COMP390

2023/24

Weather Analysis Facilitated by Unsupervised And Supervised Machine Learning Algorithms

Student Name: Aqeel Hassan

Student ID: 201608139

Supervisor Name: Blaine Keetch

DEPARTMENT OF

COMPUTER SCIENCE

University of Liverpool

Liverpool L69 3BX

Acknowledgements

A thank you to my supervisor, Dr. Keetch Blaine, for his guidance, consolation and expertise throughout the duration of this project. Mr Blaine reassured me at many points through the project and provided all the necessary assistance for me to successfully complete my project. I am truly grateful for his contributions

I am also expressing gratitude to my friends, family and the LTSO who have been a constant source of support, motivation, and encouragement. Your belief in me has fueled my determination to excel in everything I do

Abstract

Where there is global warming and resource scarcity, the imperative to accurately predict and understand weather patterns has become increasingly crucial. This project offers a tool in current weather analysis systems, offering an approach to weather pattern interpretation, with the application potentially extending to farming, mining, and sustainability prediction. The developed weather analysis system is a web application leveraging machine learning algorithms to predict the weather. By assimilating various sets of meteorological data, including temperature, humidity, wind speed, and precipitation, the system provides visual and informative predictive analytics. The intricate examination of weather attributes not only aids immediate decision-making but also contributes to long-term sustainability planning.

The central aim of this research was to enhance the credibility of weather pattern analysis.The successful implementation of the system, as validated through rigorous testing and evaluation along with creating a system for each user for their queries and data, marks a significant step toward addressing global challenges associated with weather predictions and resource management. This project shows potential for data-driven insights in addressing environmental concerns.

Statement of Ethical Compliance

Data Category: A
Participant Category: 0

I confirm that I have read the ethical guidelines and will follow them during this project.

Further details can be found in the relevant sections of this proposal

Chapter 1: Introduction

1.1 Background

Weather prediction techniques to identify weather patterns and the weather itself have been around for millenia. In 1854, Robert FitzRoy who was a Vice admiral and former captain with a naval background was responsible for developing the Met office (Met Office, n.d.).The earliest information to be collected by the office was daily weather data, recorded at 9am at set locations, known as stations, around the British and Irish coasts and sent to the Meteorological Office in London by the new telegraph system, the cutting edge technology of its day. This information consisted of pressure, temperature, wind direction and force, cloud amount, weather conditions eg. rain, thunder, fog and sea disturbance for each location. The observations were collated into Daily Weather Reports and these, renamed the Daily Weather Summary from 1980, form the longest continuous series of data in the National Meteorological Archive this data would be used to model weather patterns and harsh weathers for future voyages.

An algorithm called k-means clustering is a commonly used algorithm for finding patterns. This repetitive algorithm recalculates the mean based on surrounding values, this is a fast and optimal approach for a wide array of applications such as predictive marketing, stock market analysis, documentation, and biomedicine. Considerably a popular if not the most popular clustering algorithm, k-means is excessively used for its easy to implement, understand and effective ability.

In the domain of analysing the weather, conventional techniques depend on more archaic and less efficient approaches. As an example, analysing manually of historical weather patterns involve a time-consuming examination of data by a human, making it challenging to identify trends or patterns which are difficult to see. Mistakes can also be made. A possible solution to this is an implementation of k-means clustering which is a dynamic tool which is revolutionary in weather analysis.

K-means clustering in weather analysis allows for the autonomous identification of distinct patterns and clusters/groups within large datasets. By grouping similar weather conditions/attributes, such as temperature ranges, humidity levels, and wind speeds, meteorologists can gain valuable insights into commonly occurring patterns and trends. This idea not only improves the efficiency of analysis but also enables more accurate predictions and informed decision-making.

Consider a scenario where k-means clustering is applied to analyse for regional weather data. The algorithm can identify clusters of similar weather conditions along different seasons, helping meteorologists understand climate variations more clearly. The information is crucial for various sectors, including agriculture, disaster preparedness, and urban planning.

In conclusion, while traditional methods in weather analysis may be more process expensive and less precise, k-means clustering offers a modern and efficient alternative. By automating the identification of patterns in large weather datasets, k-means clustering contributes to more accurate and timely weather predictions, ultimately benefiting industries and communities reliant on weather information.

1.2 Problem Statement

In the present environment, the data which is very simple is recorded and predicted by the system that which is in place,they are fixated as they are limited or bounded, though they are very effective (Descriptive statistics, spatial linear analysis e.g. Linear regression), it can be shown that simpler Weather analysis is short sighted as it will limit the predictive ability of predicting the weather and may miss on more or different crucial pieces of information/insight (SciJinks, n.d.). There are studies that show within weather analysis with the use of advanced statistical models and machine learning algorithms significantly improve prediction accuracy. For example, research conducted by (Chen et al., 2023) showed that integrating machine learning techniques into weather forecasting systems can reduce prediction error, possibly for extreme weather events. This statement suggests that forward pioneering of ML can improve quality of the robustness and precision of weather predicting, addressing the constraints of simpler or more empirical

weather analysis techniques. Simplistic linear regression models are another common method which can be used for weather analysis. While linear regression gives an idea as to the relationship that is between weather attributes and their covariance, complex non linear dynamics of the atmosphere are insignificant for capture of these values.Assumes that linear relationships are between input and output variables, linear regression, which may not be true for all weather patterns. For weather analysis, relying dependently on linear regression can possibly lead to inaccurate naive predictions, especially in cases of extreme weather events or complex atmospheric interactions. Leads to lack of credibility in weather forecasting systems when it comes to accurately capturing the nuances of weather patterns. Is because there is no way to ensure the accuracy and reliability of predictions without employing more sophisticated analytical methods. One such proposed solution is the usage of Support Vector Machine (SVM) modelling alongside the usage of PCA-K-Means clustering.

1.3 Aim

Using more complex analytical methods for my project I will go about aiming to improve the confidence of weather predictions. statistical based models or machine learning algorithms to increase prediction accuracy should be used for weather predicting systems. This can rectify the cons of the existing systems that are either too naive or lack robustness, which leads to inaccurate weather forecasts. The importance of accurate weather predictions cannot be underhanded, as they apply to various domains such as agriculture, transportation/logistics, and extreme weather/disaster management.

1.4 Objectives

To achieve the aim of this project, the following objectives must be achieved:

- To be able to understand and apply concepts such as advanced statistical analysis, machine learning, and predictive modelling in the realm of weather forecasting.

- To compare various analytical methods and algorithms amongst one another that I will use in weather prediction and establish key features and limits.

- To choose suitable machine learning algorithms for weather prediction accuracy.
- To compare current weather forecasting systems.
- To look for and fix any computational or data-related issues when implementing advanced analytical methods for weather prediction.
- To pick programming languages, libraries, or packages that the system will be built using.
- To design a user interface for the project , making it accessible to meteorologists and other users.
- To create and develop a system that can analyse weather data using machine learning algorithms.
- To apply a system database with historical weather data and user queries for testing and validation purposes.
- Test the system and evaluate its performance for prediction accuracy and optimisation.
- Give functionality to users to access their previous queries for more efficiency

1.5 Research Questions

My project will answer the question from the list below:

- Can advanced machine learning techniques improve the accuracy of weather predicting?

- Are advanced analytical methods in analysing complex weather data for identifying patterns or trends clearly and concisely effective?

Chapter 2: Background Research

Weather analysis using machine learning techniques changed the way meteorological forecasting is done.Gaining insight into existing algorithms and systems designed to meet the requirements for advanced weather prediction solutions the following chapter shows an analysis of machine learning in weather analysis.

2.1 Cover of Weather Analysis Using Machine Learning

There is a need for predicting weather as time goes on. There should be an increase to lower and plan for risks of natural disasters, weather predicting can also be used to increase productivity of various sectors such as agriculture, transportation etc. Since then the application of machine learning techniques in weather analysis has become more mainstream. Machine learning offers a more streamlined path to process larger quantities of weather data and take meaningful patterns for forecasting purposes. In weather analysis, algorithms are used to process meteorological data, such as temperature, humidity, wind speed, etc extracted from various sources including satellites, weather stations, and sensors. These algorithms then find complex unobtainable by human eye relationships and patterns within the data to generate forecasts for short-term and long-term weather conditions. Weather analysis may work with challenges and constraints. Factors such as the quality of data inputs, including accuracy and coverage, can significantly impact the performance of machine learning models. Furthermore, the dynamic/varied and nonlinear nature of atmospheric processes poses objective difficulties in accurately taking and predicting the weather. Furthermore, there can be biases in the machine learning models if the training data is not representative of the range of weather patterns and conditions. Dealing with the non linearity and ensuring the reliability and fairness of weather forecasts are key considerations in the development and implementation of machine learning solutions for weather analysis. Through ongoing research and advancements in machine learning algorithms, the accuracy and reliability of weather forecasts can be continually improved, enabling better preparedness and decision-making tin response to changing weather conditions and climate variability.

2.2 Early approaches

2.2.1 Standardising Data

The initial stages of weather data analysis, to generalise data analysis by standardising data preprocessing. Standardising data involves making sure that the data is consistent in the representation of weather variables, allowing for followed analysis and prediction of values. This approach makes apparent the importance of preprocessing techniques such as normalisation and scaling the data for a more accurate modelling (Chen et al., 2023). However, standardising data could introduce limits in getting nuanced variations and extreme events, possibly changing the accuracy of weather predictions in specific scenarios.

2.2.2 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors play a crucial role in weather data analysis, particularly in techniques such as Principal Component Analysis (PCA). Eigenvalues represent the variance of data along principal components, while eigenvectors indicate the direction of maximum variance (Denton et al., 2022). By analysing eigenvalues and eigenvectors, meteorologists and climate scientists can identify dominant patterns and trends within weather datasets, providing valuable insights for prediction and modelling. However, interpreting eigenvalues and eigenvectors requires careful consideration, as they may not always capture the full complexity of weather systems and phenomena.

2.2.3 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a key technique for finding features to extract and to reduce dimensionality . PCA aims to identify the most significant patterns and trends within weather datasets by decomposing the variability into orthogonal components known as principal components (Chen et al., 2023). These principal components represent the fundamental modes of variability in weather variables such as temperature, pressure, humidity, and wind speed. However, while

PCA effectively reduces dimensionality and highlights dominant patterns, it may overlook subtle variations and nonlinear relationships present in weather data, potentially limiting its predictive capabilities in certain contexts. By retaining the principal components corresponding to the largest eigenvalues, PCA allows for the reduction of dimensionality in weather data while preserving critical information about underlying patterns. This reduction facilitates subsequent analysis and prediction tasks, enabling more efficient processing and interpretation of weather data. One of the primary advantages of PCA in weather analysis is its ability to capture complex interactions and relationships among weather variables. Through dimensionality reduction, PCA highlights the dominant modes of variability, aiding meteorologists and climate scientists in identifying key factors driving weather patterns and phenomena. However, PCA's effectiveness depends on the quality and representativeness of the data used, and improper application may lead to misleading results.

2.2.4 Feature engineering and selection For weather analysis

For an example of feature engineering PCA implementation is Fisherface, which focuses on enhancing relevant features while reducing the impact of irrelevant ones. Feature engineering involves the creation of new features or transformation of existing ones to better represent the underlying relationships in weather data. By extracting meaningful information from raw weather variables such as temperature, pressure, humidity, and wind speed, feature engineering aims to improve the performance and interpretability of weather analysis models. Similarly, feature selection techniques identify the subset of features that contribute most significantly to predictive accuracy, streamlining the modelling process and reducing the risk of overfitting. These techniques, including correlation analysis, forward/backward selection, and regularisation methods, prioritise the inclusion of relevant variables while eliminating redundant or noisy ones, thereby enhancing the discriminative power of weather analysis models. 2.2.4 Machine Learning Machine learning techniques have gained prominence in weather analysis for their ability to extract complex patterns and relationships from large-scale weather datasets. Supervised learning algorithms such as decision trees, random forests, and support vector machines (SVMs) offer effective tools for predictive modelling and pattern recognition in weather forecasting and climate research. These

algorithms learn from labelled training data to make predictions or classify weather phenomena based on observed patterns and relationships in the data. Additionally, unsupervised learning techniques like clustering and anomaly detection enable exploratory analysis and identification of hidden structures in weather datasets. By leveraging the power of machine learning, meteorologists and climate scientists can improve the accuracy and reliability of weather forecasts, enhance understanding of atmospheric dynamics, and mitigate the impacts of extreme weather events.

2.2.5 Summary of weather data analysis

Techniques for weather data analysis, various techniques have been developed to understand and predict atmospheric phenomena. While traditional methods such as statistical analysis and linear regression have provided valuable insights, they often require extensive manual intervention and are limited in their predictive capabilities. Techniques like standardising data, eigenanalysis, and principal component analysis (PCA) have emerged as powerful tools for extracting meaningful patterns and reducing the dimensionality of weather datasets. However, these traditional techniques may face challenges in capturing the full complexity of weather systems, especially in the presence of nonlinear relationships and extreme events. In contrast, modern approaches such as machine learning algorithms, including neural networks and deep learning, offer promising solutions for weather prediction tasks. These techniques can automatically learn intricate patterns and relationships from large-scale weather datasets, leading to more accurate forecasts and improved understanding of atmospheric dynamics. While traditional methods like standardising data and eigenanalysis lay the groundwork for weather data analysis, the integration of machine learning techniques represents a paradigm shift in the field. By leveraging the power of neural networks and deep learning, meteorologists and climate scientists can unlock new insights into weather patterns, enhance prediction accuracy, and mitigate the impacts of extreme weather events. Therefore, for the purposes of this project, machine learning techniques will be prioritised due to their superior performance and adaptability in addressing the project's objectives.

2.3 Weather analysis techniques

To analyse and interpret meteorological data so as to be able to make predictions and find weather patterns weather analysis techniques are used. The functionalities cover a range of algorithms and statistical methods tailored for processing weather-related data.

Weather analysis techniques are designed to analyse and interpret various meteorological data to understand weather patterns and make predictions. These techniques encompass a range of algorithms and statistical methods tailored for processing weather-related data.

2.3.1 Overview of Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, where the dependent variable is a linear function of the independent variables (Hu et al., 2019). The model aims to find the best-fitting straight line that minimises the difference between the observed and predicted values.

2.3.15 Critical Analysis of Linear Regression

Linear regression is simple and easy to understand, making it a popular choice for modelling relationships between variables. It provides insights into the strength and direction of the relationship between the variables. However, linear regression assumes linearity and independence of variables, which may not always hold true in real-world data. Additionally, it may not capture complex relationships between variables, leading to potential inaccuracies in predictions. Despite its limitations, linear regression remains a valuable tool for simple modelling tasks and initial exploratory analysis of data.

2.3.2 Hidden Markov Models

Overview of Hidden Markov Models (HMM) Hidden Markov Models (HMMs) are probabilistic models used for modelling sequences of observations. They consist of a set of states, each associated with a probability distribution over possible observations, and transitions between states governed by transition probabilities. HMMs are characterised by two types of parameters: emission probabilities, representing the likelihood of observing a particular observation given the current state, and transition probabilities, representing the likelihood of transitioning from one state to another. The "hidden" aspect refers to the fact that the underlying state sequence is not directly observable but can be inferred from the observed sequence of outputs.

2.3.25 Critical Analysis of Hidden Markov Models

HMMs are particularly useful for modelling time series data or sequences where the underlying states are not directly observable but can be inferred from observed data. They have been successfully applied in various domains, including speech recognition, natural language processing, and bioinformatics. However, HMMs have certain limitations, such as the assumption of Markovian property (i.e., the future state depends only on the current state and not on the sequence of previous states), which may not hold true in all scenarios. Additionally, HMMs may struggle with modelling long-range dependencies or capturing complex patterns in data. Despite these limitations, HMMs remain a powerful tool for sequential data modelling and have been extended and adapted in various ways to address specific challenges in different applications.

2.3.3 Overview of Support Vector Machines

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. SVM aims to find the optimal hyperplane that separates data points into different classes with the maximum margin, thereby maximising the margin of separation between classes. In classification tasks, SVM seeks to find the hyperplane that best separates the data points of different classes, while in regression tasks, SVM aims to find the hyperplane that best fits the data while minimising errors (Salcedo-Sanz et al., 2014)

.

2.3.35 Critical Analysis of Support Vector Machines

SVM is effective in handling both linearly separable and non-linearly separable datasets through the use of kernel functions, which map the input data into a higher-dimensional feature space where linear separation is possible. SVM performs well in scenarios with high-dimensional data and when the number of features exceeds the number of samples. However, SVM's computational complexity increases with the size of the dataset, making it less suitable for large-scale datasets. Additionally, SVM requires careful selection of hyperparameters, such as the choice of kernel function and regularisation parameters, which can significantly impact its performance. Despite these challenges, SVM remains a widely used algorithm in various domains, including image classification, text categorization, and bioinformatics, owing to its robustness and versatility.

2.3.4 Overview of Bayesian Networks

Bayesian Networks are probabilistic graphical models used to represent uncertain relationships between variables in a domain. They consist of a directed acyclic graph (DAG) where nodes represent random variables and edges represent probabilistic dependencies between variables. Each node in the graph is associated with a conditional probability distribution that quantifies the probability of the node's value given the values of its parent nodes (Weber et al., 2012). Bayesian Networks allow for efficient probabilistic inference, enabling reasoning under uncertainty and making predictions about unobserved variables.

2.4.45 Critical Analysis of Bayesian Networks

Bayesian Networks provide a principled framework for modelling complex dependencies and uncertainties in real-world systems. They offer transparency in representing causal relationships between variables and allow for explicit incorporation of prior knowledge through the specification of conditional probability distributions. However, constructing accurate Bayesian Networks often requires domain expertise and manual intervention to specify the structure and

parameters of the network. Additionally, inference in Bayesian Networks can be computationally demanding, especially for large networks with many variables. Despite these challenges, Bayesian Networks have been successfully applied in various domains, including healthcare, finance, and environmental modelling, where reasoning under uncertainty is crucial for decision-making. Advances in algorithms and computational techniques continue to enhance the scalability and applicability of Bayesian Networks in real-world applications.

2.3.5 Overview of PCA-K-means

The PCA-K-means hybrid model begins by applying PCA to the high-dimensional weather dataset to reduce its dimensionality while retaining most of the variance in the data. PCA transforms the original variables into a new set of orthogonal variables, known as principal components, which capture the essential features of the data. The reduced-dimensional dataset obtained from PCA is then fed into the K-means clustering algorithm. K-means clustering partitions the data into a predetermined number of clusters based on similarity, with each cluster represented by a centroid. The algorithm iteratively assigns data points to the nearest centroid and updates the centroids until convergence (Feldman, Schmidt and Sohler, 2020). By clustering similar weather patterns together, K-means enables the identification of distinct weather regimes or clusters within the dataset.

2.3.55 Critical Analysis of PCA-K-means

The PCA-K-means hybrid model offers several advantages for weather data analysis. By reducing the dimensionality of the data through PCA, the model simplifies the computational complexity and facilitates the identification of dominant patterns in the dataset. Additionally, K-means clustering provides a straightforward and interpretable way to segment the data into distinct clusters, aiding in the interpretation and visualisation of weather patterns. However, the effectiveness of the PCA-K-means hybrid model depends on several factors, including the choice of the number of principal components and clusters, as well as the interpretability of the resulting clusters. Furthermore, the model may encounter challenges in handling nonlinear relationships or complex interactions among

weather variables. Careful preprocessing and parameter tuning are essential to ensure the optimal performance of the hybrid model in weather data analysis tasks.

Comparing Weather analysis algorithms

Algorithm	Accuracy	Interpretability	Robustness	Scalability	Flexibility	Data Complexity	Overall
Linear regression	Moderate	Easy	Moderate	Strong	Moderate	Moderate	good
Bayesian networks	Low	Easy - intermediate	Strong	Moderate	Strong	Good	ok
PCA-k-means	Moderate	Intermediate	Strong	Strong	Strong	Moderate	good
SVM	Strong	Intermediate	Strong	Moderate	Strong	Moderate	good
K- nearest neighbour	Low	Easy	Moderate	Strong	Moderate	Low	ok

Table 1

Shown in table 1, Support Vector Machines (SVM) perform well between the attributes amongst the machine learning algorithms in accuracy, robustness, scalability and flexibility. PCA-K-means Hybrid shows moderate performance in most categories, with strength in robustness and scalability. For applications where high accuracy, robustness, and scalability are critical, SVM would be the preferred choice. However, for scenarios requiring a balance between these factors and moderate performance overall, PCA-K-means Hybrid could serve as a suitable alternative so chose both for an overall good capture of information and to show both an unsupervised and supervised model.

Chapter 2.5 Weather Analysis systems

A comparison of key features and limitations identified in existing weather analysis programs are shown in the following table:

System	Feature identified	Limitations
Weather Analysis of Guntur District of Andhra Region using Hybrid SVM Data Mining Techniques **(Rajasekhar, N. and V.Rajini Kanth, T. 2014)**	• Preprocesses raw weather data for analysis. • Finds patterns and groups similar weather data using K-means clustering. • SVM classification for predictive modelling based on clustered data. • evaluation metrics to assess model performance, • visualisations for understanding of data patterns and model outcomes. • regression equations for predicting future weather trends.	• SVMs are computationally intensive, challenging for real-time deployment, • difficult to interpret compared to simpler models. • Performance may vary across different regions or climatic conditions, requiring validation. • Domain expertise required
Multi-type Loads Characteristics Analysis Based on PCA-K-means Model in Extreme Weather **(Mengyuan Zhang, Yingjie Tian, Naiwang Guo, Yun Su, Yi Wu, Yingying Zhao, Deyong Yu)**	• Multi-type Loads Characteristics Analysis • PCA-K-means Model • Meteorological Data Downscaled by PCA • Identification of Principal Components • Construction of Principal Component Index • Clustering of Load Data • Validated Model	• Limited to Specific Scenarios • Dependency on Data Quality • Complexity of Model • Computational Requirements • Simplified Representation • Need for Continuous Validation • Interpretation Challenges
SHORT TERM TRAFFIC FLOW PREDICTION USING	• Uses weather conditions like sunny, rainy, precipitation, and temperature into traffic flow prediction. • Consider three parameter sets for prediction,	• Focuses only on weather parameters overlooking other influential factors. • Shortage of Training Data

MACHINE LEARNING - KNN, SVM AND ANN WITH WEATHER INFORMATION **(Faysal Ibna Rahman)**	• Optimises parameters like neighbours in KNN, kernel type in SVM, and network structure in ANN • Predictions at hourly intervals for detailed traffic pattern analysis. • Evaluates prediction accuracy using R-square value and MAPE for quantitative assessment. • Comparative analysis for different algorithms and parameter sets	• Doesn't consider dynamic factors like accidents or special events, impacting accuracy. • Effectiveness may vary across regions or road networks. • Relies on accurate weather forecasts,

Table 2

2.6 Summary

To conclude this literary review, some attributes of the specific features and ideas in existing systems would be helpful to include in my project as they would support the aims and objectives of my project. The constraints of the existing systems should be made apparent and noted so as to be aware of ensuring the drawbacks of those systems are not included in the proposed solution to the best of my ability. After having done the research, the algorithms to use for the different aspects of ML have been highlighted and shown how they are suited and relevant to the aims of this project. Moreover, the applicable concepts,algorithms and paradigms have been gone through extensively.

2.7 Proposed Solution

This project aims to develop a system that improves/pioneers the visualisation of Weather Analysis through the use of Machine learning. After conducting research,

using the conclusions driven from the tables and critical analysis, the proposed solution will include the following:

- SVM for Supervised ML
- PCA K-means for Unsupervised ML

- High accuracy
- Rich data outputs

- Ability to interact with User queries
- Queries outputted to a database

- Easy to use GUI

Chapter 3 Requirements Analysis

The requirements of the system including functional and non-function are gathered from the literature review and existing systems review.

3.1 Hardware requirements

Hardware	Purpose
Should have a adequate cpu and memory	In order to for the system to work the system would require enough memory and cpu processing for the function to run the weather analysis software
Should have some sort of internet access	In order to interact with the server or even interact with VIsual crossing database for data having access to the internet is required

Table 3

3.2 System requirements

ID	Requirement	Type	Complete
1	The system must use a form input to take in data from user	Functional	Satisfied
2	The system should be able to input the date period for data analysis	Functional	Satisfied
3	The system should be able to input the locations for data analysis	Functional	Satisfied
4	The system should be able to input the features to analyse for data analysis	Functional	Satisfied
5	System must be able to acquire data	Functional	Satisfied
6	User should see system has acquired data	Functional	Satisfied
7	System should be able to preprocess data	Functional	Satisfied
8	System should be able to analyse data	Functional	Satisfied
9	System will provide confirmation when user account has been made	Non-Functional	Satisfied
10	System GUI must be easy to use	Non-Functional	Satisfied
11	System must be able to handle user errors/mistakes by user	Functional	Satisfied
12	System must be able to handle user errors/mistakes by data	Functional	Satisfied
13	System is able to implement supervised ML	Functional	Satisfied
14	System is able to implement unsupervised ML	Functional	Satisfied
15	System should let user register	Functional	Satisfied
16	System should let user login	Functional	Satisfied
17	System should let user log out	Functional	Satisfied

18	System should create images from user input	Functional	Satisfied
19	System supervised ML should be able to have an accuracy of at least 80%	Non-Functional	Satisfied
20	System cluster weather for all countries	Functional	Satisfied
21	System cluster weather by only each country	Functional	Satisfied
22	System should keep the data secure between server and client	Non Functional	Satisfied
23	System should show outputs for query that have been asked before for logged in users	Functional	Satisfied
24	User should able to view their outputs	Functional	Satisfied
25	User shouldnt need to login to use the application	Functional	Satisfied
26	User that hasn't logged in shouldnt have access to preprocessed queries	Functional	Satisfied
27	Programmer should be able to identify and clean data after it has been collected if error is found on the data	Functional	Satisfied
28	User should be able to select arbitrarily from a pre-selected number of features	Functional	Satisfied
29	System should be able to handle error	Functional	Satisfied
30	User should be able to see visual of the output	Functional	Satisfied
31	Data should be cleansed before data analysis	Functional	Satisfied
32	Data should be preprocessed before data analysis	Functional	Satisfied
33	Data should be of a satisfactory standard before analysis	Non Functional	Satisfied
34	System should be able to process some level of Data in less than 2 minutes	Non-Functional	Satisfied

Table 4

3.3 Software requirements

* part of python library * Stand alone module from github

Software/Library/module	version	Purpose
Python	3.10.1	language used for web apps and data analysis
Flask	2.1.0	Flask is a framework which allows for web
Flask-login	0.5.0	Flask module allows for user login
Flask-sqlalchemy	3.0.0	Flask module allows for database interaction
Flask_bcrypt	0.7.1	Flask module allows from server to user
Flask-wtf	1.0.0	Falsk module allows for creation of forms to be used in the website in python
Javascript	ECMAScript 2022 (ES13)	Allows for dynamic website implementation
HTML	HTML5	Website template implementation
Bootstrap	5.3.0	Used for a more appealing visuals in a website
JQuery	3.6.0	Allows for easier use of javascript
Urllib.request	N/A	Request data from when the system uses api *
Pandas	1.3.3	Used to manipulate data
Reverse geocode	N/A	Used to find location given coordinate data*
Global land mask	N/A	Used to identify between land and ocean *
Codecs	N/A	Used for binary data encoding *
Matplotlib	3.4.3	Used for graphical visualisations
SKlearn	0.24.3	Used for implementing machine learning
Sqlite	3.36.0	Lite version of database implementation

Table 5

Chapter 4 Design

4.1 Design overview

The following shows a basic idea of my project and how the system works.

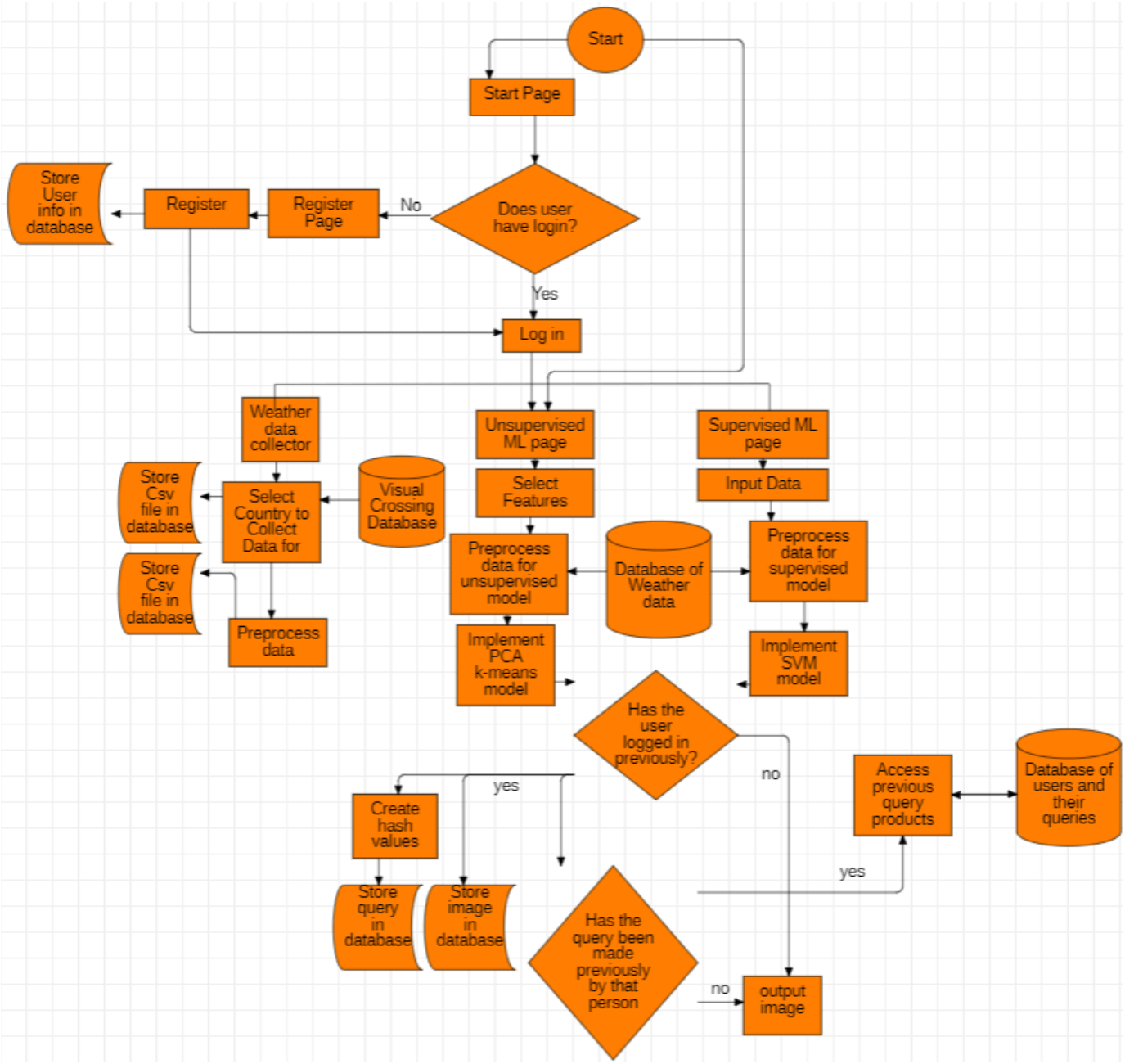


Figure 1 :Overview of Data Flow Diagram of System

4.2 Database Architecture / Entity Relationship Diagram

The following is an ER Diagram to represent the Database architecture of the system.

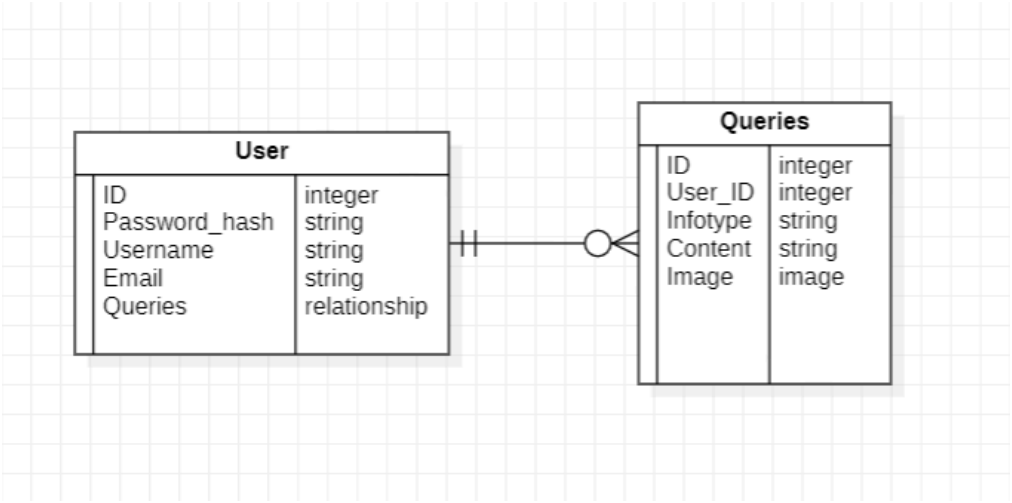


Figure 2: ER Diagram of Database architecture

4.3 Supervised ML Pipeline

The diagram below illustrates the Supervised ML of the system.

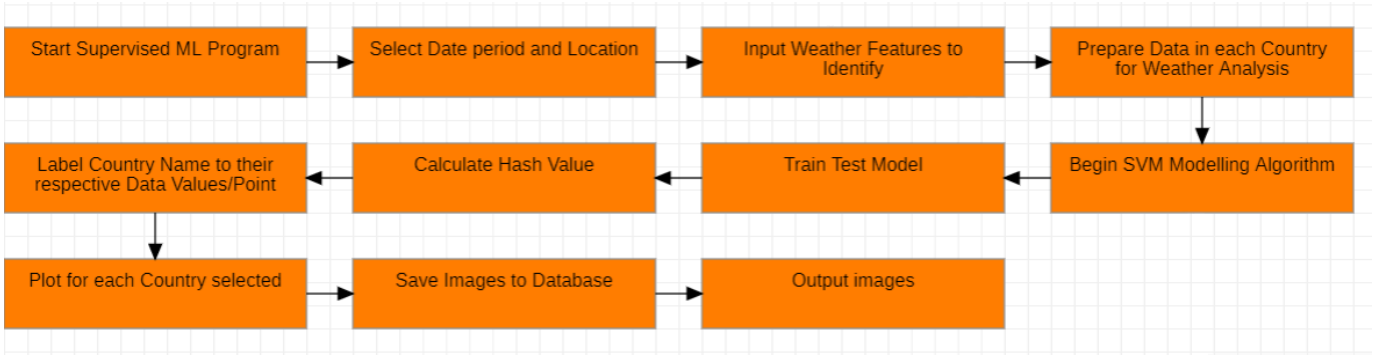


Figure 3: Supervised ML Pipeline Diagram

4.4 Unsupervised ML Pipeline

The diagram below illustrates the Unsupervised ML of the system.

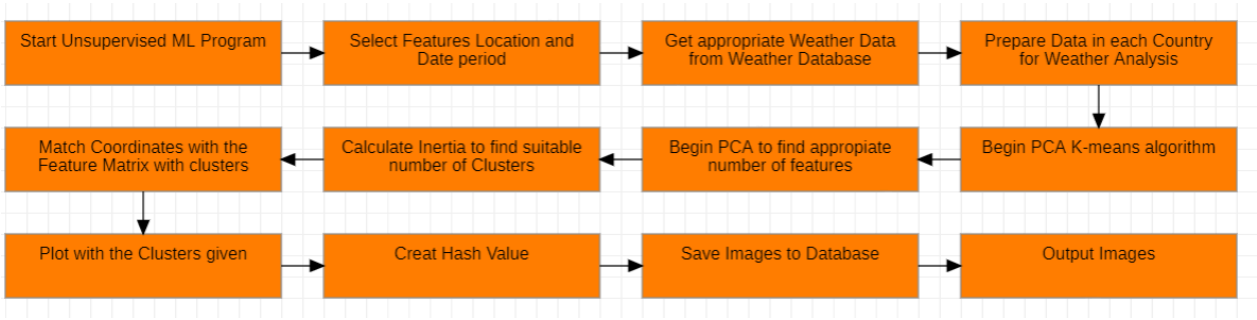


Figure 4: Unsupervised ML Pipeline Diagram

4.5 Data Collection Pipeline

The diagram below illustrates the Data Collection Pipeline of the system.

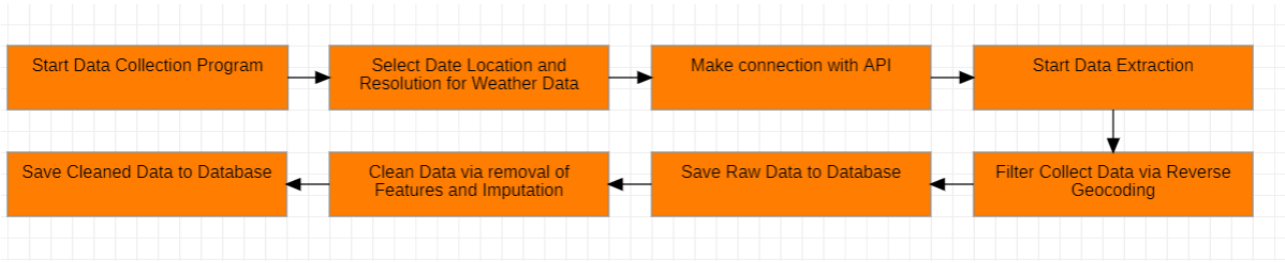


Figure 5: Data Collection Pipeline Diagram

4.6 UI (Interface Design)

The following shows the early user interface wireframes of the system .

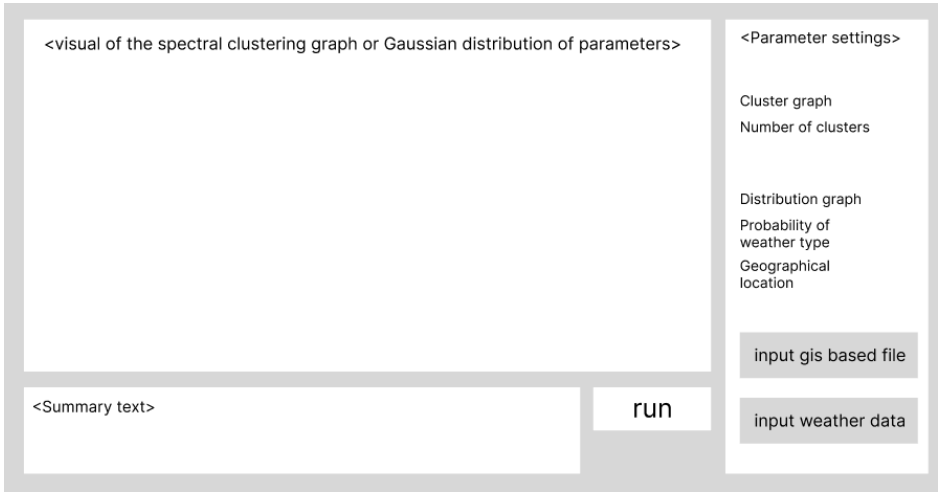


Figure 7: General UI Wireframe for both ML Webpages

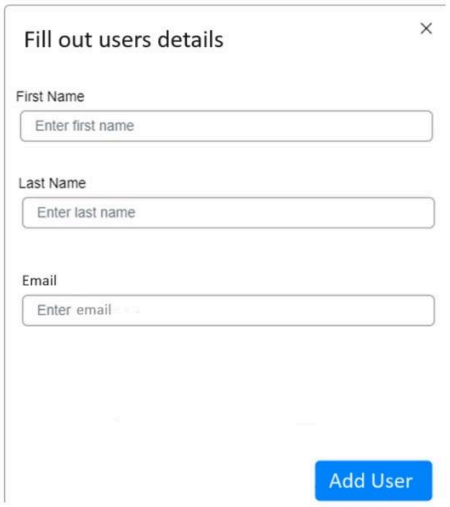


Figure 8: Early login/register UI wireframe

Figure 9: Early data collection UI web page wireframe

4.7 System Architecture

The following shows the system architecture and its interactions.

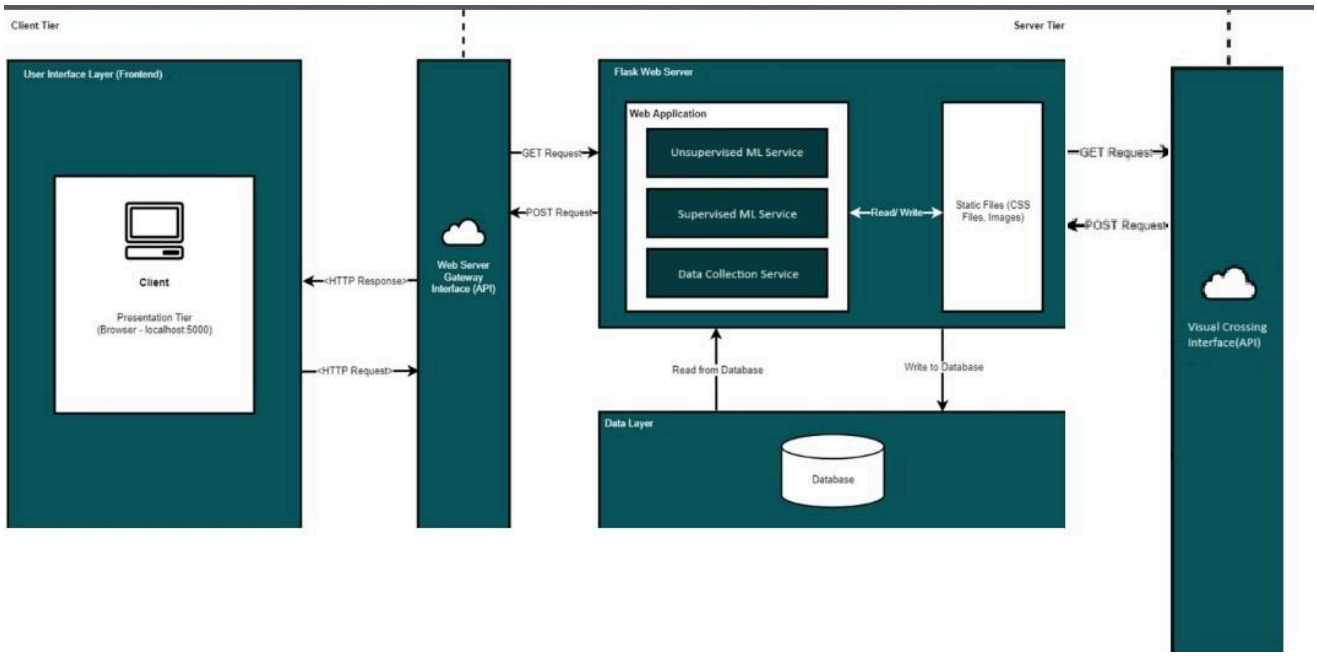


Figure 10: System architecture diagram

Chapter 5 Implementation

The following chapter shows the methodology I followed in order to satisfy and complete my project in accordance to the requirements along with explanation of the program and the code

5.1 Development process

My project was developed following the Agile Methodology. This makes sure that at each completion of a code section I would test the individual part along with all the other components as well as with each other until I was confident it had been completed. Each stage will be re-evaluated after each completed part. This makes sure that the program significantly lowers the chance of future errors or bugs as they would be addressed pre-emptively. The diagrams below illustrate the stages in the Agile Methodology and how testing is implemented.See figures below.

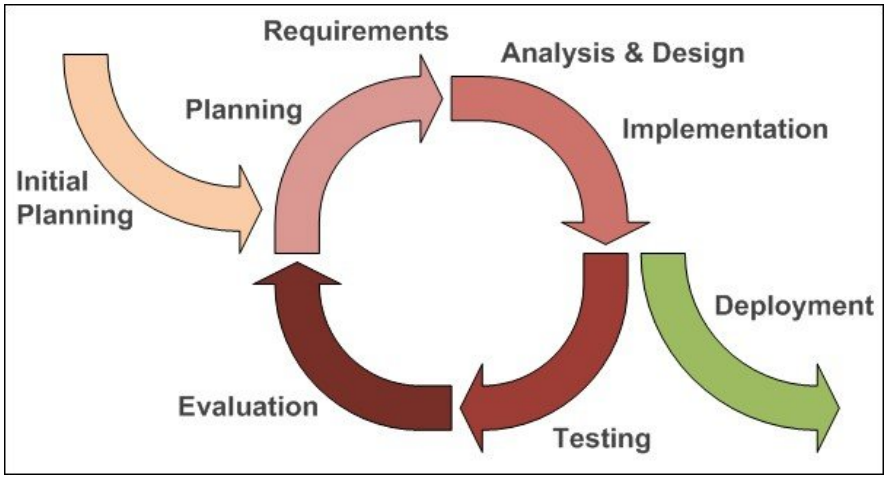


Figure 11: Agile Development Process

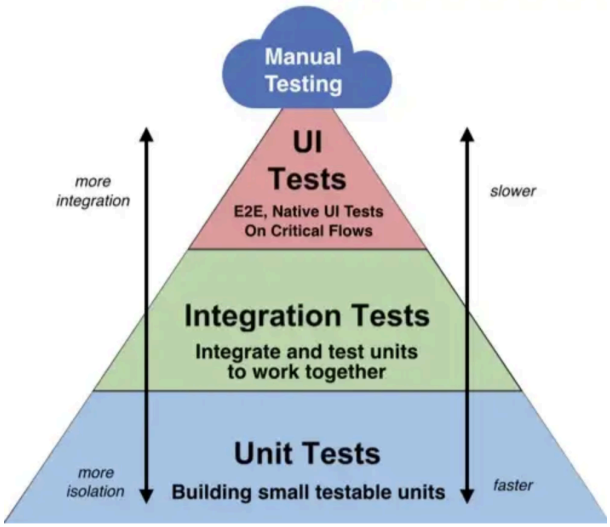


Figure 12: Software Testing Hierarchy

5.2 Languages and Frameworks

Various languages and frameworks were used to implement the system, each serving its particular purpose.

5.2.1 Front-end

The following languages and frameworks were used to implement the front-end of the system.

5.2.1.1 HyperText Markup Language (HTML)

HTML was employed to structure the web page and its content. It acted as the foundation, facilitating the display of information and data on the web page. A base.html file was created as a template to streamline the development process, eliminating redundancy across other pages.

5.2.1.2 Cascading Style Sheets (CSS)

CSS was used to style HTML elements, defining their appearance and layout on the web page. It enhanced the visual presentation by specifying font styles, sizes, colours, and overall page layout.

5.2.1.3 JavaScript (JS)

JS enabled dynamic content updates on the web page. It was utilised to handle user interactions, storing user inputs in local variables for future use. For instance, JS captured user selections from dropdown menus to dynamically update page options and the web pages themselves.

5.2.1.4 Bootstrap

Bootstrap, an open-source front-end framework, was leveraged to streamline website development. By integrating HTML, CSS, and JavaScript, Bootstrap provided pre-designed templates for various front-end components such as buttons, forms, and navigation bars. It facilitated the styling of tables, navigation bars, forms, and buttons within the system.

5.2.2 Back-end

The following languages and frameworks were employed to implement the back-end of the system:

5.2.2.1 Python

Python is a high-level interpreter based programming language that is very syntactically simple. A high-level language python abstracts many low-level details like memory management and hardware interaction, making it easier for programmers to focus on solving problems rather than dealing with technical intricacies. This characteristic, combined with its readable syntax, makes Python

an ideal choice for a wide range of applications, from web development and scripting to scientific computing and artificial intelligence .

5.2.2.2 Flask

Flask is a micro framework for Python, which makes simple developing beginner websites.the framework would consist of files and directories, reducing development time while ensuring security. The Flask debugger, WSGI, aided in efficient debugging and issue resolution .

5.2.2.3 Flask-login,Flask-sqlalchemy,Flask_bcrypt,Flask

The following are extensions of Flask for added functionality

- Flask-login provided user session management and authentication features, enhancing the security and usability of the system.
- Flask-sqlalchemy facilitated the interaction between Flask and SQLite, simplifying database operations within the Flask application.
- Flask_bcrypt enabled password hashing, ensuring the security of user credentials stored in the system.
- Flask-wtf simplified the creation of web forms in Flask applications, enhancing user interaction and data collection.

5.2.2.4 SQLite

SQLite, a file-based SQL database together with Python, was chosen for its simplicity. It stored data related to students, users, and attendance within the system.

5.2.2.5 JavaScript Libraries (JQuery)

JQuery was used to simplify JavaScript programming, enabling DOM manipulation and AJAX interactions within the system.

5.2.2.6 Urllib.request

Urllib.request facilitated HTTP request handling, enabling communication with external APIs and resources.

5.2.2.7 Pandas

Pandas, a data manipulation library, was used for data processing and analysis within the system.

5.2.2.8 Reverse geocode

Reverse geocode was employed to retrieve location data based on geographical coordinates, enriching the system's functionality.

5.2.2.9 Global land mask

Global land mask provided geographical data for spatial analysis and visualisation within the system.

5.2.2.10 Codecs

Codecs facilitated encoding and decoding of data, ensuring compatibility and data integrity within the system.

5.2.2.11 Matplotlib

Matplotlib was utilised for data visualisation, enabling the generation of plots and charts to enhance data interpretation within the system.

5.2.2.12 SKlearn

SKlearn, a machine learning library, was employed for various machine learning tasks such as classification and regression within the system.

5.3 UI implementation

The following is the basic framework for how my program is visualised and basic way in which the user would interact with the templates in the program

5.3.1 Templates

In a Flask framework, templates are key in separating the presentation layer/what you see (HTML code) from the application logic/things happening in the background (Python code). Here are my templates for my system.See figure 13 below.

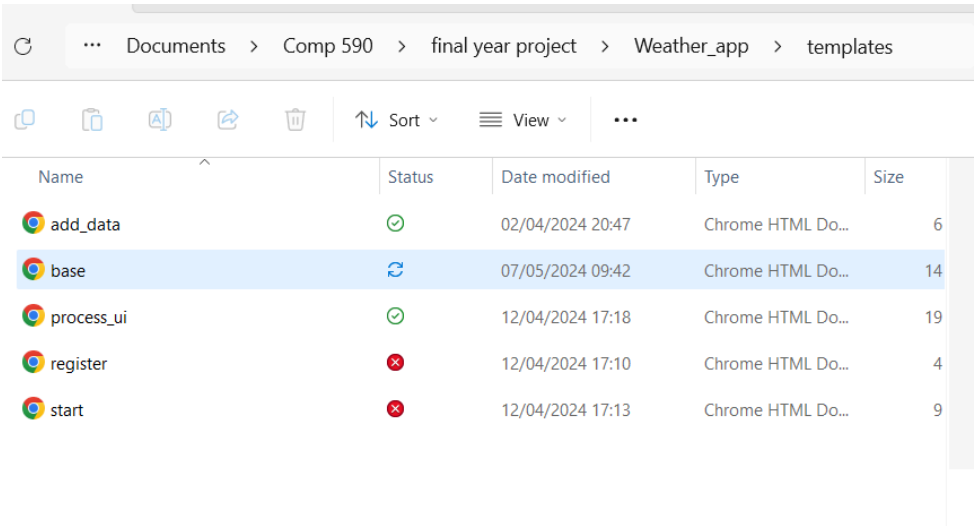


Figure 13: System Templates

5.3.1 add_data.html

-This file holds the framework for the data collectors page

5.3.2 base.html

-This file holds the framework for all functional web pages i.e supervised,unsupervised,data collection page

5.3.3 process_ui.hmtl

-Is the file representing both the supervised and unsupervised page depending on how the program is called

5.3.4 register.html

-Shows the register file and its framework

5.3.5 login.html

-Shows the login file and its framework

Each template creates a webpage for a specific functionality apart from base.html

5.4 User account creation

Many unit components (py files, html files and databases) work together to make up the system where each part adds its functionality. In this section, the application of practical key functionalities will be explained with the support of code snippets and snapshots of the system.

5.4.1 Registering

You can sign up to the system as a user. Users of this program are typically meteorologists or weather enthusiasts who would use the program to analyse their acquired weather data. To sign up the system requires the user to provide their email address, first name, and password. This information makes sure that there is a secure access to the weather analysis features and personalised user experience

(see later). Upon registration, users can use the functionalities of the system. See figure 14 below.

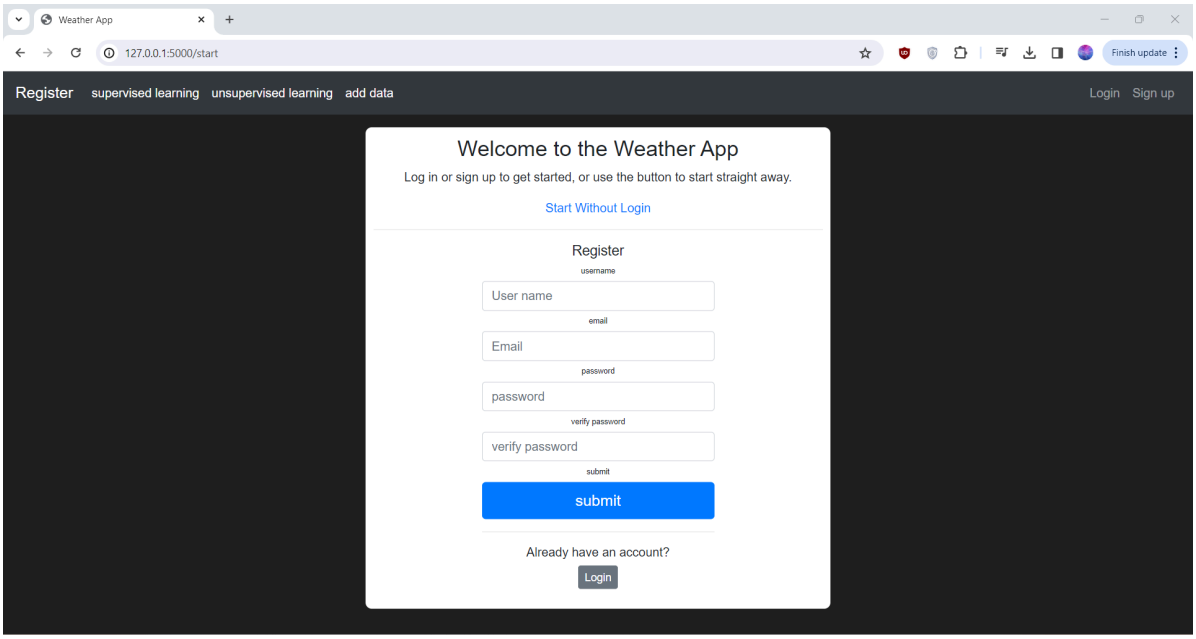


Figure 14: User Register Webpage

The user registration process for accessing the weather analysis program is allowed through the /start route in routes.py, where users input their details via a registration form. This form, managed by RegisterForm(), validates user inputs, and matches as to what is required for the form to be valid. If the form submission was successful, user details, such as username, email, and a hashed password for enhanced security, are stored in the system's database. Error handling mechanisms provide feedback to users in case of any issues during registration, guiding them through the process effectively. The rendered registration form, along with error messages when applicable, is presented to users. This approach to user registration.See figure 15 below.

```python
@app.route("/")
@app.route("/start", methods=['GET', 'POST'])
def start_page():
    print('register mode')
    print(current_user)
    form = RegisterForm()
    if form.validate_on_submit():
        user_to_create = User(username=form.username.data, email=form.email.data, password=form.password1.data)
        db.session.add(user_to_create)
        db.session.commit()
        login_user(user_to_create)
        flash(f'account created succesfully you are now logged in as: {user_to_create.username}')
        return redirect(url_for('unsupervised'))

    if form.errors != {}:
        for err_msg in form.errors.values():
            flash(f'there was an error in creating a user: {err_msg}')
    return render_template('start.html', form=form, show_register_form=True, show_login_form=False)
```

Figure 15: Register page code

Note: RegisterForm() will be explained more clearly in forms.py later from now

4.9.2 Login

If a user has just signed up or they have login details already, they can log in with those details. A valid email address and password must be entered so that the user can log in to the system. See figure 16 below.

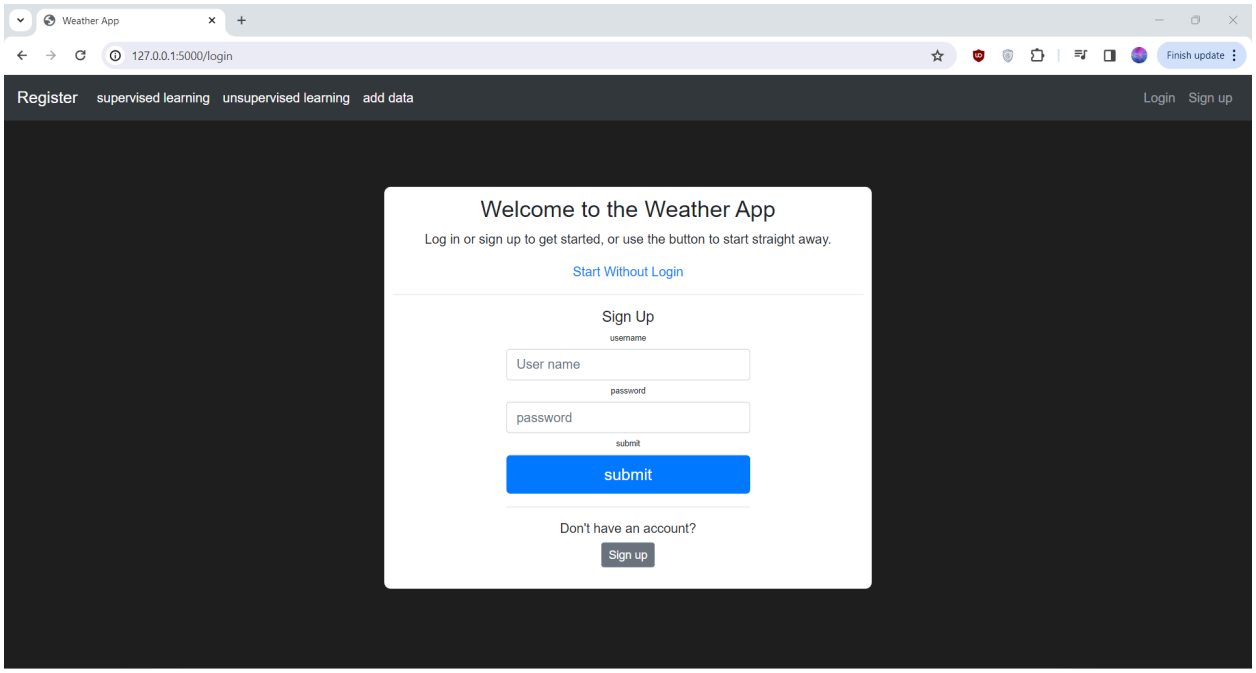


Figure 16: User Login Web Page

A POST request is carried out to get this information and then the user details are searched for in the database. Validation is applied to ensure if the user details are incorrect or if the user does not exist then they would not be logged in. Otherwise, the user is successfully logged in. See figure 17 below.

```
@app.route('/login',methods=['GET', 'POST'])
def login():
    print('login mode')
    print(current_user)
    form = LoginForm()
    if form.validate_on_submit():
        attempted_user = User.query.filter_by(username=form.username.data).first()

        if attempted_user and attempted_user.check_password_correction(
                attempted_password=form.password.data):
            login_user(attempted_user)
            flash(f'Logged in as: {attempted_user.username}',category='success')
            return redirect(url_for('unsupervised'))
        else:
            flash('username and password do not match please try again')
    return render_template('Start.html', show_login_form=True, show_register_form=False, form=form)
```

Figure 17: Login page code

Alternative to signing in one can immediately use the program just with no added functionality as described by Figure 1 before.

5.5 Data Collection

5.5.1 Data Collector Interface

To analyse the data you must acquire the data first via the Visual Crossings API service which has access to historical weather data, the data would be acquired through input of a country one wants to acquire data for then to input the time period they want to acquire the data for and the resolution of the data (which determines the detail of the output). See figure 18 below.

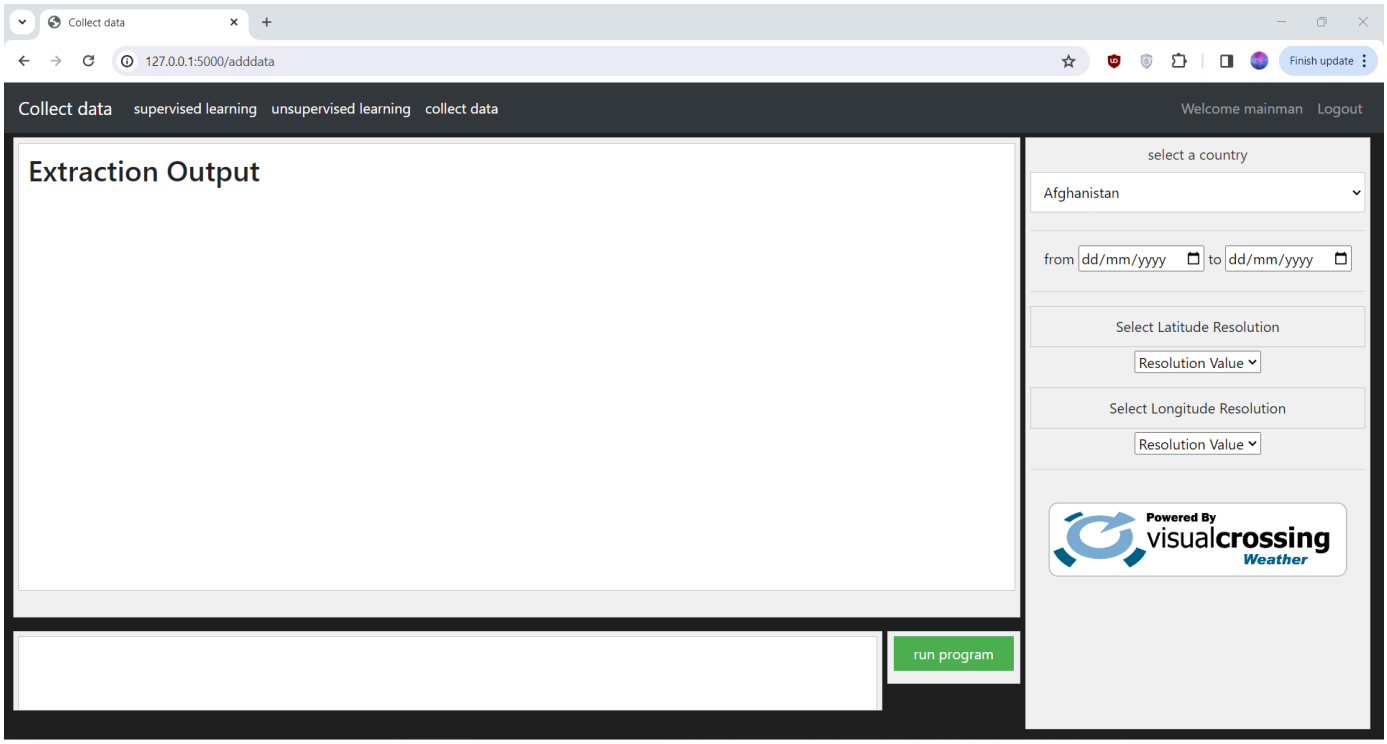
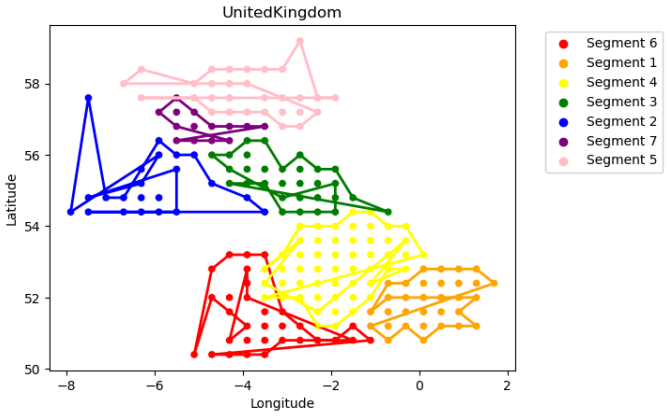


Figure 18: Data Extractor webpage

The resolution is important to ensure that the final image is detailed enough to capture the visual information but fast enough to be processed in an appropriate amount of time.See figure 18 and 19 below.



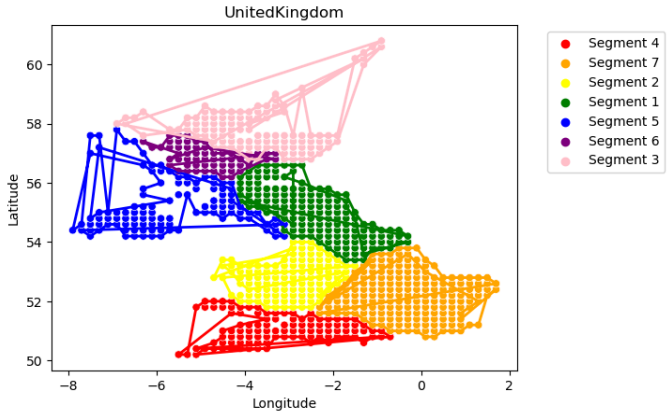


Figure 19 and 20: Lower and Higher resolution image of United Kingdom

Two folders are created, one containing the raw data saved to the raw_data directory collected from Visual Crossing and the other Acquired from cleaning the raw data and then saved into the cleaned_data directory. I've chosen for my program to work with the cleaned data due to being complete ,smaller and faster to analyse with.See figure 21 and 22 below.

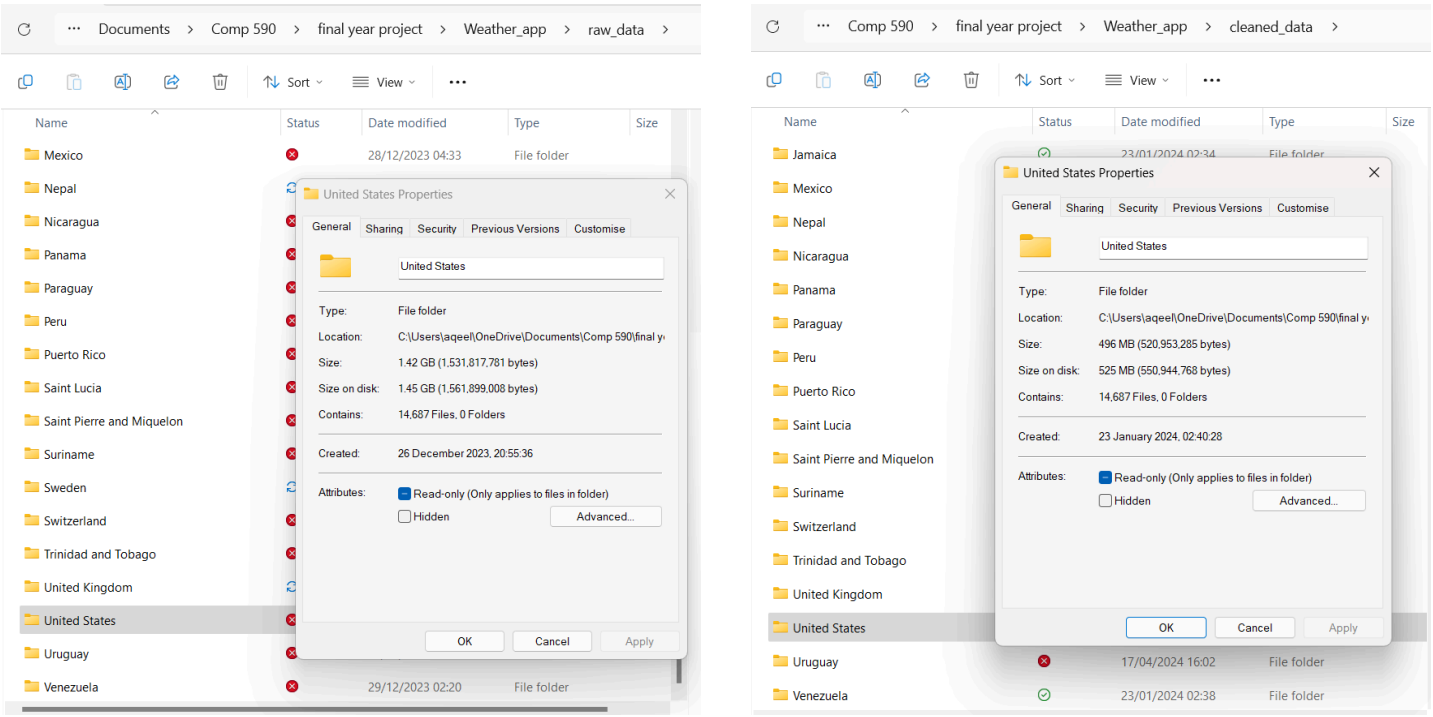


Figure 21 and 22: Raw and Cleaned Data with size comparison

5.5.2 Data Collecting Code

The following is the main data collecting code which is used to clean the data alongside acquire as part of pre-processing and extraction respectively.

5.5.2.1 data_cleaner.py

Data_cleaner.py works by taking a country folder name from raw_data and producing a processed equivalent to the directory cleaned_data, the code goes about going through the folder to then through each file inside the raw_data folder

and cleans the data via removal of values imputation column reduction and with other placeholder values for missing data (mean, window mean etc) dependent on if conditions are met data is extended/transformed so that the label contains more information which allows for better classification results

```
for file_name in os.listdir(input_country_path):
    input_file_path = os.path.join(input_country_path, file_name)

    # Check if the path is a file (not a subdirectory)
    if os.path.isfile(input_file_path):
        output_file_path = os.path.join(output_country_path, file_name)
        output.append('Cleaning coordinate data ' + str(file_name) +': '+ str(country) )
        clean_file = clean(input_file_path,file_name)
```

Figure 23: Code for going through a Country folders files iteratively

```
data['name'] = file_name
for index in range(len(data)):
    row = data.iloc[index]

    # append conditions based on numerical values
    if row['cloudcover'] < 20 and row['solarenergy'] > 19:
        row['conditions'] += ", Sunny"
    if row['windspeed'] > 20:
        row['conditions'] += ", Windy"
```

Figure 24: Code for label data Transformation/Extension

```
data.drop({'name','moonphase','feelslikemax','feelslikemin','dew',
           'feelslike','preciptype','sealevelpressure','visibility',
           'winddir','windgust','icon','stations','description',
           'solarradiation','severerisk'},axis=1,inplace=True)
```

Figure 25: Code for Feature Removal

```
    if selected_column.dtype == "float64":
        # Check if more than 80% of values are null
        if null_count / data.shape[0] > 0.8 :
            selected_column.fillna(0, inplace=True)
```

Figure 26: Code for data null value imputation

This code identifies columns with real number type values. If there are null values present but not to a significant extent (above 80%), then for the values where null values are found, it locates the surrounding 14 values above and below (over a 2-week period), computes the mean of them, and uses that to determine the value at that point (window mean). This process has been applied for all numerical data types. See the figures 27-29 below.

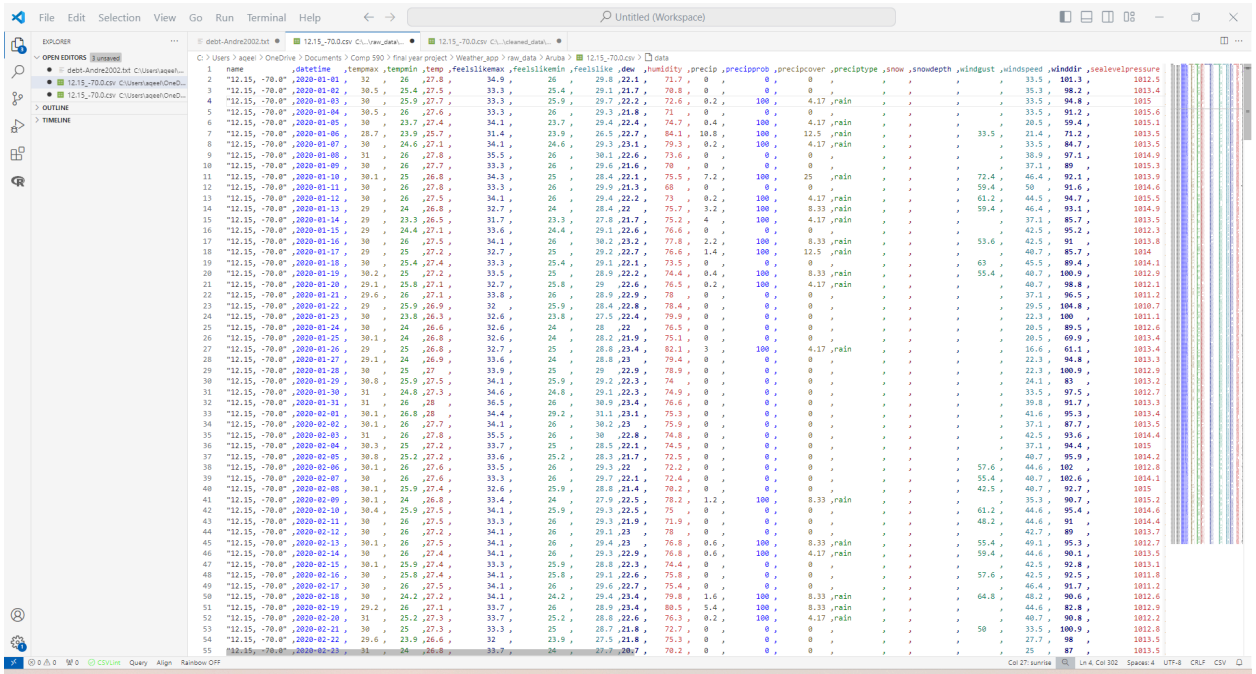
```
for column in data.columns:
    selected_column = data[column]
    null_count = selected_column.isnull().sum()
    window_size = 7
    if selected_column.dtype == "float64":
        # Check if more than 80% of values are null
        if null_count / data.shape[0] > 0.8 :
            selected_column.fillna(0, inplace=True)

        # Iterate through the DataFrame
        for missing_point in data[selected_column.isnull()].index:
            # Get the previous and next 7 values with boundary checks
            start_index = max(0, missing_point - window_size)
            end_index = min(data.shape[0], missing_point + window_size + 1)

            previous_values = selected_column.iloc[start_index:missing_point]
            next_values = selected_column.iloc[missing_point + 1:end_index]

            # Concatenate the previous and next values
            surrounding_values = pd.concat([previous_values, next_values])

            # Fill missing value with the mean of surrounding values
            data.at[missing_point, column] = surrounding_values.mean()

            surrounding_null_indices = surrounding_values.index[surrounding_values.isnull()]
            data.loc[surrounding_null_indices, column] = 0
```

Figure 27: Code for mean window imputation

This code computes the duration of sunlight by subtracting sunrise from sunset, then fills any missing values using forward-fill and backward-fill methods to ensure continuous data for analysis.

```
data['sunlight'] = (data['sunset'] - data['sunrise']).dt.total_seconds() / 60
data['sunlight'].fillna(method='ffill', inplace=True)
data['sunlight'].fillna(method='bfill', inplace=True)
```

Figure 28: Forwards/Backwards imputation



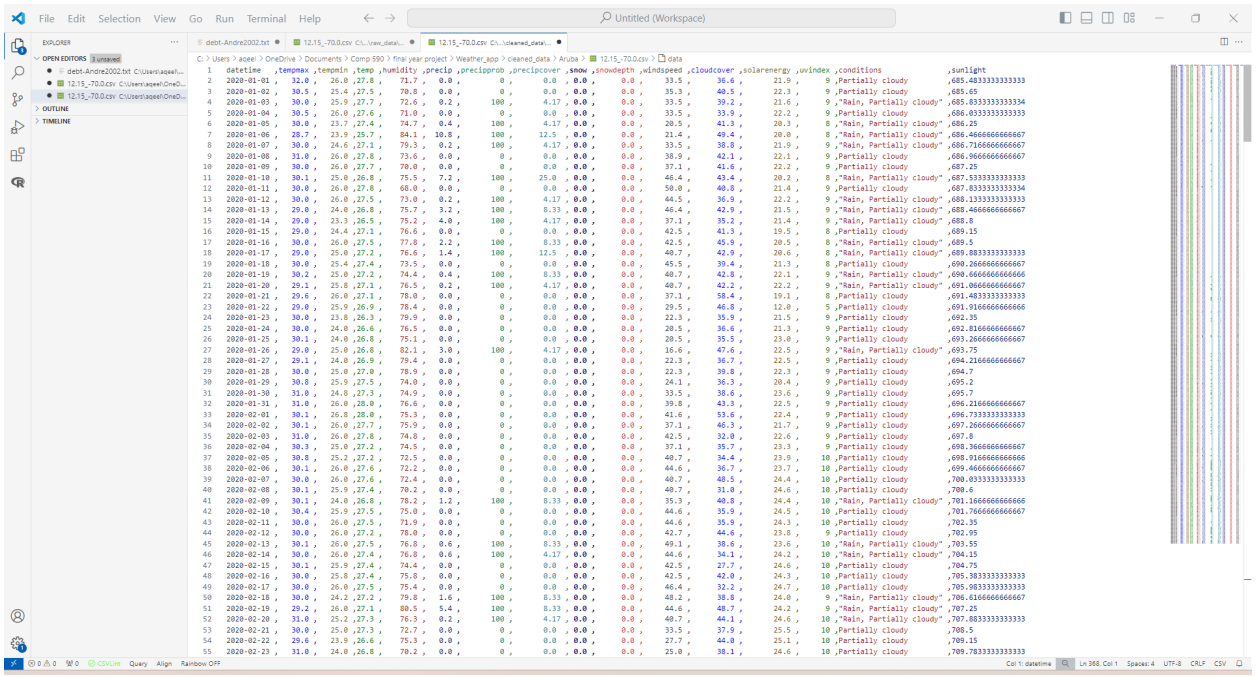


Figure 29: Raw and Cleaned data output

Note the null values in the above value and removed columns the data has been preprocessed

5.5.2.2 data_extractor.py

data_extractor.py Initialises by ensuring that previous data is written over if it exists for if you want different time periods or resolution for the country to be represented by then you can go about identifying coordinates for the country to acquire data from using a reverse geocode module (finds location given coordinates) using the coordinates found in the country the program will make an API request to Visual crossing for that data in which it will be converted to csv format for later usage. See figure 30 and 31 below.

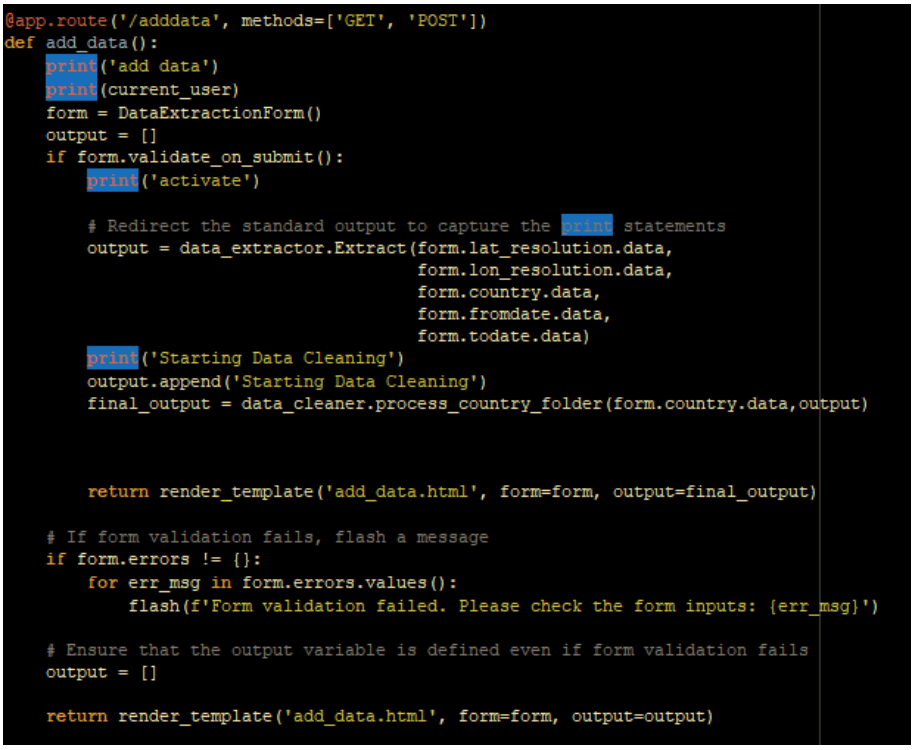
```
    if os.path.exists(directory_path):
        for file_name in os.listdir(directory_path):
            file_path = os.path.join(directory_path, file_name)
            os.remove(file_path)
        for file_name in os.listdir(directory_pathc):
            file_path = os.path.join(directory_pathc, file_name)
            os.remove(file_path)
        os.rmdir(directory_pathc)
        os.rmdir(directory_path)
        print('removed previous contents and will update with the next values')
```

Figure 30: Code for overwriting data

```
for lon in range(-1795, 1800, int(float(lon_res) * 10)):
    for lat in range(-900, 900, int(float(lat_res) * 10)):
        lat1 = float(lat / 10)
        lon1 = float(lon / 10)

        if globe.is_land(lat1, lon1):
            coordinates = [(lat1, lon1)]
            data = reverse_geocode.search(coordinates)
            country = data[0]['country']

            if (picked_country == country):
                print('Acquiring coordinate data ' + str(lat1) + "," + str(lon1) + " " + country)
                output.append('Acquiring coordinate data ' + str(lat1) + "," + str(lon1) + " " + country)


                if not os.path.exists(directory_path):
                    # Create the directory
                    os.makedirs(directory_path)
                    print(f"New directory '{directory_path}' has been added.")

                # File path to save
                file_name = f"{lat1}_{lon1}.csv"
                file_path = os.path.join(directory_path, file_name)
                print(file_path)
                # Check if file already exists
                if os.path.exists(file_path):
                    print(f"File {file_name} already exists. Skipping download.")
                    continue

                try:
                    ResultBytes = urllib.request.urlopen("https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/"
        + str(lat1) + "%2C%20" + str(lon1) + "/"+str(fromdate)+"/"+str(todate)+"?unitGroup=metric&include=days&key=KEYOMITTED&contentType=csv")
```

Figure 31: Code for Data Collection

These two programs are used together with routes.py function add_data to implement the functionality of collecting data with the visuals of the collect data webpage. See figure 32 below.



```
@app.route('/adddata', methods=['GET', 'POST'])
def add_data():
    print('add data')
    print(current_user)
    form = DataExtractionForm()
    output = []
    if form.validate_on_submit():
        print('activate')

        # Redirect the standard output to capture the print statements
        output = data_extractor.Extract(form.lat_resolution.data,
                                        form.lon_resolution.data,
                                        form.country.data,
                                        form.fromdate.data,
                                        form.todate.data)
        print('Starting Data Cleaning')
        output.append('Starting Data Cleaning')
        final_output = data_cleaner.process_country_folder(form.country.data,output)



        return render_template('add_data.html', form=form, output=final_output)

    # If form validation fails, flash a message
    if form.errors != {}:
        for err_msg in form.errors.values():
            flash(f'Form validation failed. Please check the form inputs: {err_msg}')

    # Ensure that the output variable is defined even if form validation fails
    output = []

    return render_template('add_data.html', form=form, output=output)
```

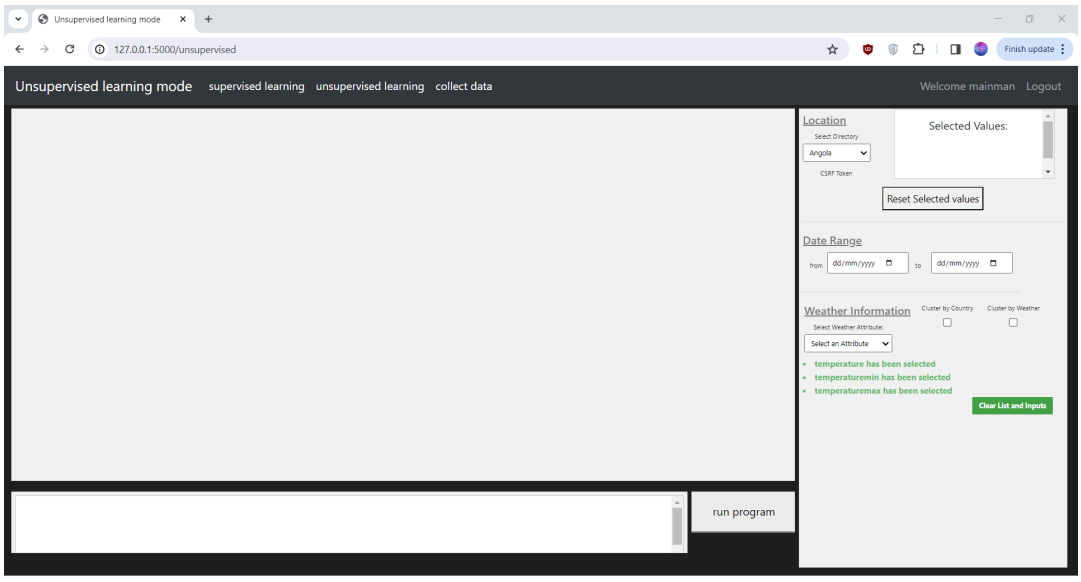
Figure 32: Data collection implementation

5.6 Unsupervised Learning

The following is the main code which is used to process data to produce a visual of clusters upon which can be analysed

5.6.1 Unsupervised Learning Interface

The user is accessing the Unsupervised machine learning webpage where they have the ability to cluster similar weather for the countries selected, this functionality is attained by firstly selecting a predetermined (by the user) list of countries to analyse the data from where they select a time period in which they want to analyse and the features to which they want to analyse the country based upon, once run an output of a graph being datapoints shaped in the country with the clusters in their respective colour will be shown as well as saved into the database. One has the option to select between cluster by country or cluster by weather where cluster by weather will cluster by which countries have similar weather patterns (determined by having the same colour in their respective images) while cluster by country will simply group by weather patterns found only in the country (note there is no difference between cluster by country or cluster by weather if only 1 country has been selected). See figures 33 and 34 below.



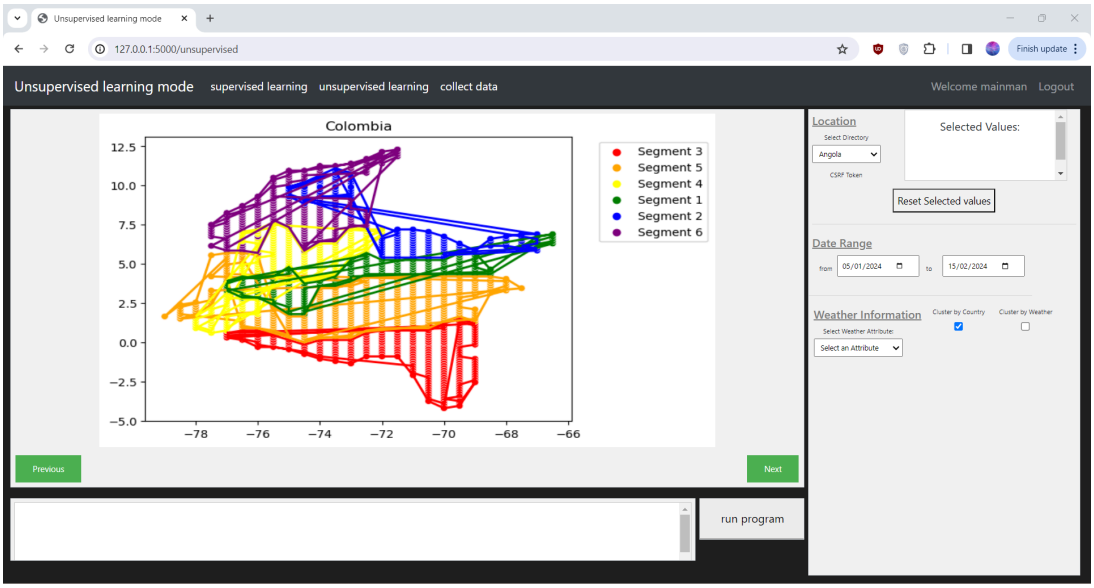
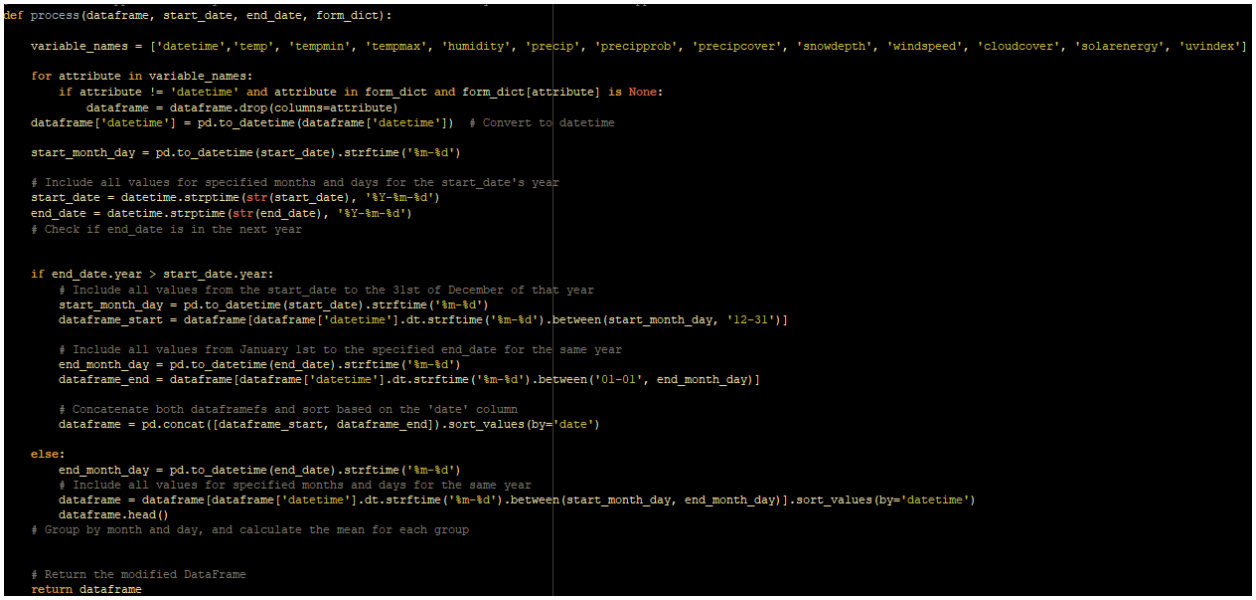


Figure 33 and 34: Interface and an Output of Unsupervised Model

5.5.2 Unsupervised Learning Code

The program implements a PCA K-means algorithm for clustering data. It begins by preparing the data for the model, iterating through files associated with each country. Each file, in dataFrame format, undergoes conversion and preprocessing to align with user-specified input requirements from the interface (the shape of the data is determined by what features and date period you selected earlier). This entails filtering rows to represent dates and columns to represent features. See figure 35 below



```
def process(dataframe, start_date, end_date, form_dict):

    variable_names = ['datetime','temp', 'tempmin', 'tempmax', 'humidity', 'precip', 'precipprob', 'precipcover', 'snowdepth', 'windspeed', 'cloudcover', 'solarenergy', 'uvindex']

    for attribute in variable_names:
        if attribute != 'datetime' and attribute in form_dict and form_dict[attribute] is None:
            dataframe = dataframe.drop(columns=attribute)
    dataframe['datetime'] = pd.to_datetime(dataframe['datetime'])  # Convert to datetime

    start_month_day = pd.to_datetime(start_date).strftime('%m-%d')

    # Include all values for specified months and days for the start_date's year
    start_date = datetime.strptime(str(start_date), '%Y-%m-%d')
    end_date = datetime.strptime(str(end_date), '%Y-%m-%d')
    # Check if end_date is in the next year


    if end_date.year > start_date.year:
        # Include all values from the start_date to the 31st of December of that year
        start_month_day = pd.to_datetime(start_date).strftime('%m-%d')
        dataframe_start = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between(start_month_day, '12-31')]

        # Include all values from January 1st to the specified end_date for the same year
        end_month_day = pd.to_datetime(end_date).strftime('%m-%d')
        dataframe_end = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between('01-01', end_month_day)]

        # Concatenate both dataframefs and sort based on the 'date' column
        dataframe = pd.concat([dataframe_start, dataframe_end]).sort_values(by='date')

    else:
        end_month_day = pd.to_datetime(end_date).strftime('%m-%d')
        # Include all values for specified months and days for the same year
        dataframe = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between(start_month_day, end_month_day)].sort_values(by='datetime')
        dataframe.head()
    # Group by month and day, and calculate the mean for each group


    # Return the modified DataFrame
    return dataframe
```

Figure 35: File Processing code

After having finished the process for each file in a country depending on whether the program was run for cluster by weather or country (cluster by weather will pass a dataframe holding all data for all countries to pca_ and by country will run pca_ for each country dataframe) the data frame will be passed to the ML algorithm section of the code for machine learning. See figure 36 below.

```
if form_dict['cluster_by_weather']:
    superdf = pd.concat([superdf, df1], axis=0, ignore_index=True)
    print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
    print(file_names)
    allnames = pd.concat([allnames, file_names], ignore_index=True)
    i += 1

    if i == len(countries):

        # This will run on all countries' dataframe and should be processed to output an image of all countries
        directory = pca_(superdf, allnames, form_dict, country)
        print('it was cluster by weather')
        return directory

elif form_dict['cluster_by_country']:
    # This will run on a single country's dataframe and should output an image of the country for all countries
    directory = pca_(df1, file_names, form_dict, country)
    i += 1
    if i == len(countries):
        print('it was cluster by country')
        return directory

return directory
```

Figure 36: Cluster by Weather and Cluster by Country Code

This hash function will be called and a hash will be made which will be explained in more detail later but will be used to name the directory where the images will be saved. See figure 37 below.

```python
def hash_it_u(form):
    # Extract and alphabetise location information
    variables_string = form['location']
    variables_list = variables_string.split(',')

    # Remove the last empty string (resulting from the trailing comma)
    if variables_list[-1] == '':
        variables_list.pop()

    # Alphabetize the list
    variables_list = sorted(variables_list)
    variables_string = ','.join(variables_list)
    # Convert date values to strings
    todate = str(form['todate'])
    fromdate = str(form['fromdate'])

    # Define variable names
    variable_names = ['temp', 'tempmin', 'tempmax', 'humidity', 'precip', 'precipprob', 'precipcover', 'snowdepth', 'windspeed', 'cloudcover

    # Remove values in the array where the corresponding value in the form is None
    filtered_variable_names = [attribute for attribute in variable_names if form.get(attribute) is not None]
    combined_info = f"{variables_string}-{todate}-{fromdate}-{','.join(filtered_variable_names)}"

    # Hash the combined information
    hash_value = hashlib.md5(combined_info.encode()).hexdigest()
    return hash_value
```

Figure 37: Unsupervised hash function code

The data will be standardised and passed to the machine learning algorithm to be processed acquiring the features for the data frame which represent the data to a suitable degree (85% or more) which will be analysed via k means clustering, before doing so, to determine the number of clusters for the program to cluster by determined by the inertia which will acquire WCSS this represents the elbow point (the number of clusters chosen now to use for the program) the program will then try to plot the data accordingly. See figure 38 below.

```
print('pca step 1')

scaler = StandardScaler()
std_df = scaler.fit_transform(data)
pca = PCA()
print(std_df.shape)
print(std_df.dtype)
print('here')
print(std_df)
print('here')
pca.fit(std_df)
print(data.shape)
cumulative_variance_ratio = pca.explained_variance_ratio_.cumsum()
component_number = 1

for i in range(0,len(cumulative_variance_ratio)):
    if cumulative_variance_ratio[i] > 0.85:
        component_number = i
        print(f"this will require {component_number} principle components for accuracy")
        break


pca = PCA(n_components=component_number)

pca.fit(std_df)

scores = pca.transform(std_df)
clusters = k_means_inertia(scores)

kmeans_pca = k_means(scores,clusters)

std_df = pd.DataFrame(std_df)

dspk = pd.concat([std_df,pd.DataFrame(scores)],axis=1)


component_strings = [f'{len(std_df.columns)+i}' for i in range(1, component_number + 1)]

dspk.columns.values[-component_number:]=component_strings
dspk = pd.concat([dspk,xy],axis=1)
print('pca part 2 step 4')
print(len(std_df))
```

Figure 38: PCA K-means Code

The following will plot the data applying the colouring to each respective cluster the data points will be based on the coordinates which will shape out the country will then have an outline be applied to the country based on find_outer_points which will draw the shape of the country then will save and plot the image. See figure 39 below.

```python
if form_dict['cluster_by_country']:
    longitude_column = dspk.iloc[:, -3].values
    latitude_column = dspk.iloc[:, -4].values
    print(latitude_column)
    print(longitude_column)
    print('just saying we got here with the coordinates')
    plt.figure()  # Create a new figure

    # Defines color palette
    custom_palette = ['red', 'orange', 'yellow', 'green', 'blue', 'purple', 'pink', 'black', 'gray', 'darkorange', 'maroon', 'deeppink', 'dimgray', 'oli

    # Creates a scatter plot for the data points
    sns.scatterplot(x=longitude_column, y=latitude_column, hue=dspk['Segment'], palette=custom_palette)
    # Find the outer data points for each cluster
    for i, cluster_label in enumerate(dspk['Segment'].unique()):
        cluster_data = dspk[dspk['Segment'] == cluster_label][['Longitude', 'Latitude']]
        outer_points = find_outer_points(cluster_data)

        # Create a polygon from the outer points and draw it around the cluster
        polygon = patches.Polygon(outer_points, edgecolor=custom_palette[i], linewidth=2, facecolor='none')
        plt.gca().add_patch(polygon)
    print('polygon clusters have just been made should be on it now')

    directory = f'Weather_app/static/images/unsupervised/cluster_by_country/{form_hash}'

    os.makedirs(directory, exist_ok=True)  # Ensure the directory exists before specifying file_path
    print('just made a new directory ')
    print(directory)
    s_country = sanitize_filename(country)
    image = f'{s_country}_image.png'
    print(f'should be cool {image}')
    file_path = os.path.join(directory, image)
    plt.title(s_country)

    # Place the legend outside the graph
    plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1))

    # Now, create the scatter plot with outer shapes and save the figure
    plt.savefig(file_path, bbox_inches='tight')  # Use bbox_inches='tight' to include the legend in the saved image
    print('image has finally been saved well as it should be')
    return directory
```

Figure 39: Plotting Code

5.5.3 Hashing algorithm

A simple extension on the hashlib library for my project's functionality the program takes into consideration the user and user inputs for defining the hash this will later be used so that the only user can only access the data if he re-inputs the same query.See figure 40 below .

```python
def hash_it_u(form):
    # Extract and alphabetise location information
    variables_string = form['location']
    variables_list = variables_string.split(',')

    # Remove the last empty string (resulting from the trailing comma)
    if variables_list[-1] == '':
        variables_list.pop()

    # Alphabetize the list
    variables_list = sorted(variables_list)
    variables_string = ','.join(variables_list)
    # Convert date values to strings
    todate = str(form['todate'])
    fromdate = str(form['fromdate'])

    # Define variable names
    variable_names = ['temp', 'tempmin', 'tempmax', 'humidity', 'precip', 'precipprob', 'precipcover', 'snowdepth', 'windspeed', 'cloudcov

    # Remove values in the array where the corresponding value in the form is None
    filtered_variable_names = [attribute for attribute in variable_names if form.get(attribute) is not None]
    combined_info = f"{variables_string}-{todate}-{fromdate}-{','.join(filtered_variable_names)}"

    # Hash the combined information
    hash_value = hashlib.md5(combined_info.encode()).hexdigest()
    return hash_value
```

Figure 40: Hash Function

The unsupervised function in the Weather_app Flask application organises and directs the unsupervised clustering process, guiding users through data analysis tasks. It handles both GET and POST requests, managing user interactions with forms for location selection and data input. Within this function, the unsupervised clustering algorithm, specifically PCA K-means, is executed through the pca_k_means.cluster() function. This algorithm partitions data into distinct clusters, returning a directory where the output of the clustering process is stored. Upon receiving user inputs and executing the clustering algorithm, the function evaluates whether the user is logged in or not. For authenticated users, it checks if a previous query matches the current input to avoid redundant computations. If a matching query exists, the function retrieves the stored images associated with that query. If no match is found, the function creates a new query, stores the output images, and displays them to the user. For anonymous users, the clustering process proceeds similarly, generating and displaying output images without associating them with a user.Furthermore, the image_array function facilitates the retrieval of image paths from the specified directory, enabling the display of clustered data images to users within the application interface. Overall, the unsupervised function ensures seamless execution of unsupervised clustering tasks, enhancing user experience and facilitating data-driven insights.. See figure 41 below.

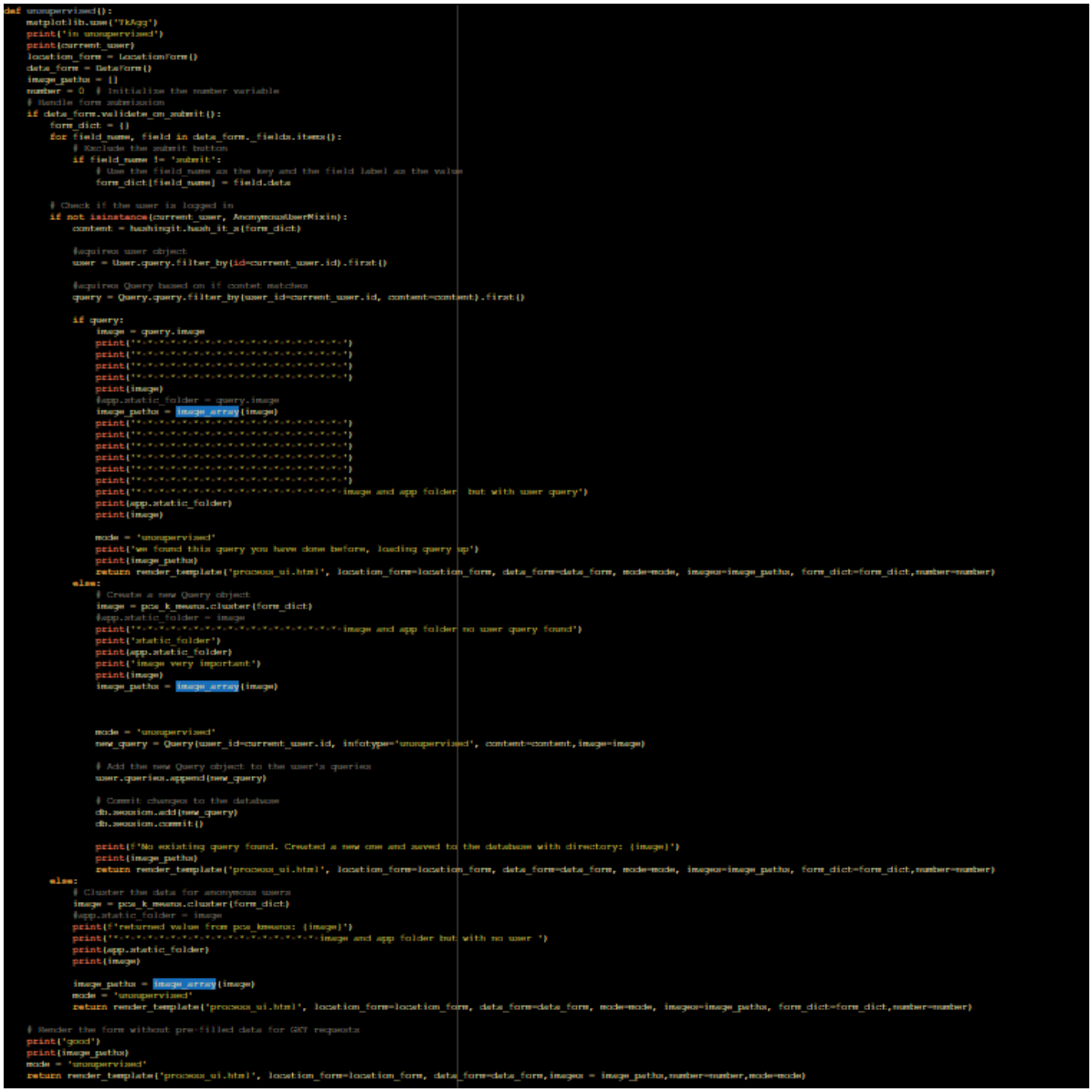


Figure 41: Route url for Unsupervised web page

5.7 Supervised Learning

5.7.1 Supervised Learning Interface

This interface is similar to that of the unsupervised learning algorithm however this would take in numerical inputs of integer or real numbers into the form as opposed to just selecting features, this data would be used to go about finding/identifying the label which the machine would try to predict what the data label is after the

training when the model tests itself it will now try go about matching the label found from the data to the input and for each coordinate will find how much each part of the country matches your queries label as a measure of similarity; this visual is then saved and outputted. See figures 42-47 below.

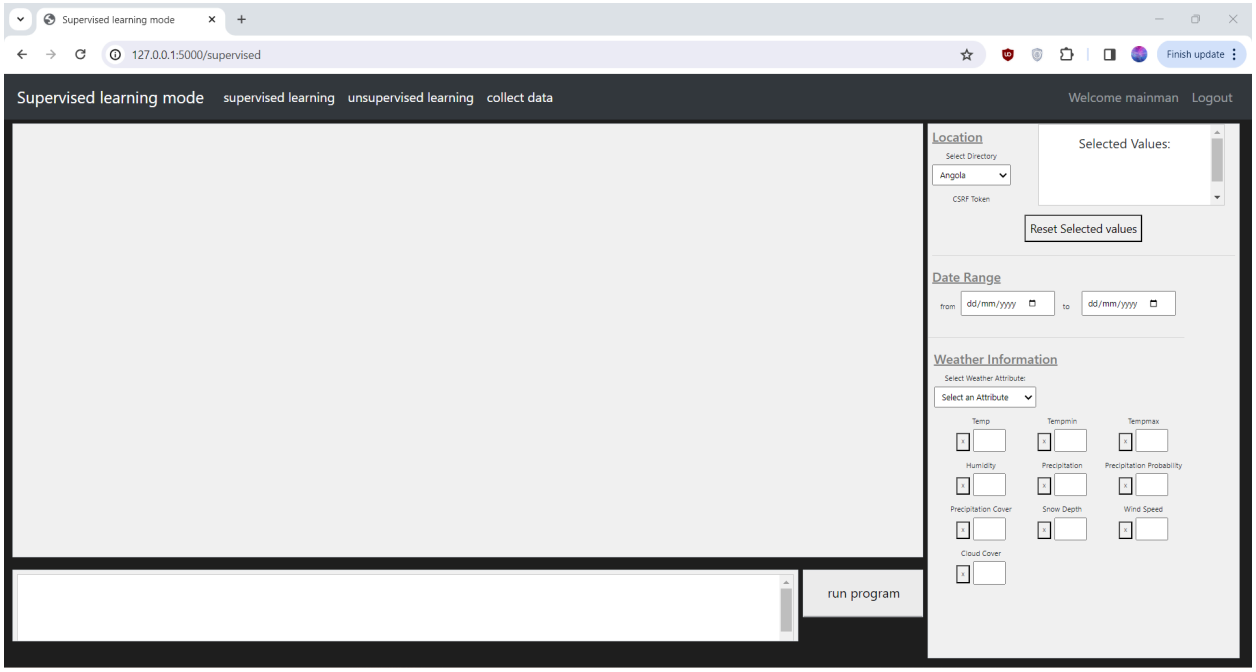
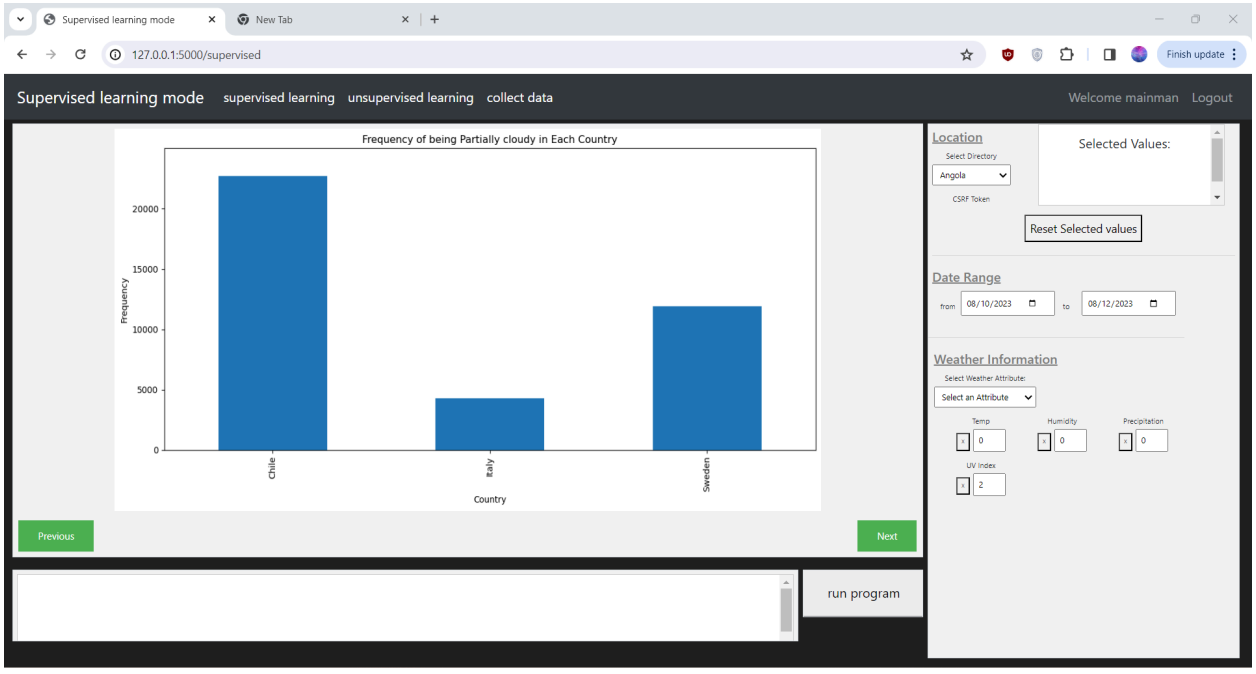
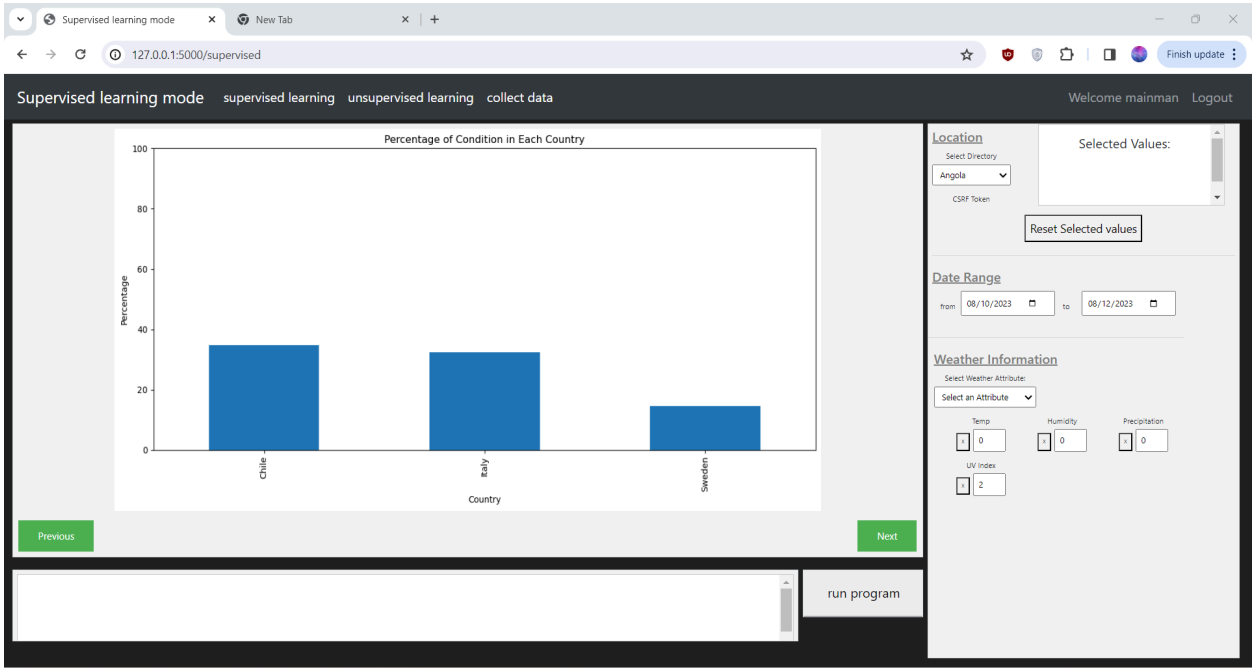
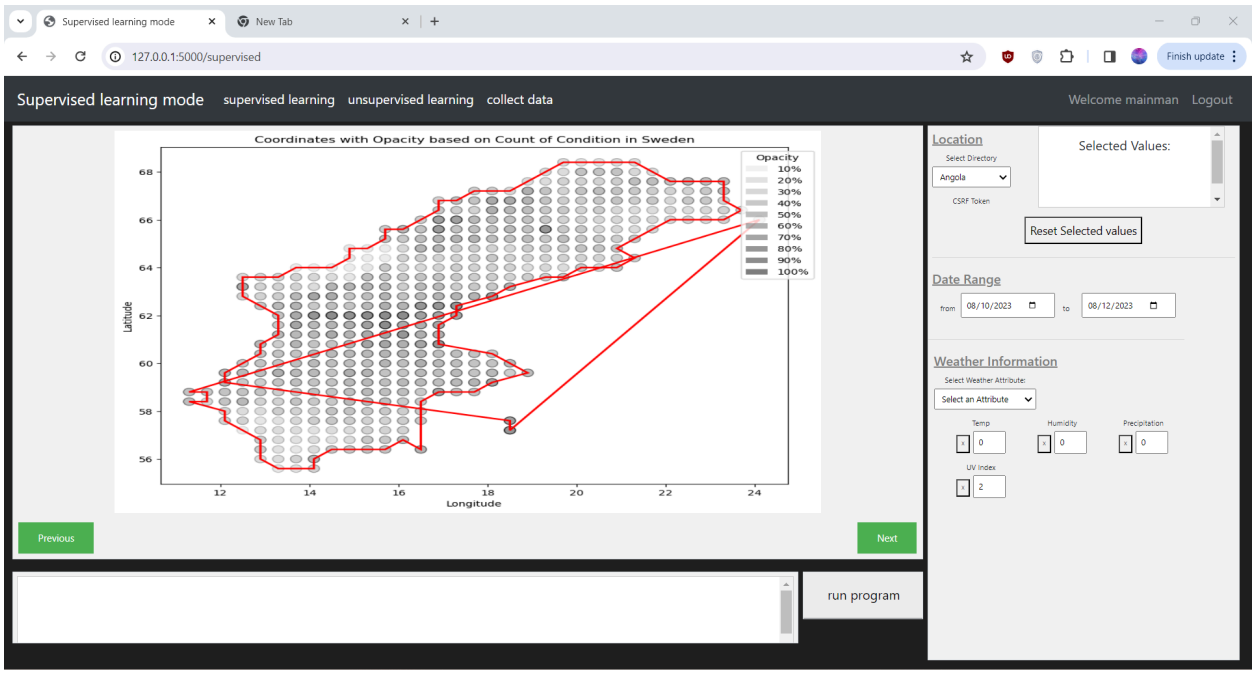


Figure 42: Supervised Model Interface




Supervised learning mode
New Tab
127.0.0.1:5000/supervised
Finish update
Supervised learning mode
supervised learning
unsupervised learning
collect data
Welcome mainman
Logout
Percentage of Condition in Each Country
Percentage
Country
Chile
Italy
Sweden
Previous
Next
run program
Location
Select Directory
Angola
CSRF Token
Selected Values:
Reset Selected values
Date Range
from
08/10/2023
to
08/12/2023
Weather Information
Select Weather Attribute:
Select an Attribute
Temp
Humidity
Precipitation
UV Index


Supervised learning mode
New Tab
127.0.0.1:5000/supervised
Finish update
Supervised learning mode
supervised learning
unsupervised learning
collect data
Welcome mainman
Logout
Coordinates with Opacity based on Count of Condition in Sweden
Opacity
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Latitude
Longitude
Previous
Next
run program
Location
Select Directory
Angola
CSRF Token
Selected Values:
Reset Selected values
Date Range
from
08/10/2023
to
08/12/2023
Weather Information
Select Weather Attribute:
Select an Attribute
Temp
Humidity
Precipitation
UV Index

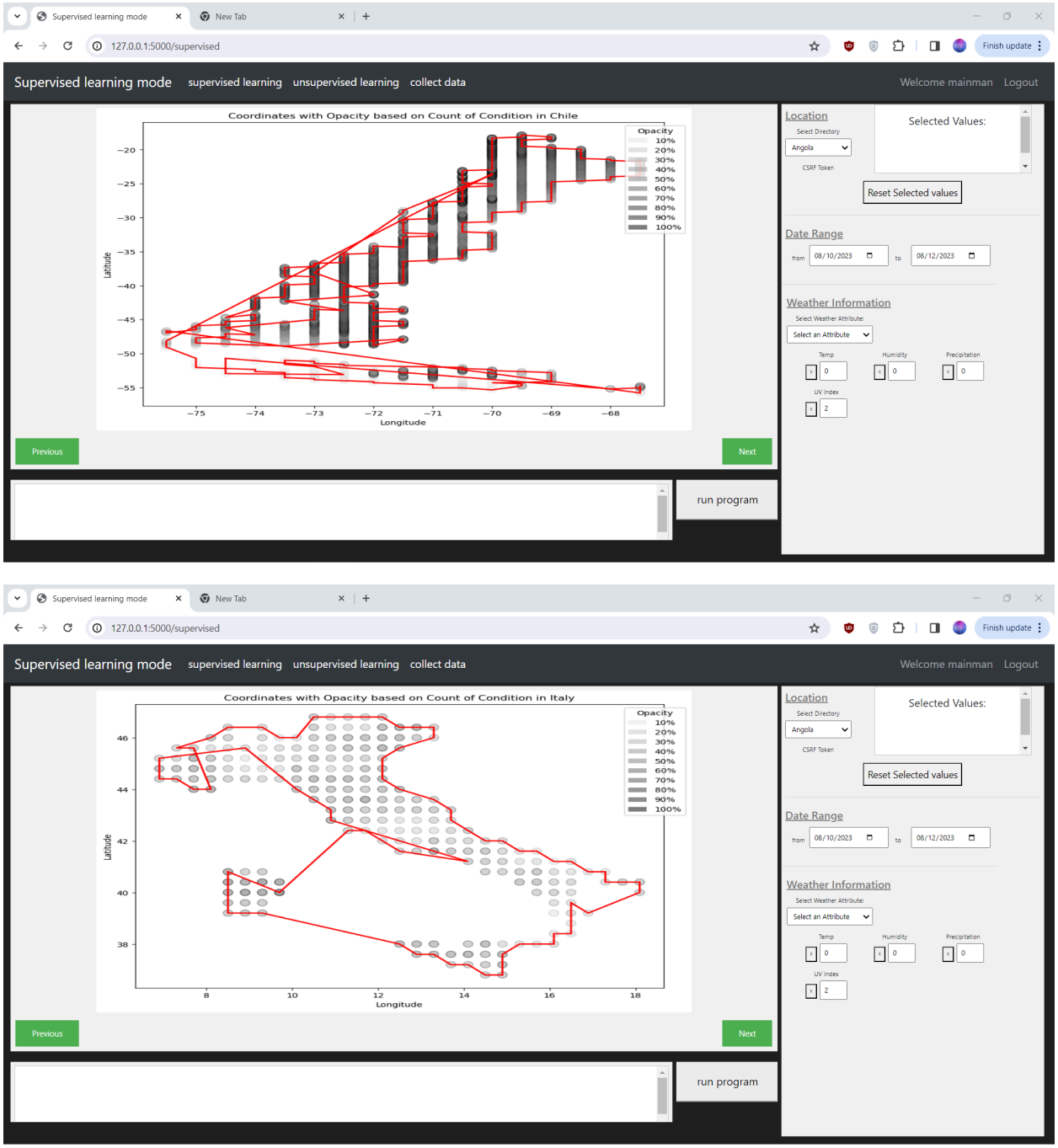


Figure 43 - 47 :Supervised Model Outputs

5.7.2 Supervised Learning Code

The program implements a Support Vector Machine (SVM) algorithm for clustering data. It begins by preprocessing the data, iterating through files

associated with each country similar to the unsupervised algorithm in that it takes user inputs and uses similar transformations however this program is formatted to how the supervised model would be used. Each file, in dataframe format, undergoes conversion and preprocessing to align with user-specified input requirements from the interface. This follows to filter rows to represent dates and columns to represent features. See figure 48 below.

```
def process(dataframe, start_date, end_date, form_dict):
    # Names of all the Features in the data
    variable_names = ['datetime','temp', 'tempmin', 'tempmax', 'humidity', 'precip', 'precipprob', 'precipcover', 'snowdepth', 'windspeed', 'cloudcover', 'solarenergy', 'uvindex']

    # Loop Through each feature will select every feature which has a null value and remove from dataframe
    for attribute in variable_names:
        if attribute != 'datetime' and attribute in form_dict and form_dict[attribute] is None:
            dataframe = dataframe.drop(columns=attribute)

    dataframe['datetime'] = pd.to_datetime(dataframe['datetime'])  # Convert to datetime


    # consider the date only by month and year (key for analysis of data)
    start_month_day = pd.to_datetime(start_date).strftime('%m-%d')

    # Include all values for specified months and days for the start_date's year
    start_date = datetime.strptime(str(start_date), '%Y-%m-%d')
    end_date = datetime.strptime(str(end_date), '%Y-%m-%d')
    # Check if end_date is in the next year


    if end_date.year > start_date.year:
        # Include all values from the start_date to the 31st of December of that year
        start_month_day = pd.to_datetime(start_date).strftime('%m-%d')
        dataframe_start = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between(start_month_day, '12-31')]

        # Include all values from January 1st to the specified end_date for the same year
        end_month_day = pd.to_datetime(end_date).strftime('%m-%d')
        dataframe_end = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between('01-01', end_month_day)]

        # Concatenate both dataframefs and sort based on the 'date' column
        dataframe = pd.concat([dataframe_start, dataframe_end]).sort_values(by='date')

    else:
        end_month_day = pd.to_datetime(end_date).strftime('%m-%d')
        # Include all values for specified months and days for the same year
        dataframe = dataframe[dataframe['datetime'].dt.strftime('%m-%d').between(start_month_day, end_month_day)].sort_values(by='datetime')

    # Group by month and day, and calculate the mean for each group


    # Return the modified DataFrame
    # Data frame will now be a matrix wiht where the row s are only months and days between the time period selected
    return dataframe
```

Figure 48: Supervised File Processing code

After having finished the pre-processing for each file in a country the program will pass a dataframe holding all data for all countries and begin to run the SVM model The data will be standardised and passed to the machine learning algorithm to be processed applying the features for the data frame which represent the data to a suitable degree which will be analysed via k means clustering before I do so to determine the number of clusters for the program to cluster by determine by the inertia which will acquire WCSS that represent the elbow point (the number of clusters for the program) the program will then try to plot the data accordingly. See figure 49 below.

```
X = df.drop(columns=['datetime','snow','conditions','sunlight'])
print('this is x')
print(X)


y = df['conditions']

# Split the data into training and testing sets (70-30 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=92)

# Initialize and train the Support Vector Classifier (SVM) model
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = svm_model.predict(X_test)

# Print the predicted labels
condition = svm_model.predict(df_row)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy of SVM model:", accuracy)
```

Figure 49: SVM Model Code

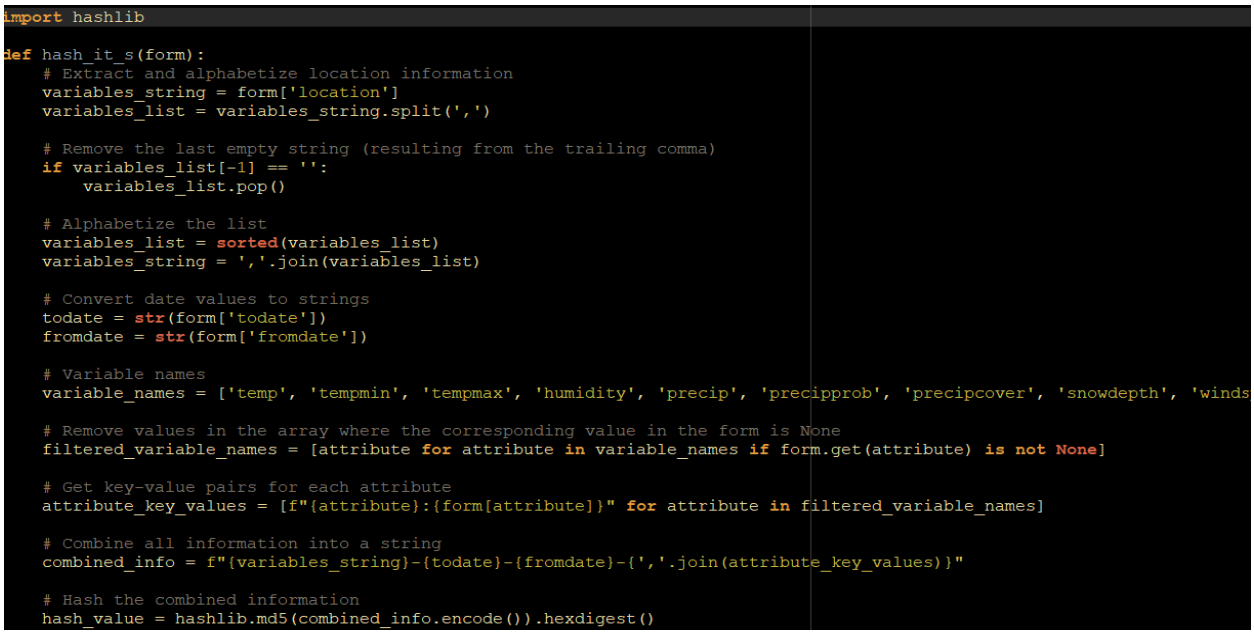
Creates as output three types of graphs one showing the percentage match the country has with the input the second shows the frequency or the total sum of days for each data point in the country has had the weather I inputted and the final thing is the visualisations of each country graph matching the weather the user input into the interface.. See figure 50 below.

```
if form_dict['cluster_by_country']:
    longitude_column = dspk.iloc[:, -3].values
    latitude_column = dspk.iloc[:, -4].values
    print(latitude_column)
    print(longitude_column)
    print('just saying we got here with the coordinates')
    plt.figure()  # Create a new figure

    # Defines color palette
    custom_palette = ['red', 'orange', 'yellow', 'green', 'blue', 'purple', 'pink', 'black', 'gray', 'darkorange', 'maroon', 'deeppink', 'dimgray', 'oli

    # Creates a scatter plot for the data points
    sns.scatterplot(x=longitude_column, y=latitude_column, hue=dspk['Segment'], palette=custom_palette)
    # Find the outer data points for each cluster
    for i, cluster_label in enumerate(dspk['Segment'].unique()):
        cluster_data = dspk[dspk['Segment'] == cluster_label][['Longitude', 'Latitude']]
        outer_points = find_outer_points(cluster_data)

        # Create a polygon from the outer points and draw it around the cluster
        polygon = patches.Polygon(outer_points, edgecolor=custom_palette[i], linewidth=2, facecolor='none')
        plt.gca().add_patch(polygon)
    print('polygon clusters have just been made should be on it now')

    directory = f'Weather_app/static/images/unsupervised/cluster_by_country/{form_hash}'

    os.makedirs(directory, exist_ok=True)  # Ensure the directory exists before specifying file_path
    print('just made a new directory ')
    print(directory)
    s_country = sanitize_filename(country)
    image = f'{s_country}_image.png'
    print(f'should be cool {image}')
    file_path = os.path.join(directory, image)
    plt.title(s_country)

    # Place the legend outside the graph
    plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1))

    # Now, create the scatter plot with outer shapes and save the figure
    plt.savefig(file_path, bbox_inches='tight')  # Use bbox_inches='tight' to include the legend in the saved image
    print('image has finally been saved well as it should be')
    return directory
```

Figure 50: Plotting Code

5.7.3 Hashing algorithm

This function is practically exactly the same as the unsupervised function. The program takes into consideration all user and user input values (not just the features like unsupervised learning) for defining the hash that will later be used so that the only user can only access the data if he re inputs the same query. See figure 51 below.



```
import hashlib

def hash_it_s(form):
    # Extract and alphabetize location information
    variables_string = form['location']
    variables_list = variables_string.split(',')

    # Remove the last empty string (resulting from the trailing comma)
    if variables_list[-1] == '':
        variables_list.pop()

    # Alphabetize the list
    variables_list = sorted(variables_list)
    variables_string = ','.join(variables_list)

    # Convert date values to strings
    todate = str(form['todate'])
    fromdate = str(form['fromdate'])

    # Variable names
    variable_names = ['temp', 'tempmin', 'tempmax', 'humidity', 'precip', 'precipprob', 'precipcover', 'snowdepth', 'winds

    # Remove values in the array where the corresponding value in the form is None
    filtered_variable_names = [attribute for attribute in variable_names if form.get(attribute) is not None]

    # Get key-value pairs for each attribute
    attribute_key_values = [f"{attribute}:{form[attribute]}" for attribute in filtered_variable_names]

    # Combine all information into a string
    combined_info = f"{variables_string}-{todate}-{fromdate}-{','.join(attribute_key_values)}"

    # Hash the combined information
    hash_value = hashlib.md5(combined_info.encode()).hexdigest()
```

Figure 51: Hash Function

The unsupervised function in the Weather_app Flask application manages the unsupervised clustering process, It handles both GET and POST requests, managing user interactions with forms for location selection and data input. pca_k_means.cluster() function from PCA K-means is run. This algorithm splits data into distinct clusters on a graph to then save then returning a directory where the output of the clustering process/graph was stored.Will check fo if the user is logged in or not, then for authenticated users, it checks if a previous query matches the current query input to avoid wasteful analytics. If a matching query exists, the function retrieves the stored images associated with that query and outputs to the screen. If no match is found, the function creates a new query, stores the output

images, and displays them to the user. For anonymous users, the clustering process proceeds similarly, generating and displaying output images without associating them with a user. The image_array function formats the directory of the image from the specified directory to be able to be seen on the page, allowing the display of clustered data images to be seen on the application interface. Overall, the unsupervised function.. See figure 52 below.

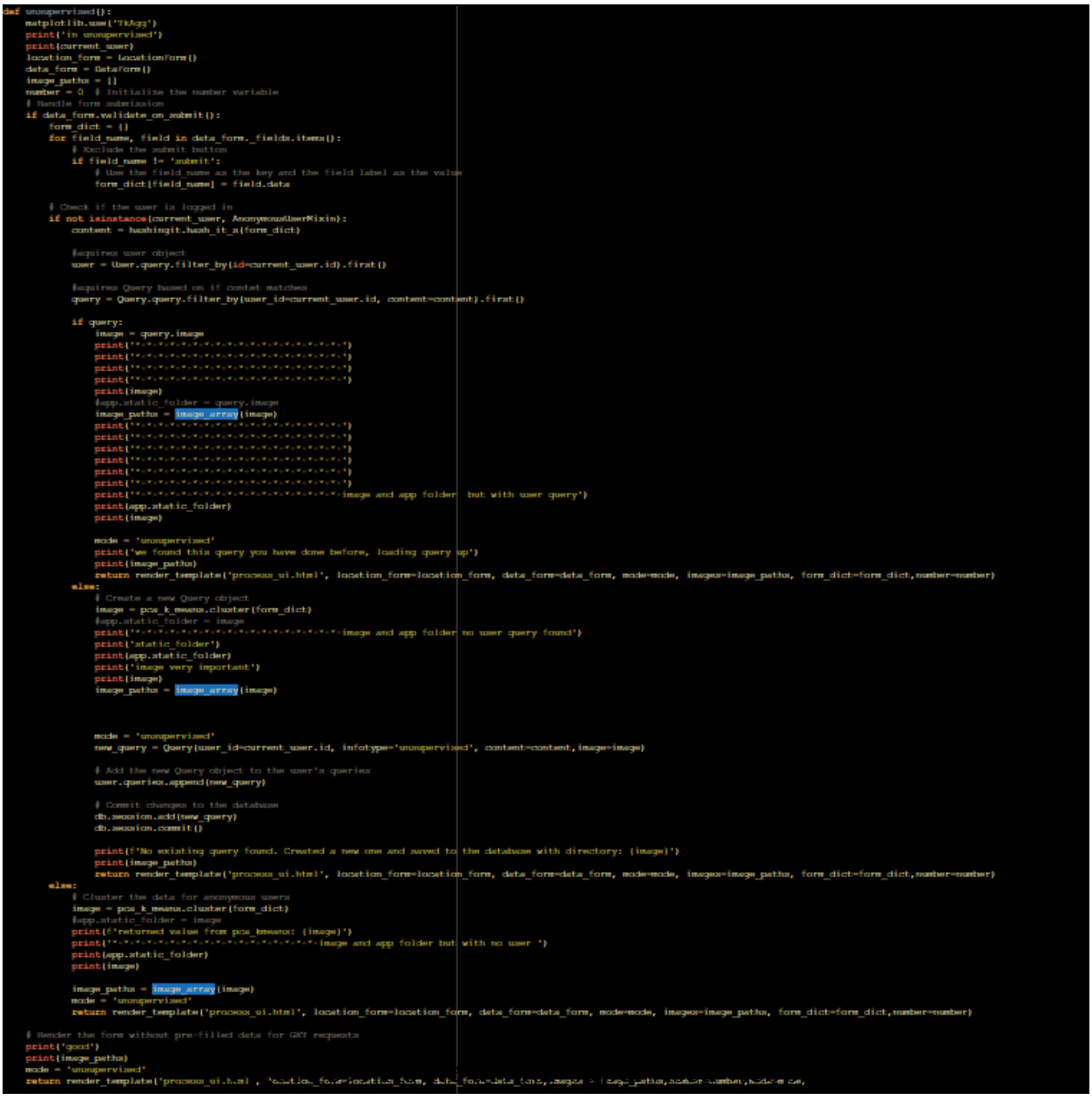


Figure 52: Unsupervised Learning routing code

5.8 Flask miscellaneous

The following are flask component not covered or covered clearly in the previous functionalities but are key to the development of the system

5.8.1 Models (models.py)

The Flask app has models which are represented by the models.py file in my program serves as the important piece for interacting with the database.

- User: The model shows the representation of the users of the application. contains fields for id, username, email, and password_hash. The password_hash field holds the hashed version of the user's password for security. Model validators make sure that the passwords are secure and protected. The database table has a one-to-many relationship with the Query table , allowing a user to have multiple queries related to their account.

- Query: The model stores information about the queries made by users. Contains fields for id, user_id, infotype, content, and image. The user_id field creates a foreign key relationship with the user table, linking the query to the user. The content field holds the data of the query in JSON format holding the content of what made the file. The set_content and get_content methods set and retrieve query content in JSON format. The image field stores the directory path for the images generated from the hash function in the data analysis for supervised and unsupervised.

5.8.2 Forms (forms.py)

In the Flask application, forms are necessary in taking input data from users and making sure that the data is valid before processing

- RegisterForm: This asks for a user input for registration by asking for a username, email, and password. It includes validation methods to ensure unique usernames and emails, and to confirm password by asking to re enter.

- LoginForm: For user login, this form requests a username and password. It validates by making sure that both fields are filled out before continuing with authentication.

- LocationForm: Passes a dropdown menu for users to pick a country from the selection choice created from a directory called cleaned_data where the folder name of each file data was stored and placed into the selection.

- DataForm: This form deals with input for the weather attributes such as temperature, humidity, precipitation, etc. There are validators to make sure that there is at least one weather attribute filled out and validates date inputs.

- DataExtractionForm: Used to input values for the data extraction web page, values include, latitude and longitude resolutions, the country to acquire data for, and date range. The resolution values and country selection are made before proceeding which is validated also.

These forms give the format of the input requirements of various web pages within the application, making sure of data integrity and user-friendly interactions.

5.8.3 views (routes.py)

The routes.py file in the Flask application defines the different URLs (routes) and the corresponding functions that handle requests to those routes. The functions, also known as view functions, process incoming requests, perform necessary actions, and return appropriate responses. Let's break down some of the key routes:

- start_page(): This route handles requests to the '/' and '/start' URLs. It renders a page where users can register. If the registration form is submitted successfully, the user is redirected to the 'unsupervised' route.

- login(): Handles requests to the '/login' URL. It renders a login page and validates user credentials. If the login is successful, the user is redirected to the 'unsupervised' route.

- logout_page(): Logs out the current user and redirects to the 'unsupervised' route.

- unsupervised(): Handles requests to the '/unsupervised' URL. This route deals with handling the unsupervised model .It provides an interface to the user and performs data clustering where the output is rendered on a template with the back to the user.

- supervised(): Manages requests to the '/supervised' URL. Similar to the 'unsupervised' route, but handles supervised learning tasks so takes values inputs instead of features.

- add_data(): Handles requests to the '/adddata' URL. This route lets users interact with the collect data function. It renders a form on a web page where users can input data parameters and initiates data extraction and cleaning processes.

These routes interact with forms, models, and a database to create a web app for weather analysis.

Chapter 6 Testing and Evaluation

The testing methodologies implemented aim to verify that every aspect of the system functions as intended and aligns with the end users' needs. Unit Testing validates individual functionalities to ensure their correctness. User Acceptance Testing evaluates the end-to-end flow of the system against functional requirements. Additionally, Detection and Recognition Accuracy Testing assesses the system's precision in face detection and recognition under various conditions.

6.1 Unit Testing

ID	Requirement	Type	Result
1	The system must use a form input to take in data from user	Functional	Pass
2	The system should be able to input the date period for data analysis	Functional	Pass
3	The system should be able to input the locations for data analysis	Functional	Pass
4	The system should be able to input the features to analyse for data analysis	Functional	Pass
5	System must be able to acquire data	Functional	Pass
6	User should see system has acquired data	Functional	Pass
7	System should be able to preprocess data	Functional	Pass
8	System should be able to analyse data	Functional	Pass
9	System will provide confirmation when user account has been made	Non-Functional	Pass
10	System GUI must be easy to use	Non-Functional	Pass
11	System must be able to handle user errors/mistakes by user	Functional	Pass

12	System must be able to handle user errors/mistakes by data	Functional	Pass
13	System is able to implement supervised ML	Functional	Pass
14	System is able to implement unsupervised ML	Functional	Pass
15	System should let user register	Functional	Pass
16	System should let user login	Functional	Pass
17	System should let user log out	Functional	Pass
18	System should create images from user input	Functional	Pass
19	System supervised ML should be able to have an accuracy of at least 80%	Non-Functional	Pass
20	System cluster weather for all countries	Functional	Pass
21	System cluster weather by only each country	Functional	Pass
22	System should keep the data secure between server and client	Non Functional	Pass
23	System should show outputs for query that have been asked before for logged in users	Functional	Pass
24	User should able to view their outputs	Functional	Pass
25	User shouldnt need to login to use the application	Functional	Pass
26	User that hasn't logged in shouldnt have access to preprocessed queries	Functional	Pass
27	Programmer should be able to identify and clean data after it has been collected if error is found on the data	Functional	Pass
28	User should be able to select arbitrarily from a pre-selected number of features	Functional	Pass
29	System should be able to handle error	Functional	Pass

30	User should be able to see visual of the output	Functional	Pass
31	Data should be cleansed before data analysis	Functional	Pass
32	Data should be preprocessed before data analysis	Functional	Pass
33	Data should be of a satisfactory standard before analysis	Non Functional	Pass
34	System should be able to process some level of Data in less than 2 minutes	Non-Functional	Pass

Table 6

6.2 User Acceptance Testing

All functionalities were thoroughly tested by being in the position of an end-user. The functionalities tested in Table 4 were tested first-hand and all performed as expected. Therefore, this method verifies that all the functionalities will work for the end-user.

6.3 Unsupervised ML Testing

The test used to quantify whether my machine learning was accurate. The dependent variables are the following

- Silhouette Score:
    - The Silhouette Score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
    - It ranges from -1 - 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighbouring clusters.

- A score close to 1 indicates that the object is appropriately placed within its cluster, while a score near -1 indicates that it might be better off in a neighbouring cluster.

- Davies-Bouldin Index

  - It measures the average similarity between each cluster and its most similar cluster, where similarity is defined as the ratio of within-cluster distances to between-cluster distances.

  - Lower values of the Davies-Bouldin Index indicate better clustering. A value of 0 indicates perfectly separated clusters.

Note: Davies-Bouldin Index does not require ground truth labels, making it useful for assessing the quality of clustering algorithms in the absence of labelled data. See figure 53 below.

```
# Compute silhouette score
silhouette_avg = silhouette_score(scores, kmeans_pca.labels_)

# Compute Davies-Bouldin index
db_index = davies_bouldin_score(scores, kmeans_pca.labels_)

print(f"Silhouette Score: {silhouette_avg}")
print(f"Davies-Bouldin Index: {db_index}")
```

Figure 53: Silhouette score and David Bouldin test code

For my independent variables I have the number of features and the date period i.e what the user inputs into the program, I will use the same country (Sweden) with the same amount of data (3 years) so that the prediction can be compared.

ID	Country	Number of input features	Number of day in date range	Number of tests	Avr Silhouette Score	Avr Davies Bouldin Index	Avr Time taken (s)
1	Sweden	1	10	3	0.99	0.40	20.66
2	Sweden	2	30	3	0.36	1.41	21.04
3	Sweden	2	50	3	0.38	1.68	20.48
4	Sweden	2	80	3	0.36	1.70	20.63
5	Sweden	5	30	3	0.34	1.54	19.84
6	Sweden	5	50	3	0.24	1.42	20.17
7	Sweden	5	80	3	0.34	1.32	20.18
8	Sweden	9	30	3	0.30	1.30	20.41
9	Sweden	9	50	3	0.29	1.31	20.36
10	Sweden	9	80	3	0.22	1.58	20.53

Table 7

Summary

PCA (Principal Component Analysis) has a time complexity of O(n^2 d + n * d^2), where: n is the number of samples, d is the number of features. KMeans has a time complexity of O(n * k * i * d), where: n is the number of samples, k is the number of clusters, i is the number of iterations. Combining PCA and KMeans in the f(g(x)) format where g(x) is pca implementation and f(x) is the k means algorithm, the time complexity put together would be O(n^2 d + n * k * i * d). This approach successfully handles high-dimensional data by first reducing dimensionality with PCA before clustering with KMeans this allows for scalability.
You can see that increase in the features and number of days to process has no to minimal effect on the clustering similarity when comparing to other clusters however the grouping between clusters is somewhat ambiguous but overall each cluster is correctly clustered. However as you increase features an duration, the

Davies Bouldin Index for each would seem to get better/grouped well, when you have more features and dates until grouping for each cluster would seem to fall off near the end for ID 10 (9, 80) this can be due to too many features. Initial data only having 1 feature used ID 1 (1,10) would seem to group very well on itself however that would make sense since clustering effects are minimised in singular dimensions. This all makes sense since I am using a pca matrix and not a conventional data frame, this tells us the values (eigenvalues) for each principal component are very similar.. See figure 54 below.

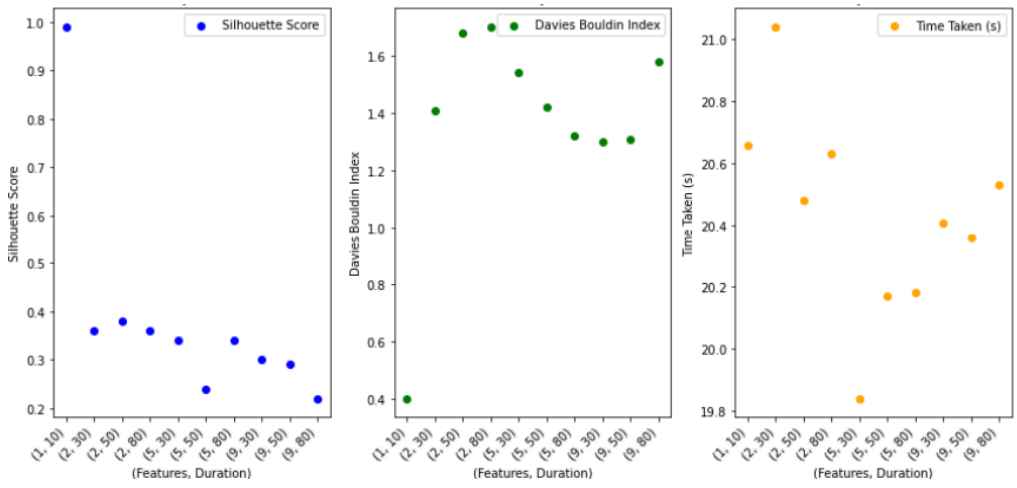


Figure 54: Unsupervised Learning routing code

6.4 Supervised ML Testing

The test used to quantify whether my machine accurate/ dependent variables are the following

- Accuracy:
    - Accuracy measures the proportion of correctly predicted instances out of the total instances. It's calculated by dividing the number of correct predictions by the total number of predictions.
- Precision, Recall, and F1-score:

    - Precision is the ratio of true positive predictions to the total predicted positives. It measures the accuracy of positive predictions.

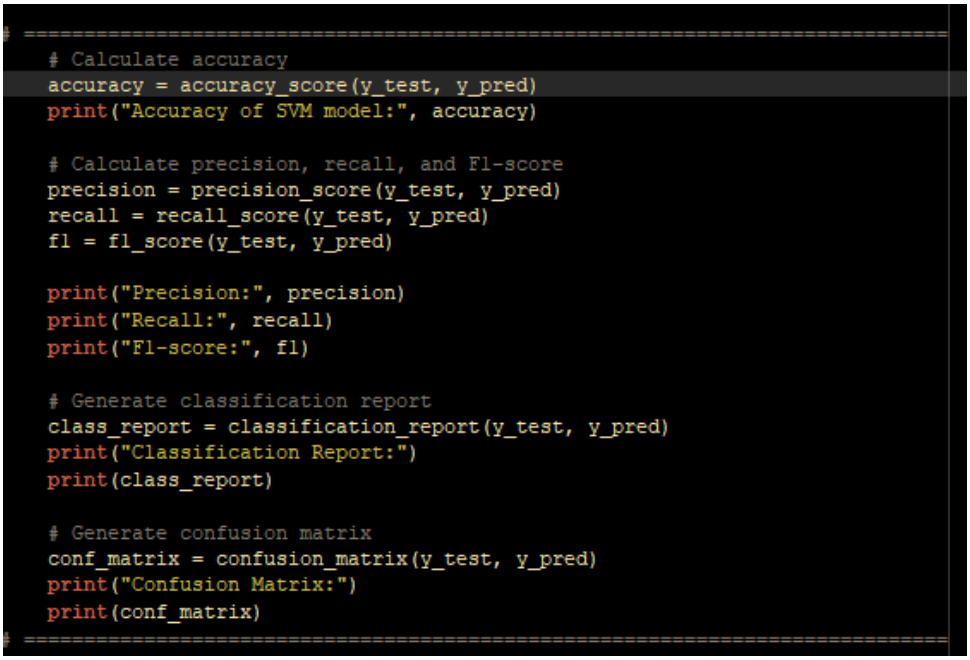
    - Recall (also known as sensitivity) is the ratio of true positive predictions to the total actual positives. It measures the model's ability to identify all relevant instances.

    - F1-score is the harmonic mean of precision and recall. It provides a balance between precision and recall.

- Confusion Matrix:

    - A confusion matrix is a table that summarises the performance of a classification algorithm. It compares the predicted classes with the actual classes in tabular form, showing counts of true positives, false positives, true negatives, and false negatives.

These metrics collectively provide insights into how well the classification model is performing, including its ability to correctly classify instances across different classes and the balance between precision and recall. They are essential tools for evaluating the effectiveness of a classification model and identifying areas for improvement.. See figure 55 below.



```
# ================================================================================
    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy of SVM model:", accuracy)

    # Calculate precision, recall, and F1-score
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)

    # Generate classification report
    class_report = classification_report(y_test, y_pred)
    print("Classification Report:")
    print(class_report)

    # Generate confusion matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:")
    print(conf_matrix)
# ================================================================================
```

Figure 55: Confusion matrix ,accuracy,precision,recall,f-score code

I will be using the same constants as I did for the unsupervised model (i.e Country:Sweden, number of tests:3)

ID	Number of input feature values	Number of day in date range	Avr Accuracy	Avr Precision	Avr Recall	Avr F1-Score	Time taken (s)
1	1	10	0.40	0.31	0.40	0.31	23
2	2	30	0.47	0.46	0.47	0.42	99
3	2	50	0.49	0.48	0.49	0.46	164
4	2	80	0.51	0.47	0.51	0.45	245
5	5	30	0.72	0.70	0.72	0.70	71
6	5	50	0.73	0.70	0.73	0.71	174
7	5	80	0.62	0.60	0.62	0.56	463
8	9	30	0.72	0.69	0.72	0.69	74
9	9	50	0.73	0.70	0.73	0.71	177
10	9	80	0.88	0.86	0.87	0.82	679

Table 7

Summary

The results show success in that, as you increase features and duration to analyse, the program becomes more precise, accurate, and has an overall better recall and F1 score. This indicates that the true positive and false negative values in the confusion matrix are increasing, which is very good, as usually recall and precision are opposing of one another. As you increase the value of the number of features, the accuracy (the main measure I consider) has an overall greater improvement than if I were to increase the duration of days to fit my program. However, both have a positive correlation with increasing all of the above metrics (precision, recall, F1-score, accuracy). The only issue is that, as I increase either two (features or duration), the time it takes for the process to complete is significantly more due to SVM exponentially being computationally more expensive and taking longer to Process

.

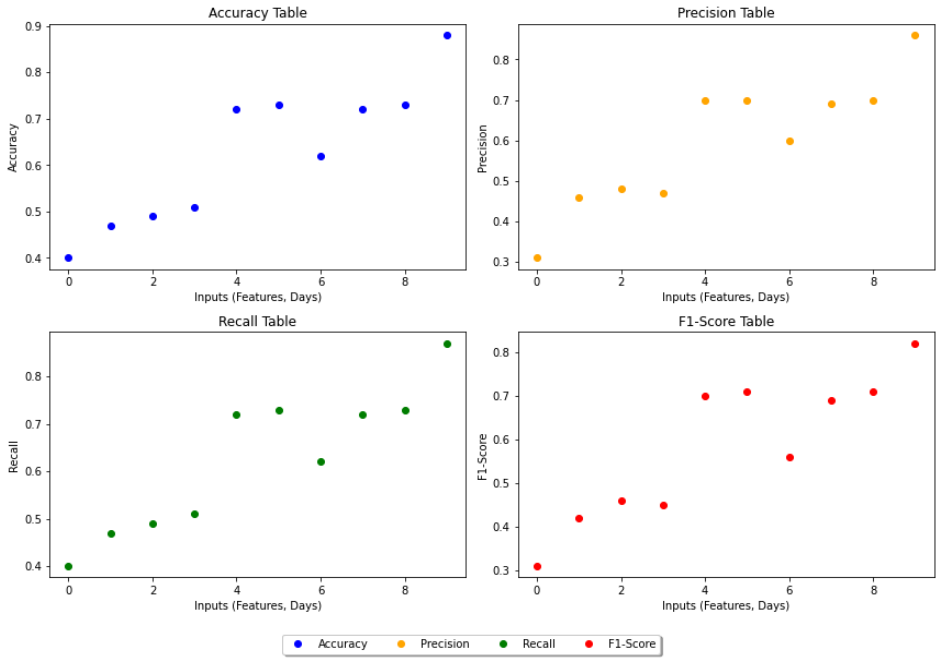


Figure 56: Unsupervised Learning routing code

Chapter 7 BCS Criteria & Self-Reflection

7.1 BCS Criteria & Self-Reflection

Learning Outcome	How I Achieve/Was i successful
LO1 - Conduct background reading, research, and user analysis	I have achieved this criterion based on the fact that my initial chapters show my research and how successful as to how I have implemented my research successfully into my project as well as to how they apply to my project.
LO2 - Demonstrate competence in project planning, risk assessment, time management, independent study, and adaptability	I have implemented a successful methodology to which had allowed for me to complete my project alongside i have completed my project to practically the entire functionality I have undergone difficulties such as deadlines outside of this project and externally of academics but maintained my quality of work
LO3 - Produce a design for an accessible and usable piece of software or research plan	I have achieved this as my design is clear and concise along with being what i wanted for my program
LO4 - Implement a technically competent piece of software or conduct in-depth research	My software has genuine real world implementation for research and personal use I have followed proper coding practises

LO5 - Evaluate project outcomes with reference to original objectives, background context, and expectations	My project has overall achieved what I desired however there have been changes made for example the usage of a GIS ordnance survey as applying the visualisation to the weather data was far to complex for me to integrate and a compromise was made in that I just used the simple plotting module matplotlib. There was also the usage of the spectral clustering algorithm in which I was adamant and excited in using however this would be deemed trivial in the functionality of the project so was omitted.
LO6 - Articulate legal, social, ethical, and professional issues surrounding the project	I have done so in chapter 7.
LO7 - Communicate technical information clearly to a broad, non-specialist audience	I've effectively fulfilled this criterion by ensuring that my technical information is presented in a clear and understandable manner where necessary, even to those without specialised knowledge in the subject matter. Through concise explanations, relatable examples, and visual aids where applicable,
LO8 - Structure and write an extended formal and technical document (dissertation) professionally	I have satisfied this requirement as all the necessities structure and formalities follow the BCS learning criteria in my dissertation

Table 8

In order for my project to satisfy the other BCS criterion I have also produced a table below highlighting six criteria and how my project satisfies them"

Criterion	How my Project satisfies this
An ability to apply practical and analytical skills gained during the degree programme	I apply practical and analytical skills gained during my degree program to collect, clean, and analyse complex weather data. These skills enable me to uncover valuable insights into regional weather patterns, contributing to a deeper understanding of climate phenomena and their geographical variations.
Innovation and/or creativity	My project is unique in that the combination of techniques and application applied tries to measure weather by looking at previous data.
Synthesis of information, ideas and practices to provide a quality solution together with an evaluation of that solution	I synthesise information, ideas, and best practices to develop a high-quality solution for weather involving integrating geospatial data, meteorological insights, and machine learning techniques. The solution is continually evaluated through validation, testing, and iterative improvement
That your project meets a real need in a wider context	My program contributes to our understanding of regional weather patterns, which is vital for applications in climate research, resource management, and disaster preparedness. By providing accessible insights into local climate changes, the project serves the larger goal of addressing climate-related challenges and enhancing our preparedness and resilience.
An ability to self manage a significant piece of work	I demonstrate my capability to independently oversee a substantial and multifaceted project. This encompasses project planning, data acquisition, analysis, software development, and iterative improvement. My ability to set milestones, adhere to timelines, and maintain project independence underscores my self-management skills in handling complex tasks.
Critical self evaluation of the process	I engage in critical self-evaluation. This involves continuous assessment of the methodologies, quality of results, efficiency of tools and practices, and responsiveness to feedback.

Table 9

Chapter 8 Project Ethics

Statement of Ethical Compliance

Data Category: A
Participant Category: 0

I confirm that I have read the ethical guidelines and will follow them during this project.

Further details can be found in the relevant sections

Legal Issues:

Complying to TOS : following the terms of service clarified by the weather data provider Visual Crossing to avoid legal issues such as access termination or legal action such as not sharing the data outside of personal business, educational or academic usages.

Social Issues:

On how Visual crossing will Address potential disparities in weather data to avoid, particularly for isolated groups affected by extreme weather events. By extension to my project the data should be representable of the area to which the data was acquired from failure to do this may lead to catastrophic possibilities if the people to use my program were to be dependant on such

Ethical Issues:

Visual crossing have defined how they handle/interact with (your personal) and their data on their website (Visual crossing 2023)

Chapter 9 Conclusion and Future Work

9.1 Achievements

Overall, the project has proven to be a success as all the basic/functional aims and objectives were complete. Developing a web application while incorporating Weather data analysis (both supervised and unsupervised models) and a database that all function together successfully I have gone from having no technical possible application in creating a web application to being able to understand a web application framework and understand how it works. Similarly, my prior knowledge of Machine learning was more theoretical and now I have successfully implemented these algorithms in my developed system. My personal aim for my final year project was to do something different and challenging that could potentially have mainstream real-world use one day, which I believe I have achieved. I have gained a huge insight into system development as I carried out the required stages in the software development cycle (software process) to complete my project. Theories and concepts from my university modules have been applied vastly throughout this project. My knowledge of Web Development, Database Management and Data Analysis have been applied in assisting the development of my program. In the same way, the understanding of SVM ,PCA k-means algorithms, API interaction and Data Interactions was learned throughout the length of this project. Though there are some areas for improvement, the final system is a great academic achievement and represents my dedicated hard work and perseverance.

9.2 Challenges

In the project, I had found a myriad of issues and weaknesses coming from the fact I had never developed or were familiar with Flask and web development. Although I had minimal previous experience with algorithmic concepts, application of this was both physically and mentally challenging. Dealing with time management was also very difficult , exacerbated by concurrent coursework demands. However, by planning ahead, I effectively allocated my time resources. Though initial hurdles

were problematic, being proactive led to me being successful in my project advancement.

9.4 Future Work

- for different time periods for each country region in which I compare instead of for a fixed time period for all the countries

- Would have made it so that registering would create an account with visual crossing and by extension gain their own api key

- Verbal explanation of the data visualisations (average for each feature in each area or explanation for such)

- detailed explanation of visualisations shown in the program

- Make the program be able to recognise more than countries but specific areas or regions defined by the user

- A page to view the queries instead of looking at the files created

There was a lot where the program could be improved overall, being more dynamic. However, this would legitimately become exponentially more difficult and would take too much time. So, I would consider these insignificant to what was actually successfully accomplished as they are only extensions to what had already been created .

9.4 Overall Conclusion

As a whole, this project has been a successful and a pioneering accomplishment and an excellent way to apply the knowledge gained through my undergraduate degree. I have acquired and applied new skills and knowledge which I can use to continue the development of this project and other future ventures.

References

1. Robert FitzRoy, Vice Admiral and former captain with a naval background, was responsible for developing the Met Office (Met Office,n.d.) .https://www.metoffice.gov.uk/research/library-and-archive/archive-hidden-treasures/robert-fitzroy

2. Chen, L., Han, B., Wang, X., Zhao, J., Yang, W. and Yang, Z. (2023). Machine Learning Methods in Weather and Climate Applications: A Survey. Applied Sciences, [online] 13(21), p.12019. doi:https://doi.org/10.3390/app132112019.

3. Denton, P., Parke, S., Tao, T. and Zhang, X. (2022). Eigenvectors from eigenvalues: A survey of a basic identity in linear algebra. Bulletin of the American Mathematical Society, [online] 59(1), pp.31–58. doi:https://doi.org/10.1090/bull/1722.

4. Reference: SciJinks. (n.d.). Forecast Reliability. Retrieved from https://scijinks.gov/forecast-reliability/#:~:text=of%20the%20time.-,However%2C%20a%2010%2Dday%E2%80%94or%20longer%E2%80%94forecast%20is,assumptions%20to%20predict%20future%20weather.

   Climate change now detectable from any single day of weather at global scale https://openreview.net/pdf?id=cHxHhdlC0S7

5. Hu, Y.H., Yu, S.C., Qi, X., Zheng, W.J., Wang, Q.Q. and Yao, H.Y. (2019). [An overview of multiple linear regression models and its application]. Zhonghua yu fang yi xue za zhi [Chinese journal of preventive medicine], [online] 53(6), pp.653–656. doi:https://doi.org/10.3760/cma.j.issn.0253-9624.2019.06.021.

6. Maruotti, A. (2011). Mixed Hidden Markov Models for Longitudinal Data: An Overview. International Statistical Review, 79(3), pp.427–454. doi:https://doi.org/10.1111/j.1751-5823.2011.00160.x.

7. Salcedo-Sanz, S., Rojo-Álvarez, J.L., Martínez-Ramón, M. and Camps-Valls, G. (2014). Support vector machines in engineering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(3), pp.234–267. doi:https://doi.org/10.1002/widm.1125.

8. Weber, P., Medina-Oliva, G., Simon, C. and Iung, B. (2012). Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. Engineering

Applications of Artificial Intelligence, 25(4), pp.671–682. doi:https://doi.org/10.1016/j.engappai.2010.06.002.

9. Feldman, D., Schmidt, M. and Sohler, C. (2020). Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering. SIAM Journal on Computing, 49(3), pp.601–657. doi:https://doi.org/10.1137/18m1209854.

10. Rajasekhar, N. and V.Rajini Kanth, T. (2014). Weather Analysis of Guntur District of Andhra Region using Hybrid SVM Data Mining Techniques. [online] https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=21B25a8a73547fe6ecffdb362e721dc608e13548

11. Zhang, M., Tian, Y., Guo, N., Su, Y., Wu, Y., Zhao, Y. and Yu, D. (2023). Multi-type Loads Characteristics Analysis Based on PCA-K-means Model in Extreme Weather. [online] ieeexplore. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10469566 .

12. Faysal Ibna Rahman (2020). SHORT TERM TRAFFIC FLOW PREDICTION USING MACHINE LEARNING - KNN, SVM AND ANN WITH WEATHER INFORMATION. (2020). INTERNATIONAL JOURNAL FOR TRAFFIC AND TRANSPORT ENGINEERING, 10(3), pp.371–389. doi:https://doi.org/10.7708/ijtte.2020.10(3).08.

13. Global land mask, land sea differentiation module. https://github.com/toddkarin/global-land-mask/blob/master/example_check_if_land.py

14. Reverse geocode function to find coordinates https://pypi.org/project/reverse-geocode/

15. Python Documentation https://docs.python.org/3/

16. Flask Documentation https://flask.palletsprojects.com/en/3.0.x/

17. Pandas Documentation https://pandas.pydata.org/docs/

18. Sklearn Documentation https://scikit-learn.org/stable/

19. How your data is used in Visual crossing https://www.visualcrossing.com/resources/documentation/weather-api/how-your-data-is-used-within-visual-crossing/

20. Depicts the standard climatological regions used by the Met Office.
https://www.metoffice.gov.uk/research/climate/maps-and-data/about/districts-map

21. Four Swedish climate zones and the position of four regions analysed.
https://www.researchgate.net/figure/Four-Swedish-climate-zones-and-the-position-of-four-regions-analysed_fig3_341702719

22. US Köppen classification
https://earthathome.org/hoe/w/climate/

Appendix A: Usage

In order to run the program you will need to set up the shown libraries but you will also need to sign up for an account with visual crossing and input an API key if you are going to use

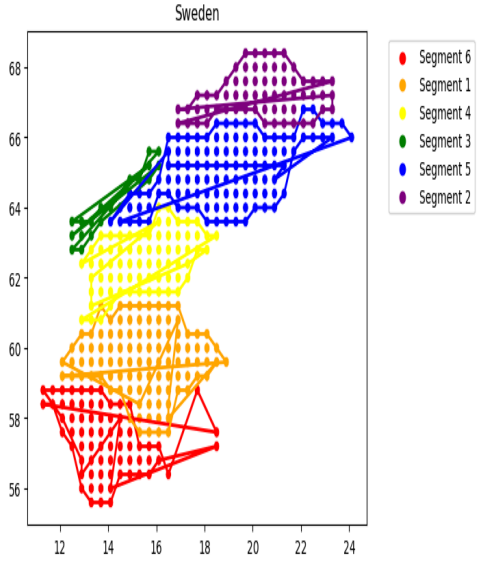
```
try:
    # IMPORTANT if you are going to run this program please sign up for an account at visual crossing  and paste the key into the key variable
    # To have a practical usage it will cost money however the program may wor for smally queries for territories instead of countries
    # e.g. Aruba ,St pierreand Miquelon etc.
    key = 'this key has been omitted'
    ResultBytes = urllib.request.urlopen("https://weather.visualcrossing.com/VisualCrossingWebServices/rest/services/timeline/"
                                         + str(lat1) + "%2C%20" + str(lon1) + "/"+str(fromdate)+"/"+str(todate)+"?unitGroup=metric&include=days&key="+key+"&
```

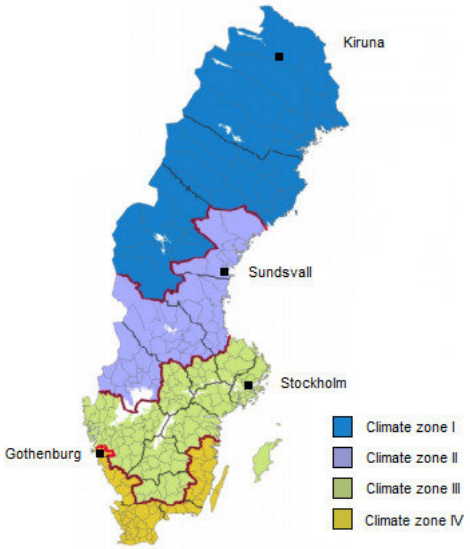
Appendix B: Risks & Contingencies

In the event of negative outcomes occurring the following contingencies will be considered.

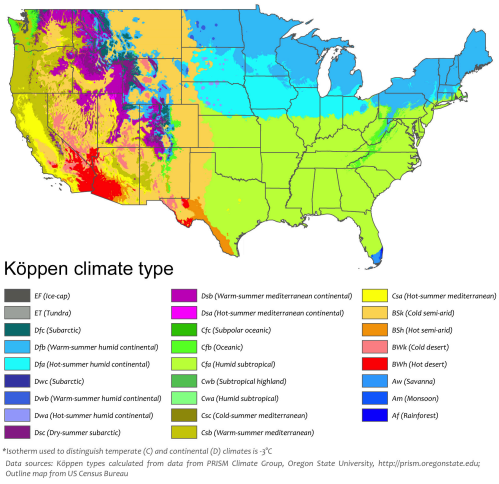
	Contingencies	Likelihood	Impact
Computer breaks down	In the event of this I will try to run my programs on the computers available in university or pass smaller pieces of information. I will also backup my data so that this shouldnt occur. Also, I will keep files of my code on the cloud so that it is accessible anywhere as long as I have access to my account.	Low and unpredictable. All I can do here is practise good computer etiquettes such as properly maintaining the computer. My computer is relatively new and should last long enough.	Medium. this wouldn't just affect my capability for this project but would have a cascading effect on other livelihood aspects increasing the amount of time required to do things but as long as an alternative solution is available there is a way.
Computer incapable of handling large amounts of data	In the event of this I will try to run my project on a more powerful computer i.e the one in the computer labs or pass smaller pieces of information.	Low. My computer has 16gb of ram which is a large amount and I shouldn't be working with data larger than a few Mb or a Gb at most.	Low. While it may be an obstacle at its worst, not only do I have the access to more powerful computers I can also use.
Data loss	Data will be backed up after each key increment being that a part of a section has been completed.	Low. data is backed up and shouldn't be affected by failure unless google servers fail.	Very low as long as I have access to where I obtained the information it shouldn't be as much of a problem as they are only used for testing the code along with my project being stored on the cloud it shouldn't be too much of an issue.
Running out of time	In this event I will allocate more time as I have left leeway during my process in the gantt chart however I must ensure key parts of the project are completed so that I am able to continue onwards I will also prioritise key parts over lower importance tasks.	Medium. What I am trying is new and untested before however implementations of smaller parts of this project have been done and should be possible for me to aggregate information based on this.	High this can affect the overall project's quality and leads to an incomplete dissertation and other aspects which will affect my grade.
Failures in sections of the project	I will have to prioritise more important parts of the project in order to ensure it functions however more time will be allocated to at least ensure the minimum.	High . The more detailed and complicated a system is, the more possible areas of failure there are to be. There are many components to my project.	High . This could impact other sections and may limit me in how I continue on with my project significantly.

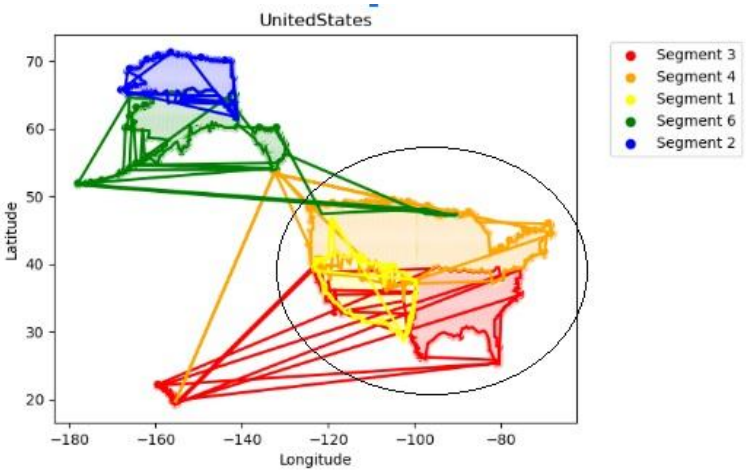
Appendix C: Shown plots compared with plots from research / media



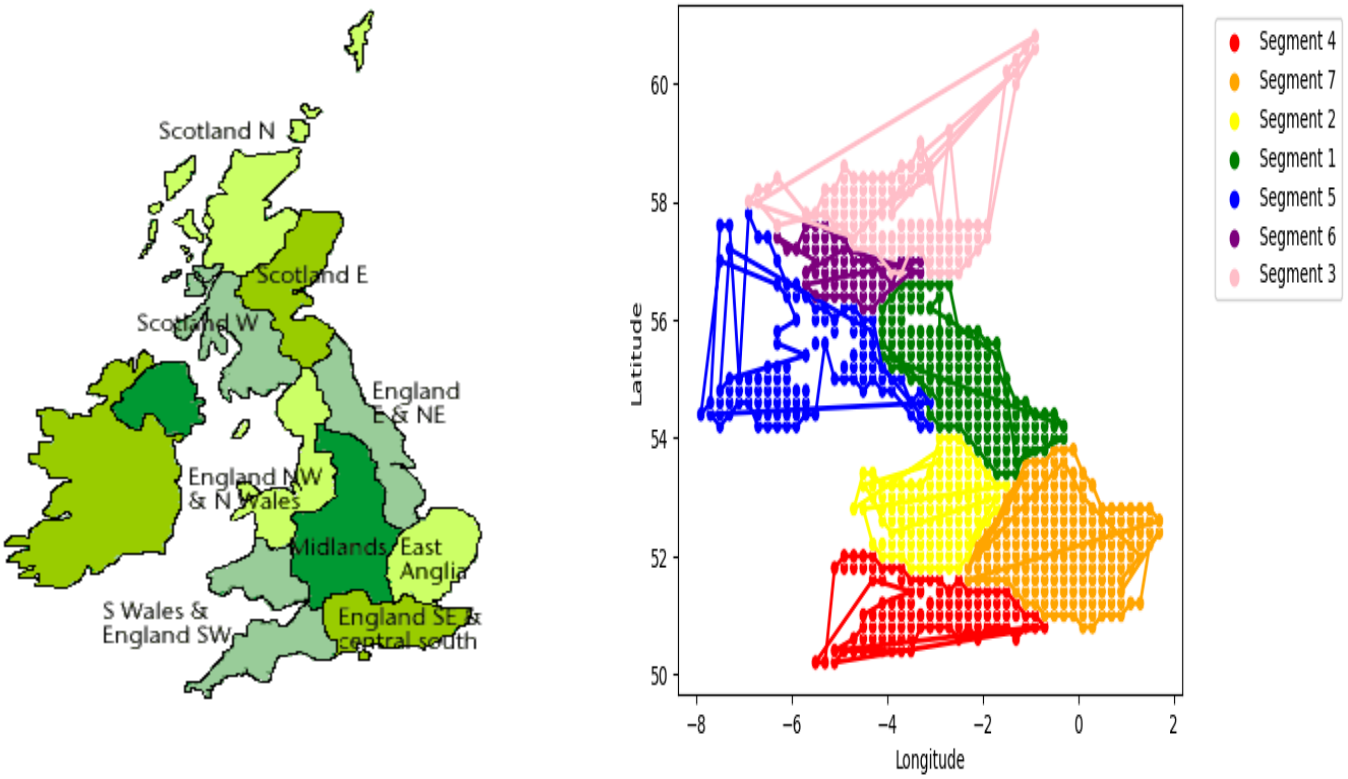


My map against Swedish climate zones



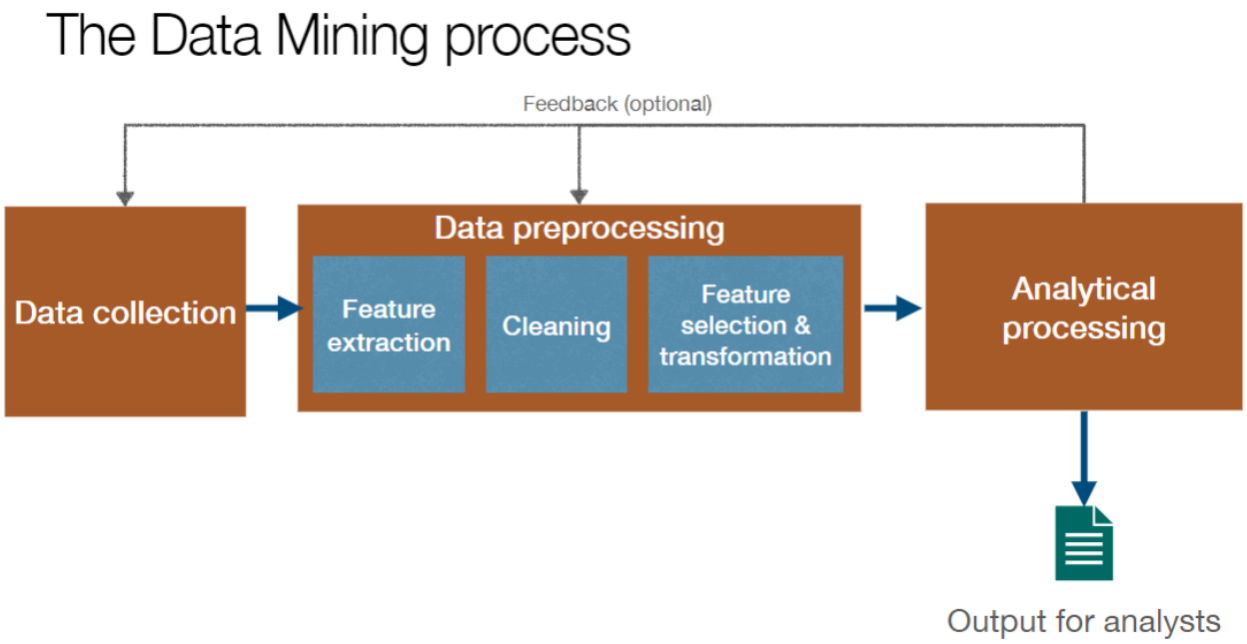


My map against US Köppen classification



My map against met office map of climatology

Appendix D: Data Mining process



Appendix E: Determining number of cluster via k-means inertia/WCSS

```python
# this will validate that we have chosen the best value to represent the clustering process in our program
# it will look for the value which represent the elbow point where it is best representing of the clusters in our graph
def k_means_inertia(scores):
    print('inertia step 2')
    wcss = []
    if scores.shape[0] < 2:
        return 1
    for i in range(1, 16):
        kmeans_pca = KMeans(n_clusters=i, init='k-means++', random_state=57)
        kmeans_pca.fit(scores)
        wcss.append(kmeans_pca.inertia_)

    # Normalize WCSS values between 0 and 1
    max_wcss = max(wcss)
    normalized_wcss = [j / max_wcss for j in wcss]

    # Identify the elbow point
    elbow_point = None
    prev_diff = 2
    for i, value in enumerate(normalized_wcss):
        diff = prev_diff - value
        if diff < 0.04:  # threshold can be adjustyed as needed
            elbow_point = i + 1
            break
        prev_diff = value

    if elbow_point is not None:
        print(f'Elbow Point (Optimal k): {elbow_point}')
        return elbow_point
    else:
        print('No suitable elbow point found.')
        return None
```