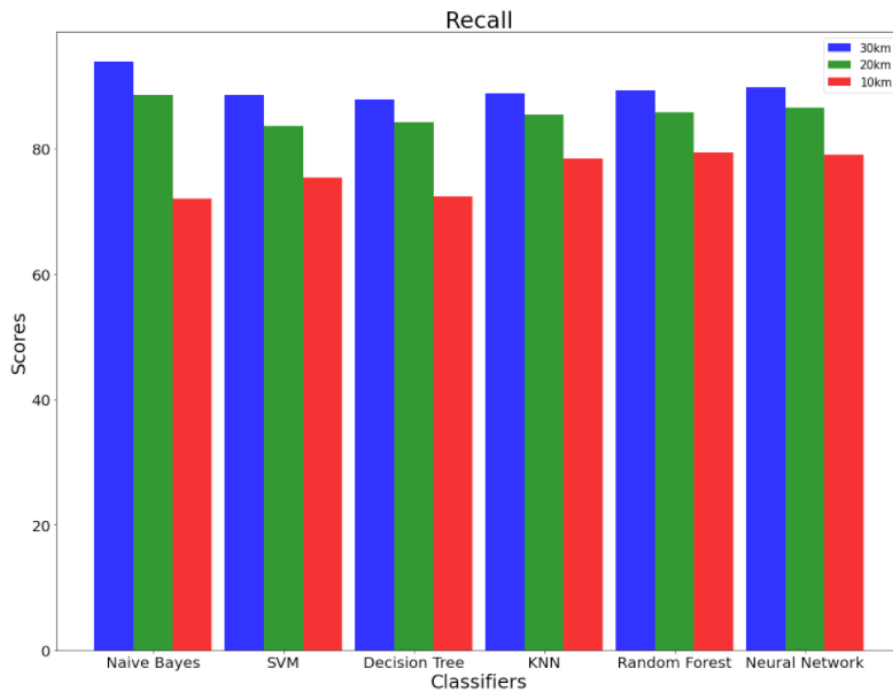


Figure 82: Bar Chart of Recall for 10, 20, 30km for each classifier

Inspection shows that all classifiers in this instance are successfully predicting a 'present' species as 'present', especially Naïve Bayes who outperformed the other classifiers for 30km and 20km grid cell sizes. Therefore, all the classifiers had a low chance of labelling the presence of a species as 'absent'.

5.1.4. F1- Score

F1-Score is the most valuable evaluation metric used in this project to determine the classifiers performance, it takes into account false positives and false negatives. The value lies between 0 and 100, where 100 signifies a perfect performance. In this scenario, the higher F1-Score indicates the majority of presences successfully predicted. The scores for the F1 value of the classifiers for each of the cell sizes is shown below.

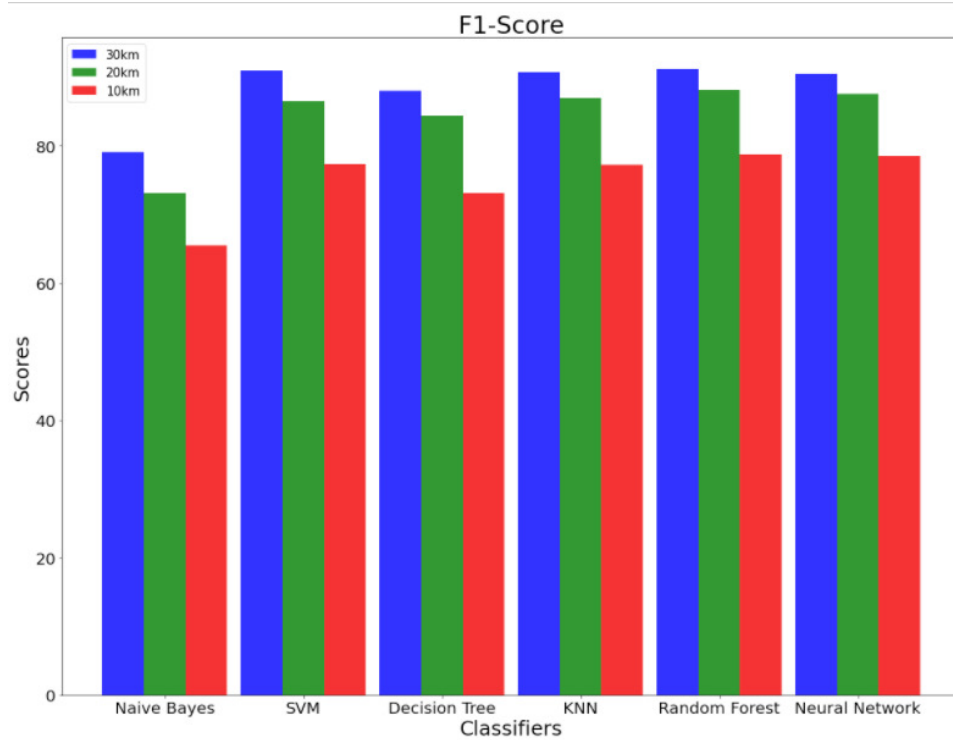


Figure 83: Bar Chart of F1-Score for 10, 20, 30km for each classifier

The graph shows that all the classifiers for all grid cell sizes achieved a relatively high F1-Score of over 65. Whilst Random Forest on all three accounts had the highest, Neural Network was a close second for 10km and 20km, Support Vector Machine for 30km. K Nearest Neighbor also achieved acceptable results. Leading us to believe these are the top performing classifiers used in this project.

In conclusion, Random Forest, Neural Network, K-Nearest Neighbor and Support Vector Machine performed consistently well given the evaluation metrics used with not much separating the scores. Although, Random Forest came out on top, Neural Network was expected to achieve a higher accuracy. These expectations may not have been met due to the parameters used, and further research and experimentation may lead to improved results. The worst performing algorithm Naïve Bayes may be due to its assumption that the features are independent of each other.

5.1.5. Grid Cell Sizes

Overall it is obvious from the results that the size of the grid cell has a direct impact on the classifiers performance. As we can see, as the grid cell size increases, its classification performance increases also. There can be numerous reasons behind this correlation. These include, presuming the classifier recognises the correlation between the Flickr data and the NBN ground truth data, there is a higher likelihood with a larger

cell that with a known Flickr presence the classifier will also predict present. In which a high precision would be expected. Also, as the cell size gets finer, the task of prediction becomes increasingly harder for the classifier as there is a less likely possibility of a ground truth record compared to when the cells are larger. Therefore for a smaller cell the classifier must be more precise at distinguishing between environmental features and finding links which is difficult if the correlation is more tenuous.

5.2. Environmental Features

A sub experiment of this project was to identify the importance of the environmental features used. From section 5.1. it was clear that Random Forest was the best all-around classifier, therefore it was used for this experiment. As a middle ground, the focus was on a 20km grid cell size. As illustrated below, the environmental features were inputted individually and with different combinations into the classifier. The results show the mean for all 11 species used.

	Environmental Features	Accuracy	Precision	Recall	F1-Score
0	Mean Temp	68.454545	74.272727	76.363636	75.181818
1	Mean Rainfall	67.000000	75.636364	73.636364	74.454545
2	Mean Wind	67.636364	76.181818	74.272727	75.090909
3	Emissions	69.818182	78.181818	75.454545	76.636364
4	Land Cover	75.363636	79.727273	82.090909	80.090909
5	Land Cover Sub	74.272727	80.909091	79.636364	79.727273
6	Population	71.090909	76.454545	78.000000	77.090909
7	Climate	79.545455	87.727273	82.545455	85.181818
8	Both Land Cover	75.090909	79.363636	81.909091	80.000000
9	Both Land Cover + Pop + emissions	77.818182	85.636364	81.181818	83.272727

Figure 84: Environmental Features Results

To clarify on the environmental features tested:

Mean Temp – Temperature mean for 12 months

Mean Rainfall – Rainfall mean for 12 months

Mean Wind – Windspeed mean for 12 months

Emissions – Emissions

Land Cover – Land Cover dominant aggregate class

Land Cover Sub – Land Cover dominant subclass

Population – Human Population

Climate – Combination of Mean Temp, Mean Rainfall and Mean Wind

Both Land Cover – Combination of Land Cover aggregate class and subclass

Both Land Cover + Pop + emissions – Combination of Land Cover aggregate class and subclass, Population and Emissions.

From analysing the results, it was found that individually the Land Cover data performed the best, which was predicted as a species occurrence will be reliant on the surface in which they thrive. Although, Mean temperature, Rainfall and Wind may appear to not be as affective features by themselves, a combination of all the climate data (Mean Temp, Mean Rainfall, Mean Wind Speed) achieved the highest accuracy and F1-score. All the environmental features selected for use in this project achieved an accuracy score of over 67, meaning they were sufficient features overall to use when predicting the occurrence of a wildlife species. A combination of all of these achieved the best performance, which has been proven from the results in section 5.1.

5.1.5. Species Environment- Further Analysis

Harvest Mouse

Harvest Mouse are found predominantly in the South of the UK in grass and arable land. Therefore land cover should do well at predicting the presence of this species. They do not have a high survival rate over the winter months due to the harsher climate.

Reasons for this being they are not adapted to these conditions as a whole and if the crops they live in die it leaves them vulnerable to predators. Hence, why they are located in the southern region where there is a warmer climate, less wind, and rain.

Given this, I predict ‘Climate’ will achieve the best results. My findings support this hypothesis with land cover and climate being the best indicator of their presence.

	Environment	Accuracy	Precision	Recall	F1-Score
0	Mean Temp	75.0	69.0	75.0	72.0
1	Mean Rainfall	66.0	67.0	63.0	65.0
2	Mean Wind	72.0	77.0	69.0	73.0
3	Emissions	73.0	70.0	72.0	71.0
4	Land Cover	79.0	80.0	78.0	79.0
5	Land Cover Sub	77.0	78.0	75.0	77.0
6	Population	72.0	68.0	71.0	70.0
7	Climate	89.0	89.0	89.0	89.0
8	Both Land Cover	78.0	80.0	76.0	78.0
9	Both Land Cover + Pop + emissions	82.0	83.0	79.0	81.0

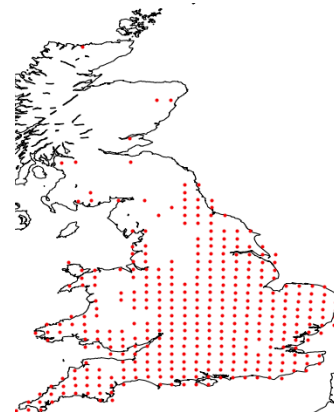


Figure 85: Harvest Mouse Environment Features Results

Red Grouse

Red Grouse is a medium sized bird with extra layers of downy feathers thus making them more comfortable in colder areas which is found in the North of the UK, mainly Scotland. They also have feathers covering their nostrils allowing for warmer air to be breathed in. [46]. For these reasons temperature in particular should be an excellent predictor of their presence along with a combination of climate data achieving the optimal accuracy. As you can see below the results found support this.

	Environment	Accuracy	Precision	Recall	F1-Score
0	Mean Temp	82.0	82.0	88.0	85.0
1	Mean Rainfall	68.0	70.0	76.0	73.0
2	Mean Wind	62.0	60.0	72.0	66.0
3	Emissions	59.0	65.0	67.0	66.0
4	Land Cover	67.0	53.0	88.0	66.0
5	Land Cover Sub	64.0	55.0	79.0	65.0
6	Population	57.0	58.0	67.0	62.0
7	Climate	92.0	91.0	95.0	93.0
8	Both Land Cover	67.0	53.0	88.0	66.0
9	Both Land Cover + Pop + emissions	69.0	68.0	79.0	73.0

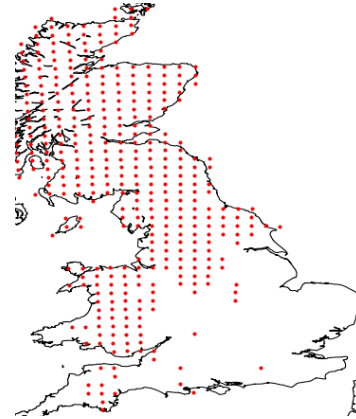


Figure 86: Red Grouse Environmental Feature Results

Chapter 6: Future Works

There are numerous ways in which the project may have been developed further, which I would have enjoyed had I'd been given more time.

Firstly, expanding on the environmental features used in this project. Although the datasets collected were sufficient for training the classifier, there was scope for improvement. Several datasets would have been interesting to add to this project, such as Terrain and Soil Type. However, these datasets were either in a format that I was unable to use or requested payment. If I were to take this project further I would access these datasets and incorporate them into my work to test if there were a negative or positive impact on the accuracy already achieved. I predict an improved effectiveness as machine learning models benefit from larger datasets.

Secondly, comparing more grid cell sizes in particular finer granularity. It would be interesting to see how much the accuracy decreases, the finer-grained the cell becomes. I would also like to do further research on why this happens. Although working adequately for the sizes used, there is a limitation of my current proposed

implementation as if you were to use a finer grain grid cell size it would lead to an extensive running time, therefore this would need to be improved.

Thirdly, this project focuses on the UK, but you can expand this scope to Europe or around the world as a whole. This could involve looking at a broader range of species. There are environmental feature data available that covers continents and the entire world however, I am unsure how much variety is available and whether it would come at a cost.

Also, there is a multitude of classifiers provided by Scikit Learn and it would be of interest to experiment with more. Although six were implemented for this project which covers a wide range, there may have been a missed classifier that would have achieved better results. Given more time, I would have chosen to implement a logistic regression algorithm.

Penultimately, while pre-processing the data for the machine learning algorithms, I used Scikit-Learn train-test split. In brief, this splits the data according to the given ratio. This is a common method used, nevertheless, it introduces bias as the model is trained on the data set aside for training which may not be as representative as possible, meaning if you were to rerun the split and the classifier again you may receive different results. To optimise on this in the future, there are a few cross-validation methods also provided by Scikit-Learn that can be used. These include KFold and StratifiedKFold. In this instance, it would split the dataset into K folds, each of the folds are used for validation while K-1 folds is utilised as the training set. [34].

Finally, for this project there was not much Flickr data available for the species selected. In future projects instead of using exclusively Flickr I would incorporate other platforms such as Twitter and Instagram. This would give me a larger and more representative dataset to use, thus leading to improved accuracy of the machine learning models.

Chapter 7: Conclusion

In conclusion, this project aims to investigate the use of machine learning classifiers along with environmental features and social media for species distribution. I also wanted to experiment into the importance of these features and investigate varying combinations to test their effectiveness. Fortunately, all the requirements set in my

initial report were met and achieved including desirable aims which were optional if time allowed. These included implementing a deep learning algorithm and comparing three grid cell sizes.

A key discovery from this project is the precedence of finding the most effective features that can aid the prediction of the presence of a species. Data collected for this project was extracted from a number of sources including a social media platform Flickr, National Biodiversity Network Atlas, Digimap etc. A variety of species were used to allow for a diverse range of results providing a more complete overview of how well the classifiers performed.

A multitude of tests were performed on six different classifiers to evaluate their performances and determine the most effective model. Five of these are machine learning algorithms along with one deep learning model. I implemented Naïve Bayes, Random Forest, Decision Trees, K Nearest Neighbor, Support Vector Machine and Neural Network. The evaluation metrics used include accuracy, precision, recall and F1-Score to evaluate and compare their performance.

As you can see from the results portion of the report, the results achieved were respectable for all classifiers. They also followed the predicted pattern of their accuracy growing as granularity increased. For 10km, the mean of their F1-Scores ranges from 65 to 79. For 20km, the mean of their F1-Scores ranges from 73 to 88 and for 30km the mean of their F1-Scores range from 79 to 91. In comparison, Naïve Bayes' performance was just below the other classifiers, whereas Random Forest performed the best with Neural Network a close second. For environmental features, it was clear that land cover data was the best individually for predicting the occurrence of a species whereas a combination of all the climate data together seen the best results.

However, there are multiple other algorithms that need be implemented and tested to ensure Random Forest fares the best along with environmental features that may be deemed more successful.

Chapter 8: Reflection on Learning

From completing this project, I have come to the realisation that there was no comparison between this and other university projects, as this project has required consistent amounts of concentration and hard work. From start to finish, I have

encountered many challenges in which to overcome. For example, due to Machine Learning being a topic I was unfamiliar with prior to taking on this project and the short time frame given to complete. Nevertheless, this just enhanced my learning experience and expanded my knowledge on machine learning algorithms, python libraries, pre-processing data and evaluation methods.

As mentioned previously, I split the project into iterations which involved collecting the data, pre-processing the data, implementing the classifier algorithms and comparing performances. However, I was concerned I spent too much time collecting the data as it was difficult to locate, and my Flickr API was only retrieving one page of data at a time and then repeating itself for a while, which led to me being unsure halfway through the semester whether I would have enough time to complete the project to the specification I had mentioned in my initial plan. Therefore, by improving my time management skills I organised myself to not just meet the requirements I set, but also my desirables set in the initial report. A crucial error on my behalf was not writing the report alongside the implementation process whilst my project was progressing. This meant I had to go back over things to refresh my memory which wasted precious time that would have been better spent making improvements.

Improvements I would have made are as follows. The species used in this project were selected from the UK Biodiversity Action Plan (BAP). However, looking back these species didn't have much data available on Flickr. Also, as mentioned previously the use of train-test split introduces bias. If I had prior machine learning knowledge, I wouldn't have used this method and implemented an alternative. Although there is room for enhancing the accuracy of the models, I am not disappointed with my results and believe they are more than adequate.

Looking back, I thoroughly enjoyed completing this project as I believe it to have valuable real world applications looking at environmental features and their impact on species occurrence along with the use of social media involving setting up my first API. Prior to starting Cardiff University 4 years ago I had no knowledge of coding or how I would embrace a project such as this, so I am pleased to see how I have developed my skills over the years and the achievements I have made this far.

References

- [1] En.wikipedia.org. 2021. *Geographic coordinate system - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Geographic_coordinate_system [Accessed 17 May 2021]
- [2] En.wikipedia.org. 2021. *Grid reference system - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Grid_reference_system [Accessed 17 May 2021]
- [3] NBN Atlas. 2021. *About the NBN Atlas*. [online] Available at: <https://nbnatlas.org/about-nbn-atlas/> [Accessed 17 May 2021]
- [4] Jncc.gov.uk. 2021. *UK BAP Priority Species / JNCC - Adviser to Government on Nature Conservation*. [online] Available at: <https://jncc.gov.uk/our-work/uk-bap-priority-species/> [Accessed 17 May 2021]
- [5] Flickr.com. 2021. *About Flickr*. [online] Available at: <https://www.flickr.com/about> [Accessed 17 May 2021]
- [6] Catt, R. and & rarr;, V., 2021. *100,000,000 geotagged photos (plus) / code.flickr.com*. [online] Code.flickr.net. Available at: <https://code.flickr.net/2009/02/04/100000000-geotagged-photos-plus/> [Accessed 17 May 2021]
- [7] Sales-i.com. 2021. *Machine Learning Explained / Blog / sales-i*. [online] Available at: <https://www.sales-i.com/blog/machine-learning-explained> [Accessed 17 May 2021].
- [8] Sciencedirect.com. 2021. *Machine Learning - an overview / ScienceDirect Topics*. [online] Available at: <https://www.sciencedirect.com/topics/psychology/machine-learning> [Accessed 17 May 2021].
- [9] Brownlee, J., 2021. *Supervised and Unsupervised Machine Learning Algorithms*. [online] Machine Learning Mastery. Available at:

<https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/> [Accessed 17 May 2021]

[10] Educative: Interactive Courses for Software Developers. 2021. *Definition: Model fitting*. [online] Available at: <https://www.educative.io/edpresso/definition-model-fitting> [Accessed 17 May 2021]

[11] Chauhan, N., 2021. *Decision Tree Algorithm, Explained - KDnuggets*. [online] KDnuggets. Available at: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html> [Accessed 17 May 2021]

[12] Sciencedirect.com. 2021. *Random Forest - an overview | ScienceDirect Topics*. [online] Available at: <https://www.sciencedirect.com/topics/engineering/random-forest> [Accessed 17 May 2021]

[13] Medium. 2021. *Support Vector Machine — Introduction to Machine Learning Algorithms*. [online] Available at: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47> [Accessed 17 May 2021]

[14] Medium. 2021. *What is the K-Nearest Neighbor?*. [online] Available at: <https://towardsdatascience.com/what-is-the-k-nearest-neighbor-862a6a30e5dc> [Accessed 17 May 2021]

[15] Chauhan, N., 2021. *Naïve Bayes Algorithm: Everything you need to know - KDnuggets*. [online] KDnuggets. Available at: <https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html> [Accessed 17 May 2021]

[16] En.wikipedia.org. 2021. *Multilayer perceptron - Wikipedia*. [online] Available at: https://en.wikipedia.org/wiki/Multilayer_perceptron [Accessed 17 May 2021]

- [17] Medium. 2021. *Metrics to Evaluate your Machine Learning Algorithm*. [online] Available at: <https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234> [Accessed 17 May 2021]
- [18] Numpy.org. 2021. *NumPy*. [online] Available at: <https://numpy.org> [Accessed 18 May 2021]
- [19] Matplotlib.org. 2021. *Welcome to the Matplotlib Basemap Toolkit documentation — Basemap Matplotlib Toolkit 1.2.1 documentation*. [online] Available at: <https://matplotlib.org/basemap/> [Accessed 18 May 2021].
- [20] Seaborn.pydata.org. 2021. *seaborn: statistical data visualization — seaborn 0.11.1 documentation*. [online] Available at: <https://seaborn.pydata.org> [Accessed 18 May 2021]
- [21] Docs.python.org. 2021. *pickle — Python object serialization — Python 3.9.5 documentation*. [online] Available at: <https://docs.python.org/3/library/pickle.html> [Accessed 18 May 2021]
- [22] J. Zhang and S. Li (2017). "A Review of Machine Learning Based Species' Distribution Modelling," International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII), Wuhan, China, 2017, pp. 199-206
- [23] Jeawak, S. S., Jones, C. B., Schockaert, S., 2017. *Using Flickr for characterizing the environment: an exploratory analysis*. In: *13th International Conference on Spatial Information Theory*, COSIT 2017. Vol. 86 of Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pp. 21:1–21:13.
- [24] Jeawak, S.S., C.B. Jones, S. Schockaert (2018) 'Mapping wildlife species distribution with social media: Augmenting text classification with species names'. GIScience 2018. Leibniz International Proceedings in Informatics.

- [25] Standardization, F., 2021. *Feature Scaling / Standardization Vs Normalization*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/> [Accessed 18 May 2021]
- [26] Foundation, C., 2021. *Birds*. [online] CK-12 Foundation. Available at: <https://www.ck12.org/c/life-science/birds/lesson/Birds-MS-LS/> [Accessed 19 May 2021]
- [27] Britannica Kids. 2021. *mammal*. [online] Available at: <https://kids.britannica.com/kids/article/mammal/353414> [Accessed 19 May 2021]
- [28] National Geographic. 2021. *Reptile Pictures & Facts*. [online] Available at: <https://www.nationalgeographic.com/animals/reptiles> [Accessed 19 May 2021]
- [29] National Geographic. 2021. *Amphibian Pictures & Facts*. [online] Available at: <https://www.nationalgeographic.com/animals/amphibians> [Accessed 19 May 2021]
- [30] Medium. 2021. *Understanding the Bias-Variance Tradeoff*. [online] Available at: <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229> [Accessed 19 May 2021]
- [31] Digimap.edina.ac.uk. 2021. [online] Available at: https://digimap.edina.ac.uk/webhelp/environment/data_information/lcm2000_land_c_over_map_of_great_britain.pdf [Accessed 19 May 2021]
- [32] SearchSoftwareQuality. 2021. *What is iterative development? - Definition from WhatIs.com*. [online] Available at: <https://searchsoftwarequality.techtarget.com/definition/iterative-development> [Accessed 19 May 2021]
- [33] Pats.cs.cf.ac.uk. 2021. *Project Allocation & Tracking System (PATs)*. [online] Available at: <https://pats.cs.cf.ac.uk/doku> [Accessed 19 May 2021]

[34] Scikit-learn.org. 2021. *sklearn.model_selection.KFold* — *scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html [Accessed 19 May 2021]

[35] Scikit-learn.org. 2021. *sklearn.metrics.accuracy_score* — *scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html [Accessed 19 May 2021]

[36] Scikit-learn.org. 2021. *sklearn.metrics.precision_score* — *scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html [Accessed 19 May 2021]

[37] Scikit-learn.org. 2021. *sklearn.metrics.recall_score* — *scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html [Accessed 19 May 2021]

[38] Scikit-learn.org. 2021. *sklearn.metrics.f1_score* — *scikit-learn 0.24.2 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html [Accessed 19 May 2021]

[39] PyPI. 2021. *convertbng*. [online] Available at: <https://pypi.org/project/convertbng/> [Accessed 19 May 2021]

[40] Standardization, F., 2021. *Feature Scaling / Standardization Vs Normalization*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/> [Accessed 20 May 2021]

[41] Medium. 2021. *Naive Bayes Classifier—Explained*. [online] Available at: <https://towardsdatascience.com/naive-bayes-classifier-explained-50f9723571ed> [Accessed 20 May 2021]

[42] Medium. 2021. *Pros and cons of various Classification ML algorithms*. [online] Available at: <https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6> [Accessed 20 May 2021]

[43] O'Reilly Online Learning. 2021. *Machine Learning with Swift*. [online] Available at: <https://www.oreilly.com/library/view/machine-learning-with/9781787121515/697c4c5f-1109-4058-8938-d01482389ce3.xhtml> [Accessed 20 May 2021]

[44] Medium. 2021. *A Quick Introduction to K-Nearest Neighbors Algorithm*. [online] Available at: <https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7> [Accessed 20 May 2021]

[45] GreatLearning Blog: Free Resources what Matters to shape your Career!. 2021. *Random Forest Algorithm- An Overview | Understanding Random Forest*. [online] Available at: <https://www.mygreatlearning.com/blog/random-forest-algorithm/> [Accessed 20 May 2021]

[46] D'Artagnan. 2021. *All About Wild Scottish Grouse | D'Artagnan*. [online] Available at: <https://www.dartagnan.com/wild-scottish-grouse-recipes-and-uses.html> [Accessed 22 May 2021]

Appendix

Figure 1- 30km Species Results

	Classifier	Metrics	Adder	Grass Snake	Great Crested Newt	Harvest Mouse	Hawfinch	Tundra Swan	Pole Cat	Red Grouse	Twite	Willow Tit	Yellow Wagtail
0	Naive Bayes	Accuracy	89.0	71.0	70.0	73.0	64.0	79.0	74.0	81.0	84.0	71.0	70.0
1	Naive Bayes	Precision	99.0	55.0	56.0	60.0	57.0	69.0	79.0	79.0	93.0	58.0	65.0
2	Naive Bayes	Recall	90.0	94.0	100.0	91.0	95.0	98.0	88.0	95.0	90.0	94.0	98.0
3	Naive Bayes	F1-Score	94.0	69.0	72.0	73.0	71.0	81.0	83.0	86.0	91.0	72.0	78.0
0	SVM	Accuracy	89.0	91.0	84.0	91.0	75.0	87.0	81.0	87.0	91.0	88.0	85.0
1	SVM	Precision	96.0	96.0	89.0	89.0	90.0	93.0	100.0	87.0	100.0	93.0	99.0
2	SVM	Recall	92.0	89.0	89.0	96.0	81.0	87.0	81.0	95.0	91.0	88.0	86.0
3	SVM	F1-Score	94.0	93.0	89.0	92.0	85.0	90.0	89.0	91.0	95.0	91.0	92.0
0	Decision Tree	Accuracy	81.0	85.0	81.0	88.0	79.0	82.0	71.0	93.0	82.0	83.0	81.0
1	Decision Tree	Precision	86.0	96.0	92.0	87.0	84.0	91.0	76.0	94.0	86.0	91.0	91.0
2	Decision Tree	Recall	92.0	82.0	83.0	92.0	88.0	83.0	86.0	97.0	93.0	84.0	87.0
3	Decision Tree	F1-Score	89.0	89.0	87.0	89.0	86.0	87.0	81.0	95.0	90.0	87.0	89.0
0	KNN	Accuracy	85.0	87.0	87.0	92.0	79.0	81.0	83.0	90.0	91.0	88.0	87.0
1	KNN	Precision	91.0	96.0	92.0	92.0	89.0	90.0	96.0	88.0	100.0	93.0	97.0
2	KNN	Recall	92.0	84.0	89.0	94.0	85.0	83.0	85.0	98.0	91.0	88.0	88.0
3	KNN	F1-Score	92.0	89.0	90.0	93.0	87.0	86.0	90.0	93.0	95.0	91.0	92.0
0	RF	Accuracy	84.0	87.0	87.0	90.0	82.0	84.0	83.0	92.0	91.0	89.0	88.0
1	RF	Precision	92.0	94.0	97.0	91.0	91.0	90.0	92.0	93.0	99.0	91.0	97.0
2	RF	Recall	90.0	85.0	86.0	92.0	86.0	87.0	88.0	97.0	92.0	91.0	89.0
3	RF	F1-Score	91.0	89.0	91.0	91.0	89.0	88.0	90.0	95.0	95.0	91.0	93.0
0	MLP	Accuracy	83.0	87.0	88.0	88.0	83.0	84.0	84.0	89.0	85.0	88.0	89.0
1	MLP	Precision	88.0	92.0	92.0	87.0	93.0	95.0	92.0	90.0	91.0	91.0	96.0
2	MLP	Recall	93.0	86.0	90.0	92.0	87.0	83.0	89.0	95.0	92.0	90.0	91.0
3	MLP	F1-Score	90.0	89.0	91.0	89.0	90.0	89.0	90.0	92.0	92.0	90.0	94.0

Figure 2 - 20km Species Results

	Classifier	Metrics	Adder	Grass Snake	Great Crested Newt	Harvest Mouse	Hawfinch	Tundra Swan	Pole Cat	Red Grouse	Twite	Willow Tit	Yellow Wagtail
0	Naive Bayes	Accuracy	77.0	70.0	75.0	78.0	70.0	71.0	64.0	78.0	62.0	69.0	59.0
1	Naive Bayes	Precision	95.0	49.0	61.0	73.0	62.0	50.0	69.0	76.0	61.0	56.0	50.0
2	Naive Bayes	Recall	80.0	87.0	99.0	85.0	95.0	93.0	78.0	84.0	84.0	96.0	94.0
3	Naive Bayes	F1-Score	87.0	63.0	76.0	79.0	75.0	65.0	73.0	80.0	71.0	70.0	65.0
0	SVM	Accuracy	81.0	86.0	80.0	86.0	76.0	80.0	70.0	91.0	76.0	85.0	85.0
1	SVM	Precision	100.0	87.0	84.0	84.0	90.0	87.0	82.0	92.0	100.0	95.0	93.0
2	SVM	Recall	81.0	87.0	84.0	92.0	79.0	79.0	77.0	92.0	76.0	84.0	89.0
3	SVM	F1-Score	89.0	87.0	84.0	87.0	84.0	83.0	79.0	92.0	87.0	89.0	91.0
0	Decision Tree	Accuracy	74.0	79.0	78.0	89.0	71.0	78.0	68.0	89.0	75.0	87.0	83.0
1	Decision Tree	Precision	81.0	80.0	79.0	88.0	83.0	85.0	76.0	95.0	88.0	91.0	87.0
2	Decision Tree	Recall	86.0	80.0	85.0	92.0	79.0	77.0	79.0	87.0	81.0	90.0	91.0
3	Decision Tree	F1-Score	83.0	80.0	82.0	90.0	81.0	81.0	77.0	91.0	84.0	90.0	89.0
0	KNN	Accuracy	84.0	83.0	83.0	84.0	79.0	81.0	70.0	91.0	81.0	87.0	87.0
1	KNN	Precision	97.0	89.0	84.0	84.0	89.0	90.0	79.0	88.0	93.0	93.0	93.0
2	KNN	Recall	85.0	80.0	89.0	88.0	84.0	78.0	79.0	96.0	83.0	88.0	90.0
3	KNN	F1-Score	91.0	84.0	86.0	86.0	86.0	83.0	79.0	92.0	88.0	90.0	92.0
0	RF	Accuracy	84.0	87.0	85.0	87.0	77.0	81.0	75.0	92.0	78.0	89.0	88.0
1	RF	Precision	96.0	90.0	87.0	85.0	87.0	92.0	87.0	93.0	95.0	95.0	94.0
2	RF	Recall	86.0	86.0	89.0	92.0	82.0	77.0	79.0	93.0	80.0	89.0	91.0
3	RF	F1-Score	91.0	88.0	88.0	88.0	84.0	84.0	83.0	93.0	87.0	92.0	92.0
0	MLP	Accuracy	86.0	83.0	86.0	84.0	79.0	82.0	69.0	93.0	84.0	86.0	86.0
1	MLP	Precision	96.0	89.0	85.0	83.0	90.0	89.0	78.0	94.0	93.0	89.0	92.0
2	MLP	Recall	88.0	80.0	92.0	89.0	83.0	80.0	79.0	94.0	87.0	90.0	90.0
3	MLP	F1-Score	92.0	84.0	88.0	86.0	86.0	84.0	78.0	94.0	90.0	90.0	91.0

Figure 3- 10km Species Results

	Classifier	Metrics	Adder	Grass Snake	Great Crested Newt	Harvest Mouse	Hawfinch	Tundra Swan	Pole Cat	Red Grouse	Twite	Willow Tit	Yellow Wagtail
0	Naive Bayes	Accuracy	59.0	71.0	73.0	76.0	71.0	70.0	58.0	62.0	61.0	69.0	62.0
1	Naive Bayes	Precision	74.0	55.0	56.0	76.0	69.0	31.0	78.0	93.0	79.0	58.0	46.0
2	Naive Bayes	Recall	64.0	78.0	88.0	65.0	72.0	70.0	53.0	54.0	62.0	88.0	98.0
3	Naive Bayes	F1-Score	69.0	64.0	68.0	70.0	71.0	43.0	63.0	69.0	70.0	70.0	63.0
0	SVM	Accuracy	61.0	85.0	81.0	82.0	72.0	74.0	64.0	89.0	75.0	81.0	81.0
1	SVM	Precision	84.0	85.0	81.0	79.0	74.0	72.0	66.0	88.0	76.0	86.0	86.0
2	SVM	Recall	64.0	83.0	82.0	73.0	70.0	63.0	59.0	88.0	78.0	84.0	86.0
3	SVM	F1-Score	72.0	84.0	81.0	76.0	72.0	67.0	63.0	88.0	77.0	85.0	86.0
0	Decision Tree	Accuracy	64.0	81.0	77.0	75.0	65.0	66.0	64.0	85.0	69.0	84.0	80.0
1	Decision Tree	Precision	71.0	80.0	80.0	65.0	70.0	47.0	60.0	85.0	74.0	91.0	84.0
2	Decision Tree	Recall	70.0	80.0	76.0	66.0	64.0	54.0	60.0	82.0	72.0	85.0	87.0
3	Decision Tree	F1-Score	71.0	80.0	78.0	66.0	67.0	51.0	60.0	84.0	73.0	88.0	86.0
0	KNN	Accuracy	65.0	86.0	84.0	82.0	72.0	73.0	67.0	89.0	75.0	84.0	84.0
1	KNN	Precision	72.0	86.0	84.0	74.0	69.0	54.0	60.0	87.0	75.0	89.0	86.0
2	KNN	Recall	71.0	85.0	84.0	75.0	73.0	66.0	64.0	89.0	80.0	86.0	90.0
3	KNN	F1-Score	72.0	85.0	84.0	75.0	71.0	60.0	62.0	88.0	77.0	87.0	88.0
0	RF	Accuracy	67.0	87.0	84.0	83.0	73.0	75.0	69.0	89.0	76.0	87.0	84.0
1	RF	Precision	77.0	85.0	83.0	78.0	74.0	58.0	62.0	86.0	79.0	93.0	88.0
2	RF	Recall	72.0	87.0	85.0	76.0	73.0	70.0	67.0	88.0	79.0	87.0	89.0
3	RF	F1-Score	74.0	86.0	84.0	77.0	73.0	63.0	65.0	87.0	79.0	90.0	89.0
0	MLP	Accuracy	66.0	86.0	82.0	83.0	72.0	75.0	69.0	91.0	75.0	84.0	85.0
1	MLP	Precision	73.0	86.0	81.0	79.0	72.0	62.0	63.0	89.0	76.0	89.0	87.0
2	MLP	Recall	72.0	85.0	84.0	76.0	72.0	67.0	67.0	90.0	79.0	86.0	91.0
3	MLP	F1-Score	73.0	86.0	83.0	77.0	72.0	64.0	65.0	89.0	78.0	88.0	89.0