University of Essex
Department of Mathematical Sciences

MA981: DISSERTATION

Predication of rainfall in the UK using machine learning techniques

Mobin Ahmed
2202584

Supervisor: Dr Osama Mahmoud

October 9, 2023
Colchester

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Motivation

Climate change is becoming a big issue around the world. So it is important to understand how the climate is changing so that we can prepare for the future. Knowing the determinants of climate change like temperature, rainfall, wind speed, etc. is very important. In [1], it is written that: "temperature in UK is increasing day by day and becoming more serious for humans". We are experiencing random rainfall, sometimes more , sometimes less, but not in a systematic way, causing loss to the economy, agriculture, industry, and human daily lives. [1]

In terms of climate change, in [1], it is written that: "the UK is experiencing rising temperatures. From 2012 to 2021, it has been on average 1 °C warmer than the 1961 to 1990 average. Since 2003 was the warmest year in 10 years. UK's hottest year, which was 2022, with temperatures above 10 °C. Along with this, many other changes in the climate are occurring. Many other changes such as warming in the oceans, melting polar ice and glaciers, rising sea levels, and more extreme weather events". [1]

As greenhouse gas emissions continue to rise, we are experiencing warmer and wetter winters, hotter and drier summers, and more frequent and intense weather extremes. In [1], it is written that: "if this gas increase continues, by 2070, UK winters are predicted to be between 0.6 and 3.8 °C warmer, and summers are predicted to be between 1.3 and 5.1 °C

warmer". This will be a threat to public health. As seen, warmer and drier summers increase the risk of wildfires in the UK. [1]

Summer rainfall is also affected by climate change in the UK. Especially in urban areas of the UK, there is flash flooding. In [1], it is written that: "this is predicted to cause flash floods twice as often as in 1990. Increased flooding will affect the environment, infrastructure, transport, and water systems very badly". [1]

So, it is very important to have regular information about climate change. For this, forecasting or predicting some of the determinants of climate change like temperature, rainfall, wind speed, etc. is also important. In [1], it is written that: "the benefits of the determinants forecasting and predicting and climate change information will be felt globally, like improving agricultural productivity, increasing innovation, improving health and wellbeing, increasing food security, and reducing biodiversity loss".

1.1.1 The main objective

So the focus of my study is that I will analyze data gathered by government institutions in the UK to model changes in climate and predict potential future scenarios of climate determinants such as rainfall across various regions of the United Kingdom. The main objective of this report is to make prediction of rainfall in the UK using machine learning techniques.

1.1.2 Aiming to

I aim to examine the existing literature on machine learning for climate prediction, focusing on studies related to the UK context. I aim to gather and pre-process UK MET Office Weather Data from Kaggle [11], to select and apply appropriate machine learning techniques to train predictive models to make prediction of rainfall in the UK using machine learning techniques, and I also aim to evaluate and compare the performance of the developed models and identify the most effective approaches for prediction. Interpret the results obtained from the predictive models and discuss their implications for the prediction.

1.2 Contributions

This report aims to contribute to provide a comprehensive analysis of machine learning techniques for predicting total rainfall for a month of a specific year across various regions of the UK. The findings of this study can help researchers understand the potential of machine learning to address climate change challenges on a regional scale. Moreover, the identified best techniques and recommendations can guide the development of more accurate and reliable climate change determinant prediction models, assisting in evidence-based decision-making and proactive planning for climate change mitigation and adaptation actions.

1.3 Previous works

This section gives some overviews of research works that use spatio-temporal data to train rainfall forecast models, temperature forecast models, and wind speed forecast models based on various machine learning techniques like Support Vector Machine (SVM), Neural Networks, long-short-term memory (LSTM) Networks, XGBoost, etc.

The authors **Barrera-Animas and others (2022)** in the study [13] presented a comparative analysis using rainfall prediction based on machine learning techniques and some deep learning techniques. In the study [13], Long Short Term Memory (LSTM), Stacked LSTM, Bidirectional LSTM Networks, and XGBoost were used to predict hourly rainfall. The cities of Bath, Bristol, Cardiff, Newport, and Swindon were used for regional rainfall prediction. In [13], it is written that: "the dataset consists of variables such as temperature, pressure, humidity, wind speed and direction, percentage of clouds, volume of rain and snow, city name, latitude and longitude coordinates, timezone code, date, and code for label direction".

In the study [13], the pre-processing procedure was performed as follows:

- Separating weather data by each city
- Filling up missing values
- Removing the sea and ground-level measurements
- Removing the city name and date-time features

- Refilling the missing values of rain 1h, rain 3h, snow 1h, and snow 3h features with zero
- Separating weather condition codes to help build vector rows of invariant length
- Standardizing the number of weather condition codes to allow the building of five datasets with the same feature length [13]
- Computing one-hot encoding for the weather id codes to downsize values in range one or zero One means the weather condition is present, and zero means the weather condition is absent. [13]
- Structuring the feature vector rows to structure five major UK cities with the same structure [13]

For the feature selection, the study [13] used the Pearson correlation coefficient. Features that had a coefficient value equal to or greater than 0.7 for each city are retained. Highly correlated features are considered for removal for each city. Finally, the dataset was downsized from 43 to 11 features. [13]

Rainfall prediction process in the study [13]:

LSTM Networks, Stacked LSTM networks, and Bidirectional LSTM Networks against an XGBoost decision tree and an one of the algorithm from AutoML used in the study [13] for predicting 8 hours of rainfall on time series data from five major UK cities. The study [13] used the XGBoost model for each of the five datasets as the basis. To get the best hyperparameter values for training each model, the grid search was performed. AutoML tools are used with regressor algorithms. LSTM Networks, Stacked LSTM Networks, and Bidirectional LSTM networks showed the best accuracy for each dataset. In the study [13], grid search was performed for adapted models with each dataset of the cities. It helped to find the optimizer and the learning rate. The tested optimizers were Adam and Stochastic Gradient Descent (SGD). Learning rate values were 0.3, 0.1, 0.05, 0.01, 0.007, and 0.001 while testing. The activation function in the output was ReLU for all models, and the loss function was Mean Square Error (MSE). [13]

In the study [13], the time-series weather dataset sorted in ascending order and then divided into training, testing, and validation. Training and testing for the XGBoost and

AutoML models were 85% and 15% respectively.

Evaluation results with conclusions in the study [13]:

From all of the models adapted in the study [13], the Stacked LSTM model and Bidirectional LSTM gained lower evaluation values with RMSE, MAE, and RMSLE. The stacked LSTM gained loss values range between 0.0014 and 0.0001 through all of the used cities; RMSE values range between 0.0375 and 0.0084; MAE values range between 0.0071 and 0.0013; and RMSLE values range between 0.0157 and 0.0037. Where Bidirectional LSTM obtained values range between 0.0014 and 0.0001, 0.0377 and 0.0099, 0.0072 and 0.0015, and 0.0111 and 0.0044 for Loss, RMSE, MAE, and RMSLE, respectively. Then, a stacked LSTM Network with 10 hidden layers showed the worst results in the training and validation loss curves. And stacked LSTM with three hidden layers showed the worst training and validation loss curves. So, in the study [13], it was proven that these neural networks with more hidden layers are less perfect in hourly rainfall prediction. Though it is possible to use Bidirectional LSTM for rain prediction, it has one major drawback, which is the inability to generalize adequately. For this reason, models can perform well on training data and cannot perform well in test data. [13]

From [13], some more information about LSTM:

The most commonly used Recurrent Neural Network (RNN) cell variant is long-short-term memory (LSTM) Networks. It is popular because of its ability to learn long dependencies within sequential data. Fields like human trajectory prediction, traffic forecasting, and speech recognition have been used with LSTM. It is capable of learning the dependencies of at least two previous states and the current state. The use of three gates with a hidden state minimized the vanishing gradient effect. In [13], it is written that: "the gates are commonly referred to as input (which defines the amount of information of the new state that is used), output (which defines the amount of information that is used from previous states), and forget (which defines the amount of information of the internal state that passes to the next layer)". [13]

The authors **Cifuentes, Jenny, Marulanda, and others (2020)** in the study [15] reviewed different machine learning strategies from different literature. For temperature forecasting,

machine learning techniques can predict temperatures based on input features including previous values of temperature, relative humidity, rain, wind speed, and solar radiation. The study [15] found deep learning techniques produce smaller errors than Artificial Neural networks (ANN) for 1 step ahead. Support Vector Machine (SVM) gained good accuracy at the global and regional scales, where good accuracy depends on input combinations, good architecture, and learning techniques.

The study [15] mentioned that to predict temperature with time series data, Artificial Neural Networks (ANN) and Support Vector Machines (SVM) would be most popular. ANN models developed to predict temperature which are MultiLayer Perceptron Neural Networks (MLPNN) and Radial Basis Function Neural Networks (RBFNN) with Levenberg-Marquardt and Gradient Descent. The Long Short Term Memory (LSTM) has been used to predict hourly air temperature for 1 step ahead with small errors. [15]

The study [15] stated that with a percentage of 57%, ANN were developed to predict monthly, daily, and hourly temperatures and 43% SVM. Where SVM reported better performance. In [15], it is written that: "ANNs such as MultiLayer Perceptron Neural Networks (MLPNN) and the Radial Basis Function Neural Networks (RBFNN) were the most commonly used techniques to predict air temperature. LevenbergMarquardt and Gradient Descent were the most commonly used optimizers. For the hidden layer, the activation function was Sigmoid. For the output layer, the activation function was Pure Linear. Cross-validation used to set the hyper-parameters". [15]

The study [15] mentioned the following gaps as well:

- The review mostly focused on the local analysis of the air temperature.
- This review does not include a time horizon analysis elaborately.
- The size of the dataset for training and testing should be determined to compare techniques.
- A comparative analysis with all the available ANN-based techniques should be carried out for different time horizons. [15]

- More related features or variables should be used to increase prediction accuracy.
- Different feature selection techniques like Automatic Relevance Determination, closely related sparse Bayesian learning, or the Niching genetic algorithm have not been used. [15]
- Deep learning strategies have not been used.
- In the case of a recurrent neural network (RNN), the size of the time series required to accurately predict temperature should be studied. [15]
- Statistical significance tests have not been taken. [15]

The authors **dos Santos and Rochelle Schneider (2020)** in the study [16] focused on spatio temporal satellite data to predict maximum temperature. The study [16] divided the work into two divisions. Number one, the work is estimated for London at 2 km2 during summer time, in the years 2006 to 2017. For this, the study [16] used statistical strategies and satellite provided features. Number two is the gradient boosting technique which is used to predict the maximum temperature for an urban place. Six regression models with six satellite products, three geospatial predictors, and 29 stations were used. In [16], it is written that: "the regression method for training and testing and machine learning for prediction. Land surface temperature, Julian day, normalised difference vegetation index, digital elevation model, solar zenith angle variables were used with the gradient boosting model. The performance was found as R2 = 0.68, MAE = 1.60 ° C, and RMSE = 2.03 ° C". The study [16] mentioned that this work can be replicated in other UK cities, which is beneficial for national temperature related mortality measurements.

Data used in [16]:

- **Meteorological data**

  Spatiotemporally adjusted with predictors, data from the Met Office Integrated Data Archive System 2006 for maximum temperature has been used. [16]

- **Satellite data**

  Terra is a sun-synchronous polar orbit satellite with two sensors: one is MODIS and the other is the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) for satellite data. Some pre-processing steps were performed on the data. [16]

- **Auxiliary data**

  Four auxiliary variables were added, which were reported in the literature. [16]

Methodology in [16]:

In [16], it is written that: "from the 6 methods, the Method 0, which is stepwise linear regression to explore the potential of each predictor regarding maximum temperature in each model. Method 1, which is linear regression model, defines the relationship between the target and the predictors as a straight line. Method 2, which is a decision tree designed to determine the most logical data splits it into smaller subsets. Method 3, which is Supervised Random forest, this multiple tree-based structure helps the model become more robust than a single Decision tree and reduces overfitting. Method 4, which is Supervised gradient boosting, built using multiple decision trees with weighted conditions. Method 5, which is Support Vector Machine with a Linear Kernel. And finally Method 6, which is Neural Network with Multi-Layer Perceptron Feedforward Class". [16]

The study [16] collected the data for 29 weather station regarding maximum temperature and the predictors were combined in a unique and large database. In the study [16], the latitude, longitude, and julian day variables were included as spatial and temporal components along with the maximum temperature. Then, the predictor variables are the normalized. The study [16] used pearson's correlation coefficient to find the correlation among predictors and target variable. For all the regression model, the data set was split into 70% training 30% testing. In the study [16], R2, MAE, and RMSE were used for performance evaluation. Also nine groups of predictor variables were made using stepwise linear regression to evaluate the 6 methods where sample size was 6.442. [16]

Performance of the models in [16]:

In the study [16], the range of R2 between 0.32 (for Method 2, Groups 2 and 3) and 0.68 (for Method 4, Groups 4 and 5). The RMSE ranged from 2.99 ° C (for Method 2, Group 2) to 2.03 ° C (for Method 4, Group 5). The MAE ranged from 2.27 ° C (for Method 2, Group 3) to 1.59 ° C (for Method 4, Group 4). The study [16] found that Gradiant Boosting method performed best than the other methods in all groups except Group 1, where the lowest RMSE = 2.46 ° C and the lowest MAE = 1.92 °C were found in Method 5 and Method 6. Method 2, which was decision tree performed worst was not considered. [16]

The authors **Ghorbani, MA, Khatibi, and others (2016)** in the study [17] used Artificial Neural Networks (ANNs) and Genetic Expression Programming (GEP) to predict short term wind speed. Eight years of collected data from Kersey site in Colorado, USA, were used. From the dataset, the study [17] used 7 years (2005 to 2011) for training and a year, 2012, for testing. The results found from ANNs and GEP, compared with multiple linear regression (MLR) and the Persistence method. The model performances were evaluated using the correlation coefficient, RMSE, Nash Sutcliffe efficiency coefficient, and Akaike information criterion (AIC). From the estimates, The study [17] found that forecasting wind speed is possible using past records, but for this, the maximum lead time is 14 hours. The results showed that ANNs and GEP are equally important, but you cannot ignore MLR.

Source of data in [17]

In [17], it is written that: "Wind speed time series on hourly basis were got from the Kersey site in Colorado, USA. Where latitude 40°22‘36“ north, longitude 104°31‘ 55“ west, and altitude 1409.7 m above sea level." The dataset was downloaded from the Colorado Climate Center. Here is the link: [2]

Applied Methodology in [17]

In [17], it is written that: "input variables can be determined by the cross correlation between the present wind speed time and wind speed at time lagged. In cross correlation, values greater than zero and equal to zero are counted to determine models". The study [17] found that 1 hour to 14 hours means 1 (M1) to 14 (M14) models of time lagged. In [17], it is written that: "to predict the wind speed one can use maximum previous 14 values to train and test the developed models. The M1 to M14 models with different input structures were trained and tested". [17]

Used ANN and performance in [17]: The study [17] used correlation coefficient (CC), root mean square error (RMSE), and Nash Sutcliffe efficiency coefficient (E) and statistic values found 0.892, 0.839 m/s, and 0.796, respectively which was best performance on training. For the validation performance, CC, RMSE, and E statistic values were 0.914, 0.874 m/s, and 0.834, respectively. So, in this case, ANN (M8, F8, 1) is the most effective and appropriate model. [17]

Used GEP (Genetic Expression Programming) and performance in [17]: The performances from model M1 to model M14 in the study [17] was evaluated regarding GEP. For the training, the best performance found on the M2 model with F3 (fuction type). Where CC = 0.889, RMSE = 0.869 m/s, E = 0.79, but AIC = −1615.9, which is not good. For testing, the same model with same function on training phase, performs best where CC, RMSE, and E statistic values were 0.884, 0.909 m/s, and 0.781, respectively. However, the quality of these performance parameters drops during testing phase. Consequently, M14 model with F2 gained the best performance on the training phase, where AIC = −1110. [17]

Used MLR (Multiple Linear Regression) and performance in [17]: The study [17] used MLR for the wind prediction. In [17], it is written that: "CC = 0.888, RMSE = 0.874 m/s, E = 0.788 and AIC = −1527.0 on the training phase regarding model M13. On ther other hand, for the testing phase, it was found CC = 0.882, RMSE = 0.914 m/s, E = 0.778 regarding model M6. Consequently, with AIC = −1487.4, the model M1 found better". [17]

Used persistence method and performance in [17]: The study [17] run this persistence method without training. For example, this model was only made with M1, which had no parameters. Where the study [17] found that CC = 0.886, RMSE = 0.905 m/s, E = 0.772 and AIC = −1148.1 on the training phase. On ther hand, for the testing phase, it was found that CC = 0.879, RMSE = 0.958 m/s, E = 0.757, and AIC = −12.3. [17]

The best model in [17]

The study [17] found that ANN model performs well. And it was better than the GEP, MLR and Persistence method regarding *RMSE*. But the Persistence method produces the highest Akaike information criterion (AIC) values. [17]

The authors **Kumar, V Bharat, and others** in the study [18] did correlation analysis for input parameters where higher correlations were checked and got them as input for wind speed prediction. The study [18] compared errors for deep learning techniques like long short term memory (LSTM) and gated recurrent units (GRU). In terms of errors, in [18] the

authors discovered that GRU outperformed LSTM in predicting wind speed.

Implementing LSTM and GRU in [18]

1. **Data:**
   A year's worth of data on hourly wind speed was obtained from the Amrita School of Engineering, Bangalore, weather station. 8760 samples were considered, of which 6132 were used for training and 2628 for testing. Min-max standardization is run to reach a 0 to 1 scale for training time minimization. [18]

2. **Feature extraction:**
   After correlation analysis, all of the considered variables were less correlated with wind speed, where the variables are temperature, precipitation, air density, surface radiation, radiation toa, and cloud cover. [18]

3. **Data converted to sequential data:** In [18], it is written that: "the data was converted into sequential data with one step lag, two step lag, and so on. Lag means preceding terms to be considered while predicting. For example, a two-step lag implies that data from the preceding two terms must be considered".

Working process in [18]

In [18] the author's working process was simple, as usual. Importing data, pre-processing, splitting data into training and testing, model selection, and wind speed prediction. From the data sample, in [18] the authors used 70% for training and 30% testing and used long short term memory (LSTM) and gated recurrent units (GRU) to train the dataset. For these models, hyperparameters such as Hidden layers, Neurons, Epocs, Batch size, and activation function were used. [18]

Computational time and hardware requirements in [18]

8 GB of RAM, an Intel Core i5, and a 7200U CPU with 2 cores used to process the LSTM and the GRU. It took 54 ms for the LSTM run and 51 ms for the GRU run. [18]

Performance evaluation and results in [18]

The results showed more correctness and little error in prediction. Gated recurrent units (GRU) showed better performance with less lag time, but when the time lag increased, long short term memory (LSTM) had less error than GRU. [18]

However, in [18], it is written that: "both LSTM and GRU help with issues like vanishing and exploding gradients. Moreover, for models, computational time was less. It helped to consider a larger number of samples for prediction". [18]

The authors **Bochenek, Bogdan, and others (2022)** in the study [14] did an analysis of the 500 most relevant articles published since 2018 regarding machine learning in the field of numerical weather prediction and climate change using Google Scholar. In [14], it is written that: "the most common topics in numerical weather prediction are research in photovoltaics, wind energy, and atmospheric physics and its processes. And in climate research, the most common topics are parametrizations, extreme events, and climate change". In [14] the authors mentioned that in these topics, it is possible to extract the most commonly examined meteorological fields, like wind, precipitation, temperature, pressure, and radiation, using methods like Deep Learning, Random Forest, Artificial Neural Networks, Support Vector Machines, and XGBoost regarding the mentioned countries.

In [14] the author's main goal was to present a review of the machine learning techniques and applications for the topics, which are meteorology and climate analysis. Important and complex problems in weather prediction and climate change over different temporal and spatial dimensions can be solved using machine learning. [14]

Results from the study [14]:

1. Ten most common phrases from reviewed articles related to numerical weather prediction and machine learning.There is a figure for that. From the figure, I saw that the highest count was found for the wind forecasting. [14]

2. Ten most common phrases from reviewed articles related to climate and machine learning. There is a figure for that. From the figure, I saw that the highest count was found for climate change. [14]

3. Five most common meteorological fields in numerical weather prediction related articles. There is a figure for that. From the figure, I saw that the highest count was found for the wind feature. [14]

4. Five most common machine learning methods in numerical weather prediction related articles. There is a figure for that. From the figure, I saw that the highest count was found for the Deep learning (DL) method. [14]

5. Five countries, like China, the USA, Australia, India, and Germany, were found to be related to the study [14] about climate. Where China gained the highest count and got 40% with specified regions through out the papers. [14]

Some key points found in their review [14] are as follows:

- To determine and analyze problems like meteorology and synoptic climatology, a successful technique can be machine learning. [14]

- In many circulation problems, quantitative definitions that are unambiguous can be found in the weather type, while in others, there are usually easy and clean definitions without quantitative criteria and indices regarding local scale. [14]

- Machine learning can be used to determine the objectives of the mentioned applications, both in a supervised and unsupervised way. To determine circulation, weather types, characteristics of meteorological variables, and environmental variables, KMeans clustering is used. [14]

- In [14], it is written that: "the use of ML techniques in many studies is because of the increase in computer power and new technologies, the development and access to specialized software, and improved reanalysis. Though scientists faced problems with limitations like related knowledge, for this workshops and seminars online can help to tackle this problem. Interdisciplinary cooperation between scientists is another key aspect of using machine learning". [14]

- To correctly implement any ML methods, this is really good to understand the working process and interconnections between the meteorological and environmental variables regarding the problem. In [14] the authors believed that ML methods will be key to

weather prediction. Methods mentioned in the study [14], like random forest (RF), need less knowledge in the field of machine learning (ML) and easy for the beginners. But for convolutional neural network (CNN), one have to be experienced. [14]

The authors **Tharun, VP, Prakash, and others** in the study [19] did a comparative study of regression techniques like support vector machine (SVR), random forest (RF), and decision tree (DT). The main objective of the study [19] was to predict rainfall in Nilgiris, Tamil Nadu, using statistical modeling and regression techniques. In [19] the authors used methodologies such as data collection, data pre-processing, model development, and performance measures. In [19] the authors collected the data from the India Meteorological Department which is in Chennai and from the Public Works Department of Coonoor, for a period of 9 years (2005 to 2014). In [19] the authors divided the dataset into 80% for training and 20% for testing. In the study [19], after pre-processing, 28 inputs were labeled separately and the target variable was rainfall. Three models such as SVR, RF, and DT were implemented with various parameter tunings. To provide evaluation results, in [19] the authors used a scatter plot of actual vs predicted values, an R2 score, and an adjusted R2 score. The results showed that RF with a 0.981 R2 score and a 0.980 Adjusted R2 score performed best. In [19] the authors also mentioned that statistical modeling fails to provide good accuracy in rainfall prediction because of the complexity of the input parameters. For future work, in [19] the authors suggested that to use nonlinear Neural Networks to predict rainfall because the dataset followed a nonlinear pattern. [19]

In the following sections of the report, It will be like, in Section 2, I will explain the methods that I will use and will describe the dataset. In Section 3, experiments and results will be explained, where subsection 3.1 will describe exploratory data analysis and subsection 3.3 will describe results. In Section 4, I will provide a conclusion of the report.

CHAPTER 2

Methods

In this part of the report, I am going to include background methods that I will use in the project. This includes machine learning, different techniques of machine learning, and evaluation measures of the models of the project. I will describe the dataset as well.

2.1 Overview of machine learning

Definition 2.1.1 *(see [15]) It is written that: "machine learning is a part of artificial intelligence whose algorithms are developed to obtain a mathematical model for fitting the data. This mathematical model learns the estimation of unknown parameters in the model based on given data. After fitting the known data accurately, this is used to perform the prediction on new data".*

Definition 2.1.2 *(see [12]) It is written that: "machine learning is a category of artificial intelligence that enables computers to think and learn on their own so that computers can modify their actions in order to improve the actions to gain more accuracy".*

There are three common criteria implemented in machine learning: supervised learning, unsupervised learning, and reinforcement learning.

- Supervised learning

  Definition 2.1.3 *(see [12]) It is written that: "a set of training rows are provided with the correct output labels, and on the basis of these training rows, the algorithm learns to respond more accurately by comparing its output with the given label".*

- Unsupervised learning

  Definition 2.1.4 *(see [15]) It is written that: "in unsupervised learning, the training data does not have information about the output label; the learning algorithm must find patterns to give labels to the input data".*

- Reinforcement learning

  Definition 2.1.5 *(see [9] [15]) It is written that: "reinforcement learning is a part of machine learning concerned with how intelligent agents take actions in an environment in order to maximize the reinforcement signal to perform the learning process. This reinforcement signal can be positive or negative, where positive means reward and negative means punishment".*

2.2 Machine learning techniques

Support Vector Machine

Definition 2.2.1 *(see [12] [10]) It is written that: "Support Vector Machines can be used for classification as well as regression problems. It is a supervised learning algorithm. It works on the concept of margin calculation. This algorithm is used to create the optimal hyperplane in a high dimensional space that can separate the data points into different classes in the feature space. The hyperplane makes the margin between the closest points of different classes the maximum. If the input features are two, then the dimension of the hyperplane is just a line, and for three features, the hyperplane becomes a two dimensional plane".*

In [15], it is written that: "the Support Vector Machine (SVM) technique has been considered one of the most robust among machine learning techniques. It is a kernel-based technique. It is found that SVM has been used in forecasting, classification, and regression applications. SVM maps the input data x into a high-dimensional feature space by means of a nonlinear mapping and generates an optimal hyper-plane (see equation 2.2.1

$$w.x + b = 0 \tag{2.2.1}$$

for hyperplane) in this new space. This technique tends to minimize the upper bound of the generalization error. The norm of the vector w must be minimized while the margin defined

between the two classes

$$\frac{1}{||x||}$$

is maximized like the equation 2.2.2". [15]

$$min_{i=1,\dots,n}|(w,x)+b| = 1 \tag{2.2.2}$$

The SVM regression finds predicted output y from the input x is given by the equation 2.2.3

$$y^* = f(x,\alpha,\alpha^*)\sum_{i=1}^{N}(\alpha_i,\alpha_i^*)K(x_i,x_j)+b \tag{2.2.3}$$

where is kernel function K defined as: [15]

- Linear kernel
- Polynomial kernel
- Radial kernel
- Sigmoid kernel

Random forest

Definition 2.2.2 *(see [19] [8]) It is written that: "Random decision forest is an ensemble learning method for regression and classification tasks with the use of multiple decision trees and a technique called bagging (Bootstrap and Aggregation) that uses the mean prediction of the individual decision tree for predictive modeling. Random Forest has multiple decision trees as base learning models that randomly perform row sampling and feature sampling from the data, forming sample datasets for every model known as bootstrapping".*

For better understanding of the random forest regression technique, see the figure 2.1

Long short term memory (LSTM)

Definition 2.2.3 *(see [13]) It is written that: "long-short-term memory (LSTM) is a cell variant of RNN that has the ability to learn long dependencies within sequential data. The vanishing gradient effect of RNN is minimized by LSTM".*

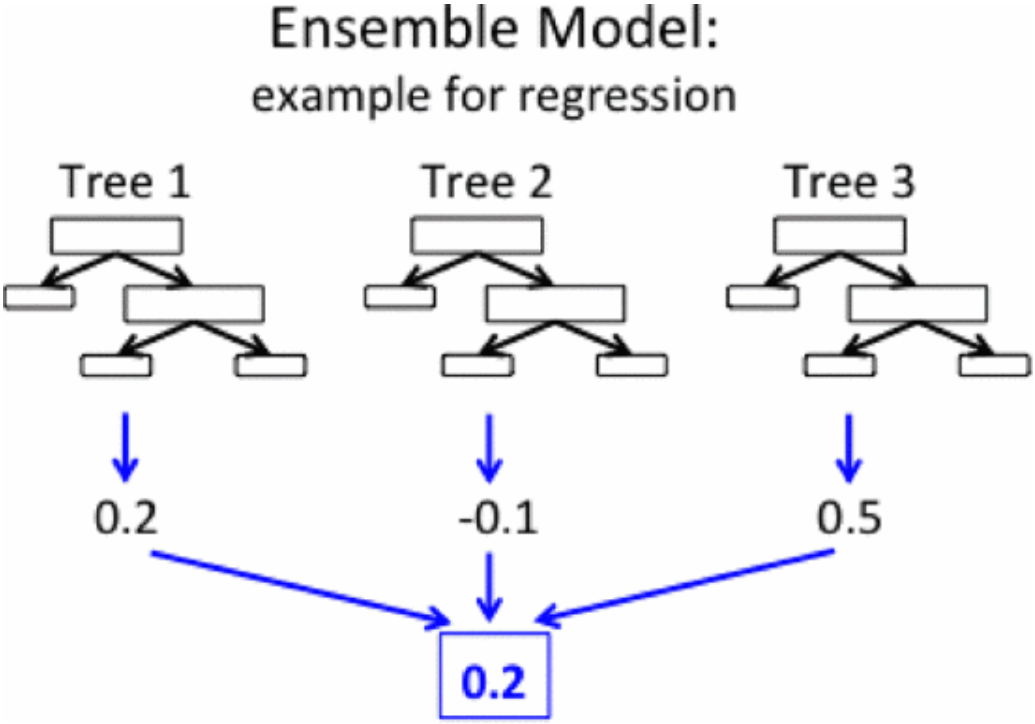


Figure 2.1: Random forest regression technique

This image adopted from [19]

Step by step of LSTM

- Cell state

  The horizontal line running through the top. See the figure 2.2

- Forget gate layer

  This is a sigmoid layer by which LSTM decides what information will go away from the cell state with the help of anequation,n which is mentioned in the figure. See figure 2.3 for a visualization of the forget gate layer with the equation of this layer. [5]

- Input gate layer with the tanh layer and updating the old cell state into the new cell state. With this combination, deciding what new information will be stored in the cell state and updating the old cell state into the new cell state with the help of an equation that is mentioned in the figure. See figure 2.4 for visualization of the input gate layer with the equation of this layer, and figure 2.5 for the equation updating the old cell state into the new cell state. [5]

- Output gate layer

  In this step, running a sigmoid layer that can decide which cell state will be output

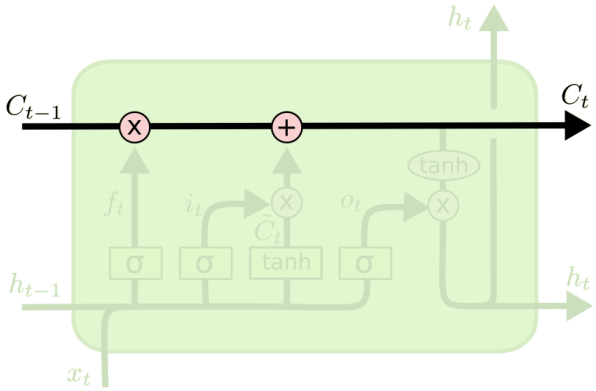
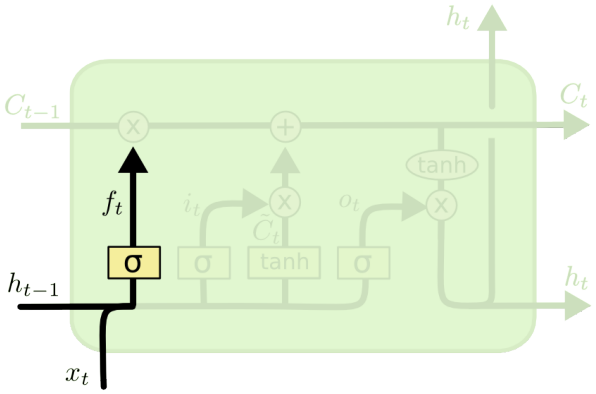


Figure 2.2: Cell state from LSTM architecture

This image adopted from [5]



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

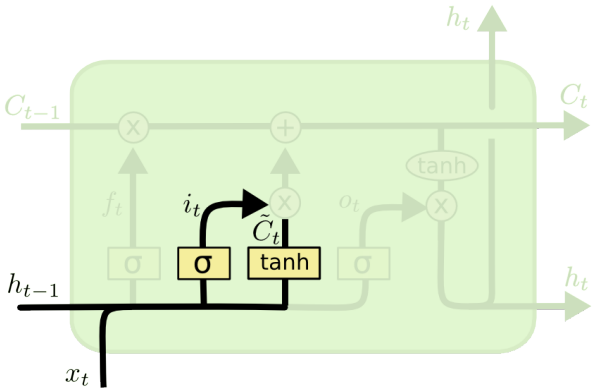
Figure 2.3: Forget gate layer from LSTM architecture

This image adopted from [5]

with the help of an equation that is mentioned in the figure and multiply this output with tanh so that it can decide only output part it wants using another equation that is also mentioned in the figure. See figure 2.6 for the visualization of this gate and these equations. [5]

Stacked long and short term memory

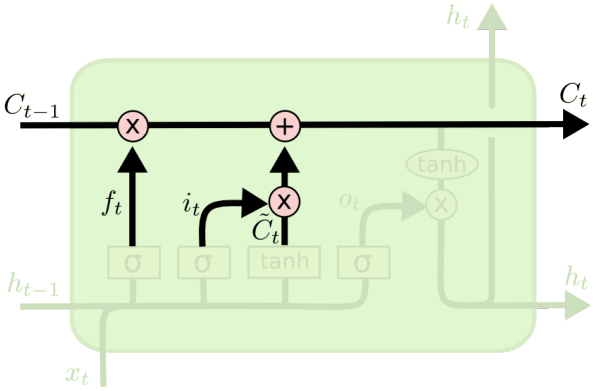
Definition 2.2.4 *(see [13]) It is written that: "stacked LSTM is composed of two or more LSTM linked one after another as hidden layers. This architecture can be applied with a higher level of representation of time-series data than only LSTM".*



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 2.4: Input gate layer from LSTM architecture

This image adopted from [5]



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 2.5: Updating part of the input gate layer from LSTM architecture

This image adopted from [5]

2.3 Evaluation measures

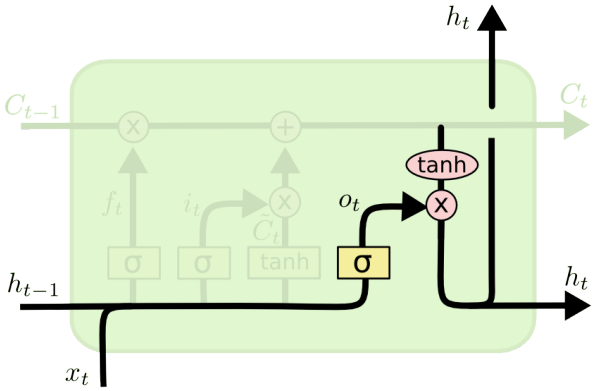
To evaluate machine learning strategies, there are some measures to compare the predicted output with the actual value. These measures are the following:

1. Mean Squared Error

   The average squared difference between the predicted and actual values is defined as the mean squared error (MSE). The equation for mean squared error is in the figure 2.3.1 [15]

$$MSE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|^2 \tag{2.3.1}$$

2. Mean Absolute Error



$$o_t = \sigma\left(W_o\ [\,h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 2.6: Output gate layer from LSTM architecture
This image adopted from [5]

The average of the distances between the estimated and observed data is defined as the mean absolute error (MAE). The equation for mean squared error is in the figure 2.3.2 [15]

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.3.2}$$

3. R Squared (R2)

   This R Squared score tells you how well your model performed. A score close to 1 means the model is the best fit. The source of this information and the formula of the *R Squared* will be found at this link: [7]

4. Adjusted R Squared (R2)

   While adding new features to the data, the R2 score starts increasing or remains constant. But it never decreases because It assumes that the variance of the data increases. But the problem is, when we add an irrelevant feature to the data, It sometimes starts increasing, which is incorrect. But Adjusted R Squared controls this incorrectness. The source of this information and the formula of the *Adjusted R Squared* will be found through this link: [7]

2.4 Dataset

For the rainfall prediction, I am going to use the dataset from Kaggle. The name of the dataset is UK Met Office Weather Data. The link to the dataset is [11]

The data originally was sourced from government institution in the UK named Met Office. [6]

In [11], it is written that: "the dataset contains variables namely year (Year in which the measurements were taken), month (Month in which the measurements were taken), tmax (mean of daily maximum temperature in °C), tmin (Mean of daily minimum temperature °C), af (Days of air frost recorded that month in days), rain (Total rainfall of the month in mm), sun (Total sunshine duration in hours), and station (Station location where measurement was recorded)".

CHAPTER 3

Experiments and results

3.1 Exploratory Data Analysis

The dataset contains 8 variables and 37049 observations. From all of the variables in the dataset, 7 variables contain float64 as a datatype, and 1 variable named *station* contains object or string type values. From the dataset, some numeric statistics are found for the numerical variables like year, tmax (maximum temperature), tmin (minimum temperature), af (air forst), rain (rainfall), and sun (sunshine). In regard to the year variable, the minimum year is 1853 and the maximum year is 2020. In regards to tmax (maximum temperature), the mean value is 12.705 °C, the min value is -12.705 °C, and the max value is 28.300 °C. In regards to tmin (minimum temperature), the mean value is 5.926 °C, the min value is -8.600 °C, and the max value is 17.000 °C. In regards to af (air frost), the mean value is 3.486 days, the min value is 0 days, and the max value is 31. In terms of rainfall, the mean value is 73.421 mm, the min value is 0 mm, and the max value is 568.800 mm. In regards to the sun (sunshine duration), the mean value is 118.486 hours, the min value is 2.800 hours, and the max value is 350.300 hours. While checking unique values for each variable of the dataset, I've found float values, *nan* values, and object or string values

(a) Observing missing values in the dataset

From the Figure 3.1, I can see that *sun (sunshine)* variable contains the highest missing values, then *af (air forst)*, *tmax (maximum temperature)*, *tmin minimum temperature*, *rain (rainfall)* respectively, contain missing values in terms of decreasing order. For the remaining variables, *year*, *month*, *station* contain missing values of 20, 20, and 0, respectively. The numerical percentage of missing values for each variable are as follows:

```
year contains:  0.053 % missing values
month contains:  0.053 % missing values
tmax (maximum temperature) contains:  4.369 % missing values
tmin (minimum temperature) contains:  4.162 % missing values
af (air forst) contains:  8.013 % missing values
rain (rainfall) contains:  4.043 % missing values
sun (sunshine) contains:  25.649 % missing values
station contains:  0.0 % missing values
```

(b) Observing correlation among variables

From the figure of *correlation plot* 3.2, I can see that the correlation value among variables is less than 0.6. But I've noticed that the correlation value between *tmax (maximum temperature)* and *tmin (minimum temperature)* variables is 0.937, which means these two variables are strongly positively correlated. Furthermore, if the correlation value between textitaf(air forst) and textittmax (maximum temperature) is -0.692, then the correlation value between textitaf(air forst) and textittmax (maximum temperature) is -0.692. and *tmin (minimum temperature)* is -0.759 meaning these two variables are also negatively correlated.

(c) Checking outliers of variables in the dataset

While checking outliers for the variables such as *tmax (maximum temperature)*, *tmin (minimum temperature)*, *af (air forst)*, *rain (rainfall)*, and *sun (sunshine)*, I've plotted a boxplot for this. From the figure of boxplot 3.3, I can see that a huge amount of outliers appear within the variable *rain (rainfall)*, and then I can see that variables such as *sun (sunshine)*, *af (air forst)*, and *tmin (minimum temperature)* have some

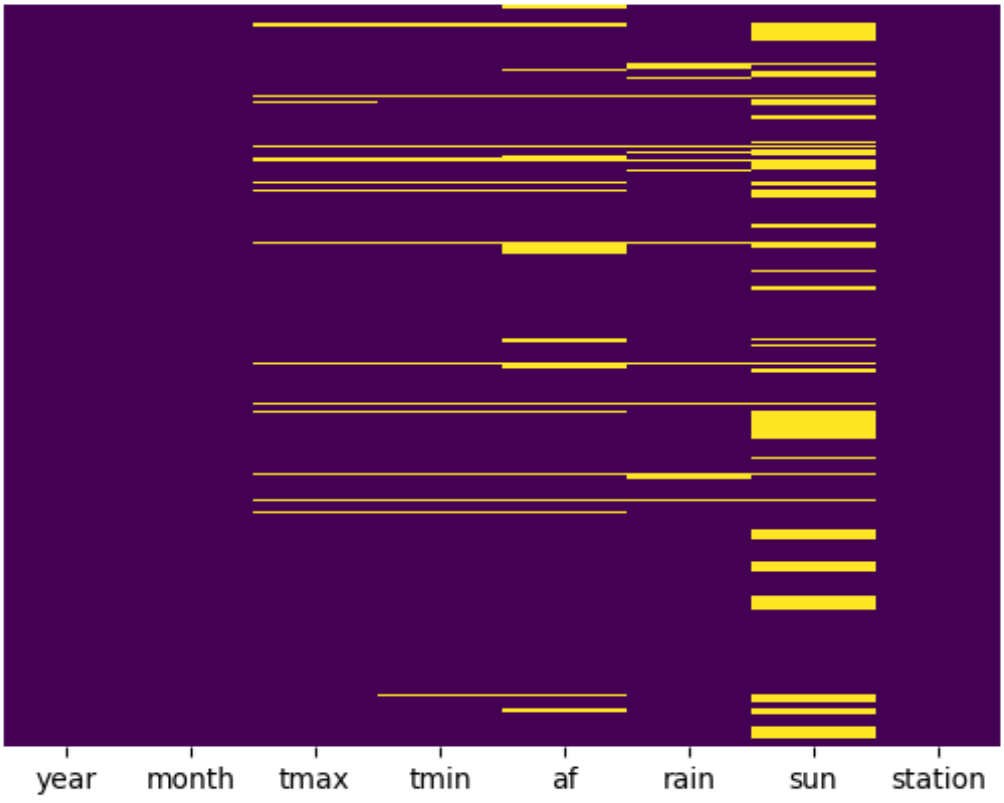


Figure 3.1: Missing values of the variables

outliers as well.

(d) Time series analysis with the target variable 'rain (rainfall)'

Before exploring, I've got some questions in my mind and these are as follows:

```
Questions:
What trends are the time series plots following?
Is there any seasonality present?
```

For the time series plot, I've divided the plotting with some periods regarding *year* variable. The periods are as follows:

i. *(1853 to 1900)*

ii. *(1901 to 1934)*

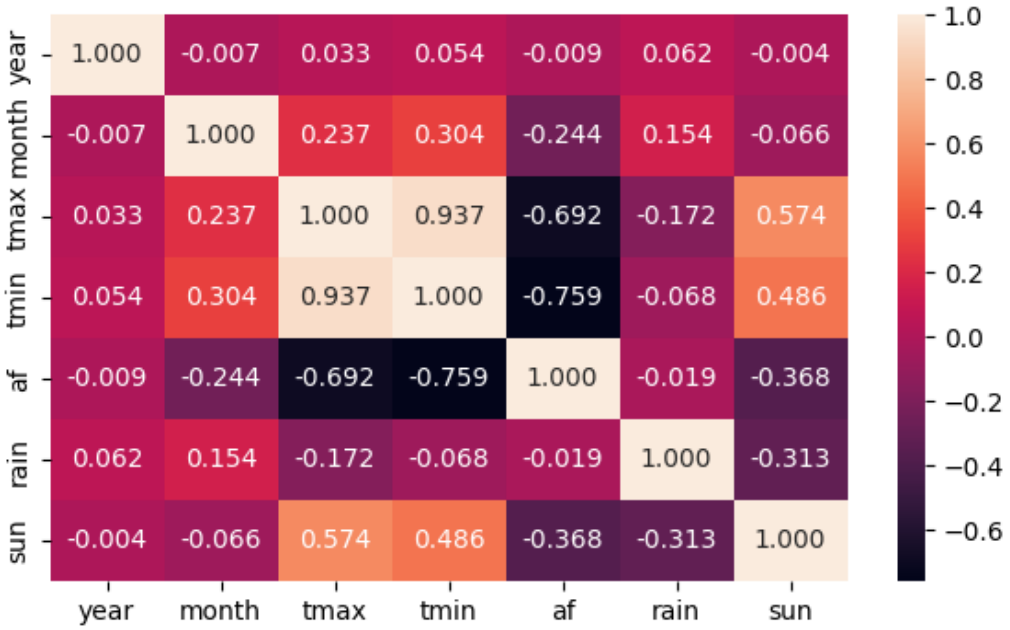


Figure 3.2: Correlation among variables

iii. *(1935 to 1974)*

iv. *(1975 to 1999)*

v. *(2000 to 2020)*

The figures 3.4, 3.5, 3.6, 3.7, and 3.8 are showing the time series plots for the periods mentioned above, which are *a*, *b*, *c*, *d*, and *e* respectively. From the figures, I can observe that there is no upward or downward trend on the plots. Each time series plot is stationary in nature. If I talk about seasonality, then for each plot, I can see that for every year with various weather stations, there are repeating data points on the plots. One other thing that I have noticed on the plots is that for the periods *a, b, and c*, the rainfall is not much with respect to the amount of 200 mm, where for the periods *d and e*, a good amount of rainfall is equal to or above 200. So, I can say that average rainfall is increasing year by year.

(e) Exploring the feature *year*

Before exploring, I've got some questions in my mind and these are as follows:

```
Questions:
How many unique values are in the year feature?
```

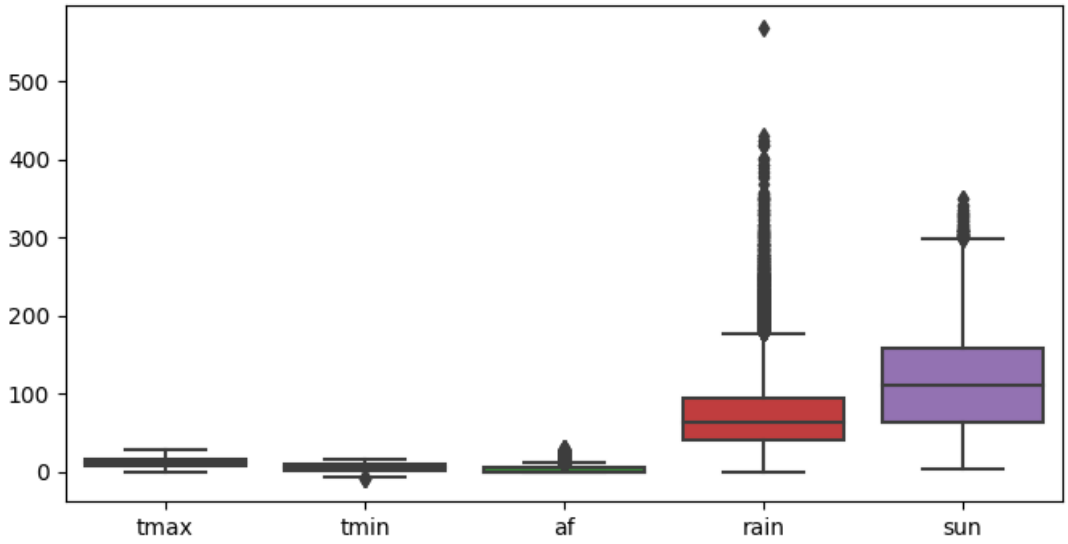


Figure 3.3: Outliers within the variables

```
How many values count for each unique year?
What is the relationship between year and station variables?
```

There are 168 unique values in the feature *year*. From Figure 3.9, it is visible that from value counts for the years, some years have value counts almost 400 and these are 1938, 1940, etc. I've found the maximum value count to be 432. While I was trying to find a relationship between features *year* and *station*. From Figure 3.10, I can see that locations from the *station* feature such as armagh, bradford,

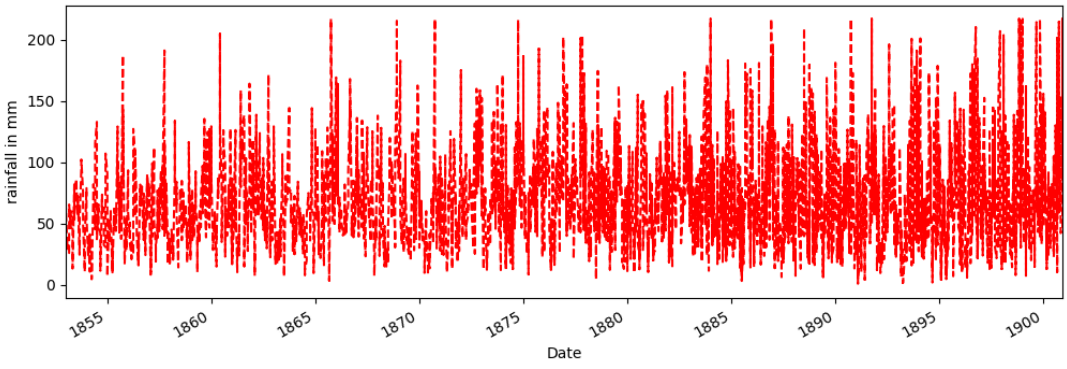


Figure 3.4: Time series plot with the period *(a)* regarding rainfall

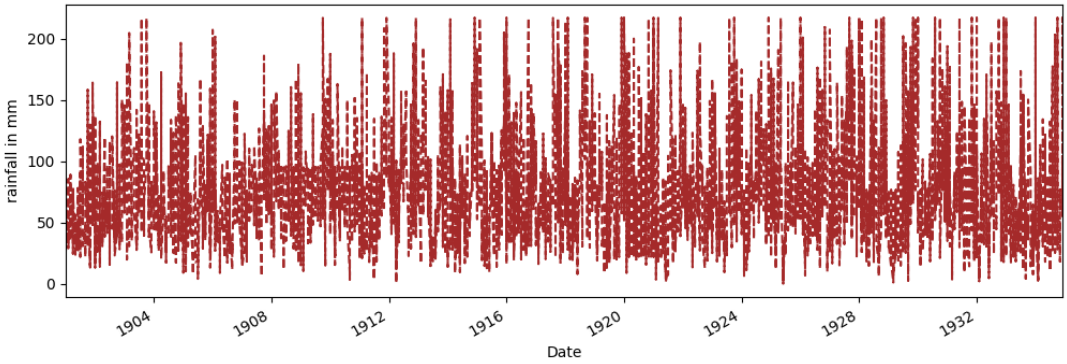


Figure 3.5: Time series plot with the period *(b)* regarding rainfall

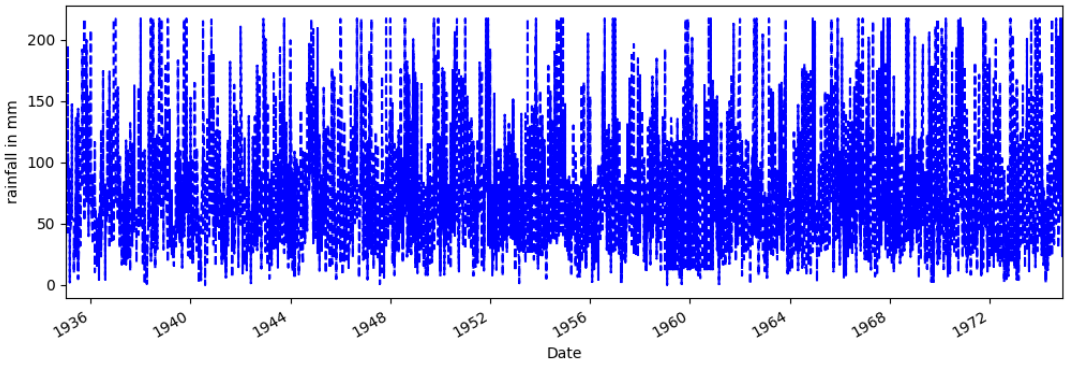


Figure 3.6: Time series plot with the period *(c)* regarding rainfall

durham, eastbourne, oxford, sheffield, southamton, stornoway, wickairport have records for maximum years, but other locations have records for fewer years.

(f) Exploring the target variable *rain (rainfall)*

Before exploring, I've got some questions in my mind and these are as follows:

```
Questions:
What do the records look like for rainfall in a month?
In which months has the rate of rainfall been highest and lowest?
For which stations was the rainfall maximum regarding the month?
For which stations was the ranfall minimal regarding the month?
```

I've plotted the histogram of rainfall for all of the records with the defined *xlabel* and *ylabel*. From the figure 3.11, I can see that for the maximum records of more

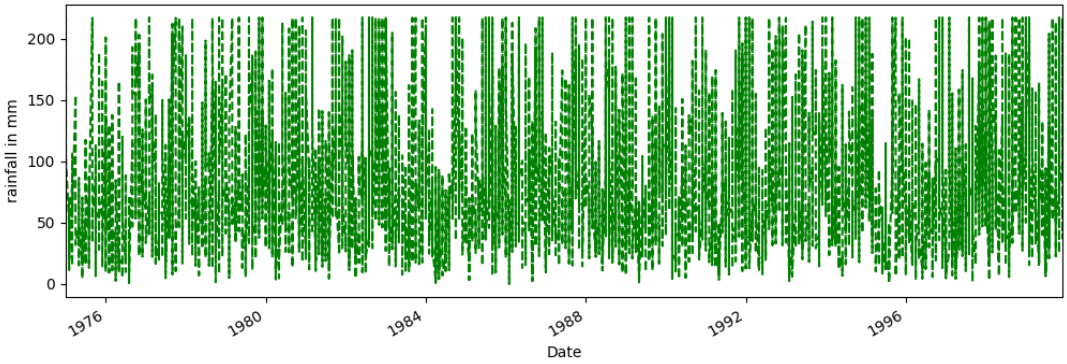


Figure 3.7: Time series plot with the period *(d)* regarding rainfall

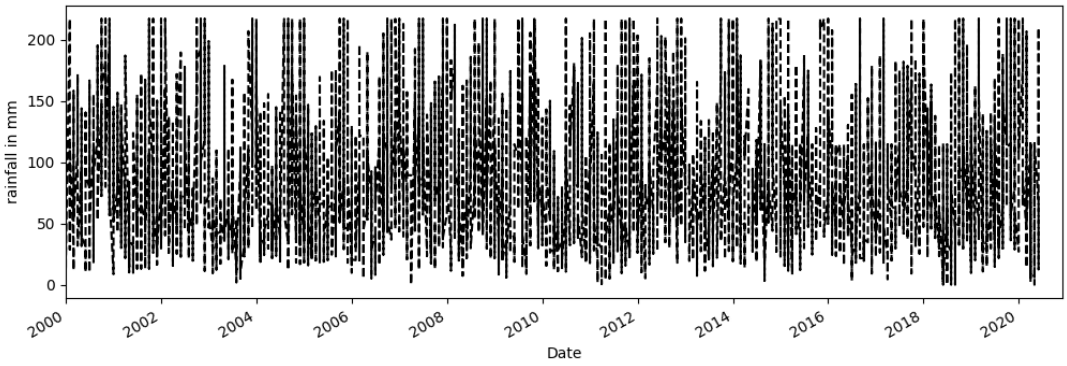


Figure 3.8: Time series plot with the period *(e)* regarding rainfall

than 8000 of the dataset, the rainfall amount ranges from approximately 45 to 75 mm. The second maximum has been found to range from approximately 30 to 80 mm, with records ranging from approximately 4200 to 7000. Moreover, I've found that minimum value of the rainfall is 0 and maximum value is 217.23.

While exploring rainfall by month, I've got the top 5 month names such as *January, February, October, November, and December* that have the highest amount of rainfall, which is 217.23 through various weather stations in the UK. On the other hand, I've got the month names such as *February, April, June, and August* which have the lowest amount of rainfall, which is 0 through various weather stations in the UK. Here's what's observed from the figure 3.12 where weather stations are in the *x-axis* and counts are in the *y-axis*, showing how many times each station has met

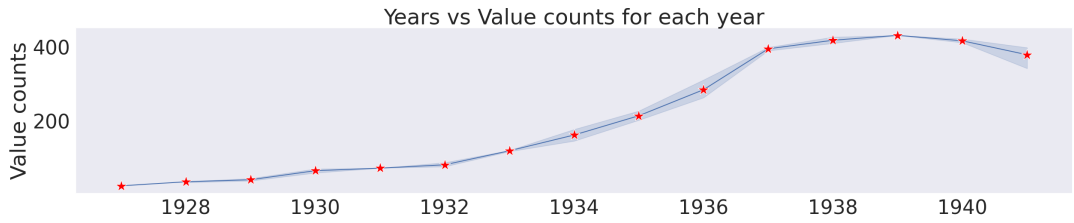


Figure 3.9: Relation between year and location

the highest rainfall in a month for different years. And then, from the figure 3.13 where this is showing how many times each station has met the lowest rainfall in a month for different years.

(g) Exploring the feature variable *station*

Before exploring, I've got some questions in my mind and these are as follows:

```
Questions:
How many unique weather stations are in the dataset?
What are the percentages of stations used for different dates?
What are the stations that have had rainfall less equal to 10 mm?
What are the stations that have had rainfall greater equal to 200 mm

What was the condition of the rainfall after the year 2000?
What was the condition of the rainfall from 1975 to 1999?
What was the condition of rainfall before the year 1975?
```

The number of unique weather stations in the dataset is 36. Presenting the figure 3.14 for observing the percentages of participation of weather stations for different dates (*Date*) in the dataset. I can see that from all of the weather stations, only 7 stations have higher percentages, which range from 4 to 5.4%. 35 weather stations are found that have rainfall less than or equal to 10, and 27 weather stations are found that have rainfall greater than or equal to 200.

While I've started to explore the rainfall of the different weather stations in the UK for the years after 2000, it has been found that the average rainfall is 77.702,

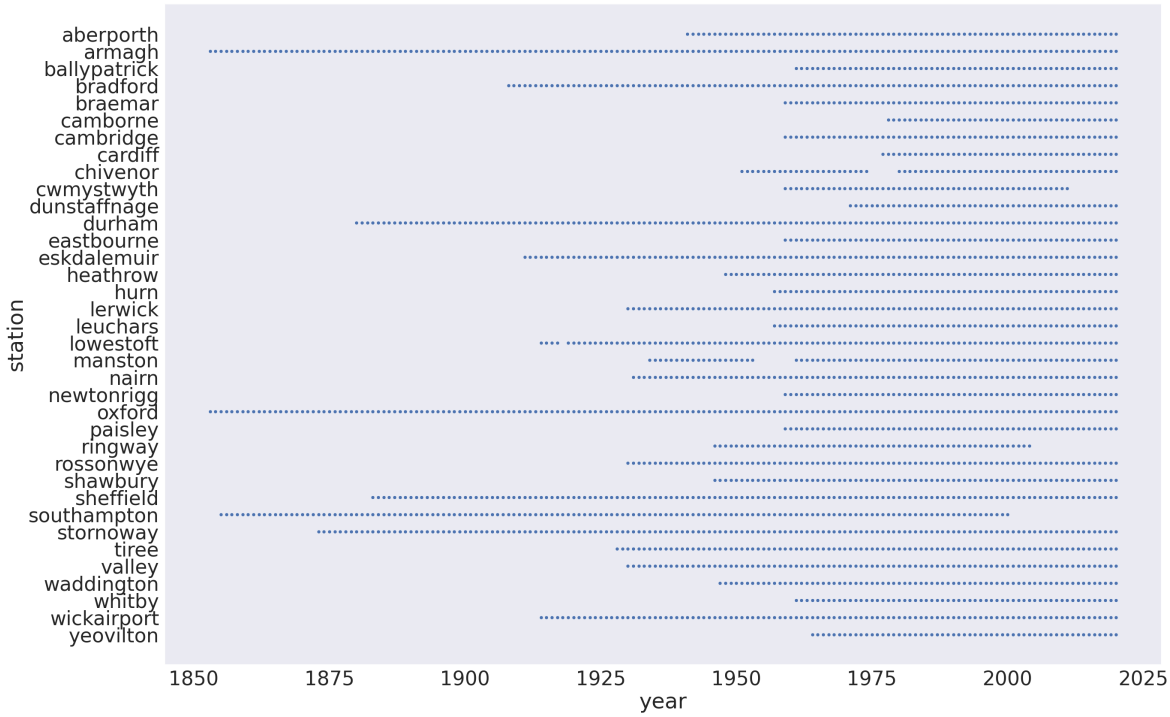


Figure 3.10: Yearś value counts for weather stations

the standard deviation is 47.150, the minimum rainfall is 0.200, and the maximum rainfall is 217.237. Then, for the years between 1975 and 1999, it has been found that the average rainfall is 73.605, the standard deviation is 48.168, the minimum rainfall is 0, and the maximum rainfall is 217.237. After that, for the year before 1975, it has been found that the average rainfall is 69.889, the standard deviation is 41.923, the minimum rainfall is 0, and the maximum rainfall is 217.237. From this information, I can state that the average rainfall and standard deviations for these different periods of the year are increasing year by year, where the minimum rainfall is almost zero and the maximum rainfall of 217.237 remains the same.

(h) Exploring the feature variables *tmax (maximum temperature), tmin (minimum temperature), af (air forst), and sun (sunshine)* using histogram

From the figure 3.15 for the feature variable *tmax (maximum temperature for months)*, I can see that most of the records fall in the value range approximately from 7 to

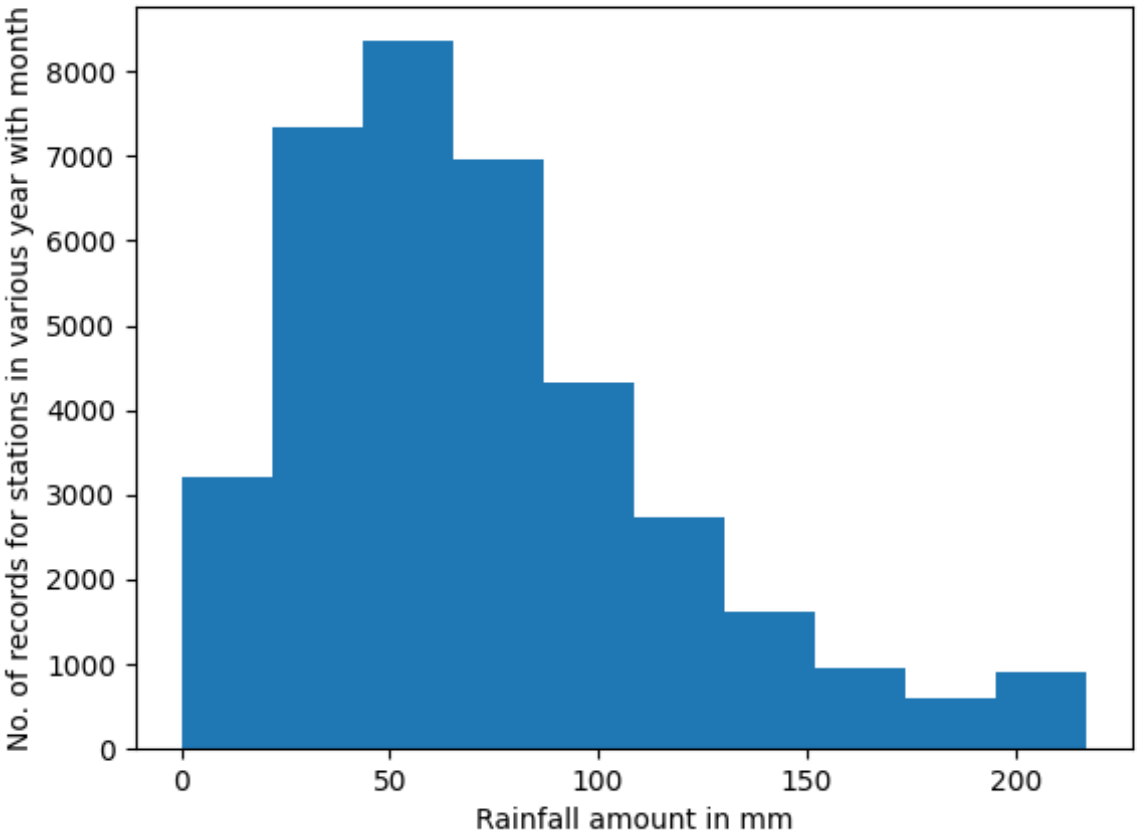


Figure 3.11: Histogram regarding the target variable *rain (rainfall)* with all the records

22 temperatures and are following the *Bell Curve (Gaussian distribution)*. In the case of the figure 3.16 for the feature variable *tmin ((minimum temperature for months)*, I can see that most of the records fall in the value range approximately from 2 to 12, and it is following the *Bell Curve (Gaussian distribution)* too. Then, from the figure 3.17 for the feature variable *af (amount of air forst for months)*, I can see that most of the records fall in the value range approximately from 0 to 2.4 air forst, and it is skewly distributed. And finally, from the figure 3.18 for the feature variable *sun (amount of sunshine for months)*, I can see that most of the records fall in the value range approximately from 48 to 180°C and are following the *Bell Curve (Gaussian distribution)*.

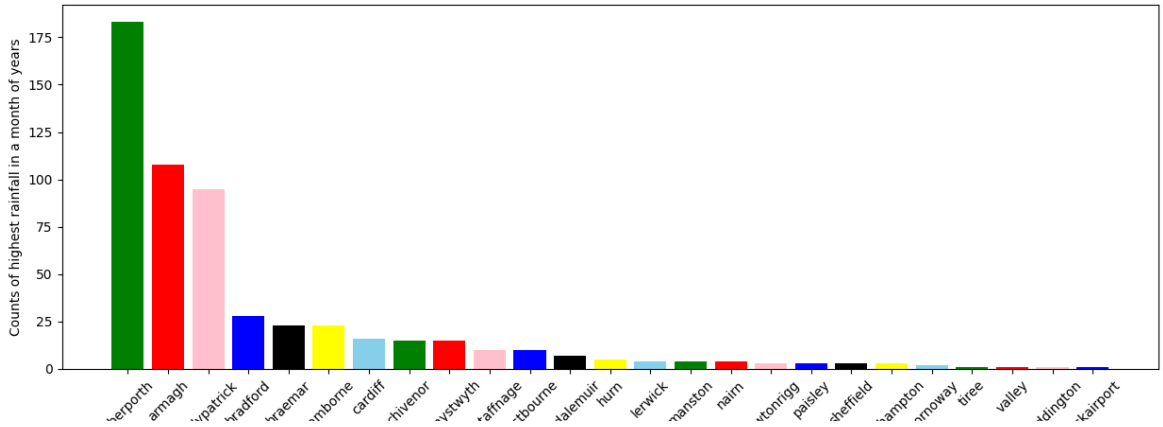


Figure 3.12: Barplot regarding weather station names on highest rainfall

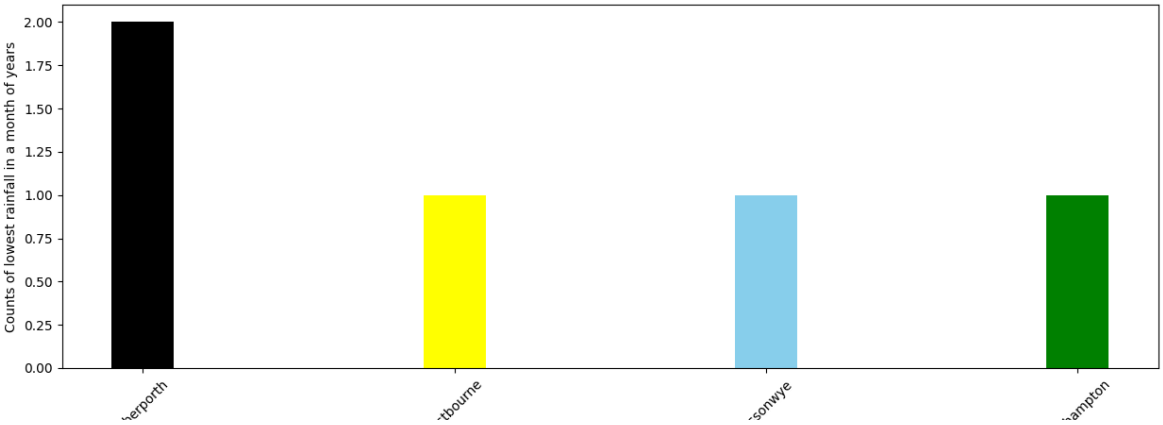


Figure 3.13: Barplot regarding weather station names on lowest rainfall

Data pre-processing

(a) Handling missing records

In data pre-processing, from the very beginning, I handled missing records of the variables in the dataset. Regarding missing records for the features *year* and *month*, I've used *dropna()* method to handle missing records because records without dates do not make sense, so it is better to delete these records. With the rest of the variables except *station* I've used *ffill and bfill* methods to handle missing records because for these features, I needed to fill the missing values depending on previous values or next values (records of the previous month more or less will be similar with the present month, for example, the mean temperature of the month).

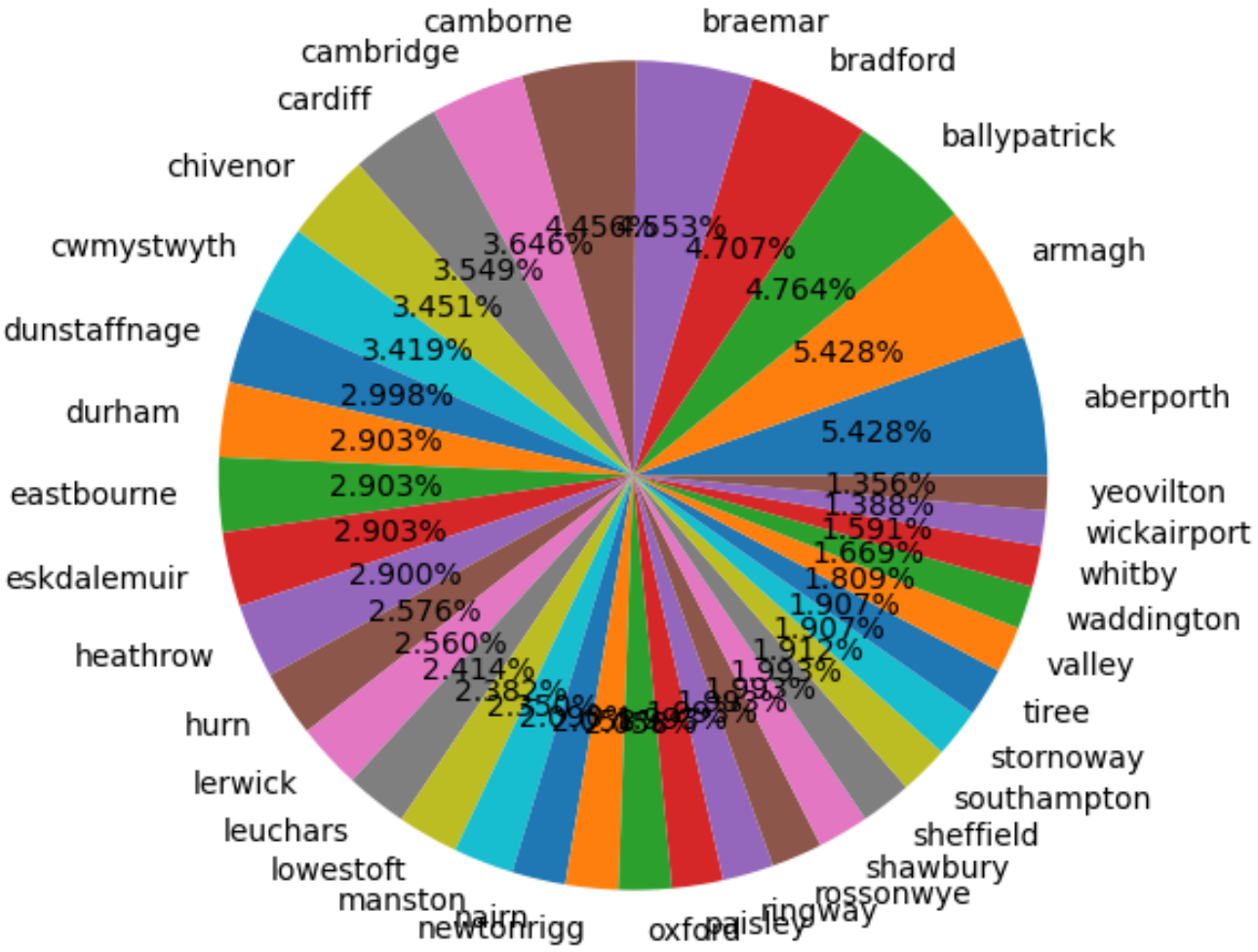


Figure 3.14: Pie chart is showing percentages of weather stations

(b) Decision of variable's correlation

As mentioned in the correlation exploratory part above, some correlation values among some specific variables have been found to be near 1. But these variables, such *af (air forst)*, *tmax (maximum temperature)*, and *tmin (minimum temperature)* are really different variables in a real sense. So, regarding my prediction task, I can't eliminate any of these variables. Consequently, I am going to proceed with all the variables in the dataset.

(c) Handling outliers

As *tmax (maximum temperature)* (See Figure 3.19), *tmin (minimum temperature)* (See Figure 3.20), *rain (rainfall)* (See Figure 3.21) variables are normally distributed, and *sun (sunshine)* (See Figure 3.23) is nearly normally distributed. For this, I've

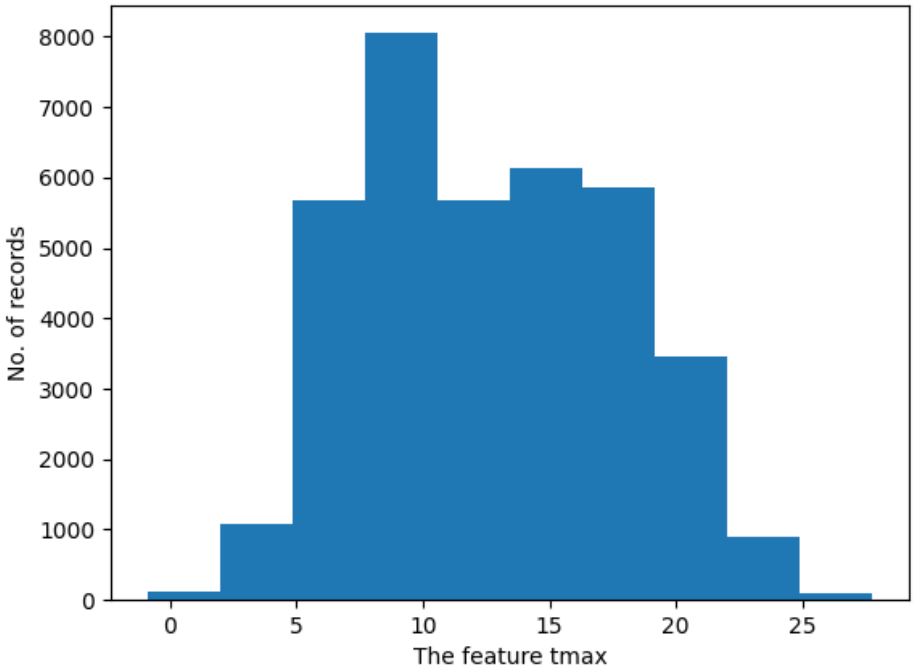


Figure 3.15: Histogram regarding the feature *tmax (maximum temperature)*

handled outliers of these variables using the standard deviation (SD) approach, where I've counted upper limits +3 from the means of these variables for the upper boundary value and lower limits -3 from the means of these variables for the lower boundary value, then I've used the capping technique for setting the values in between the lower boundary value and the upper boundary value for the outliers of variables. On the other hand, *af (air forst)* (See Figure 3.22) is a skewed variable in distribution, so, I've handled outliers of this using the interquartile range (IQR) approach including extreme outliers.

(d) Converting categorical variables to numeric

As I've seen, there is only one variable, which is *station* containing categorical values or objects as string values. As machine learning algorithms do not understand categorical values, I need to convert the categorical variable *station* to numeric. For this, I've used *OneHotEncoding* technique because the values are the nominal categorical values.

(e) Finishing touch of pre-processed data

I've saved two versions of pre-processed data. For the first version, I've just saved

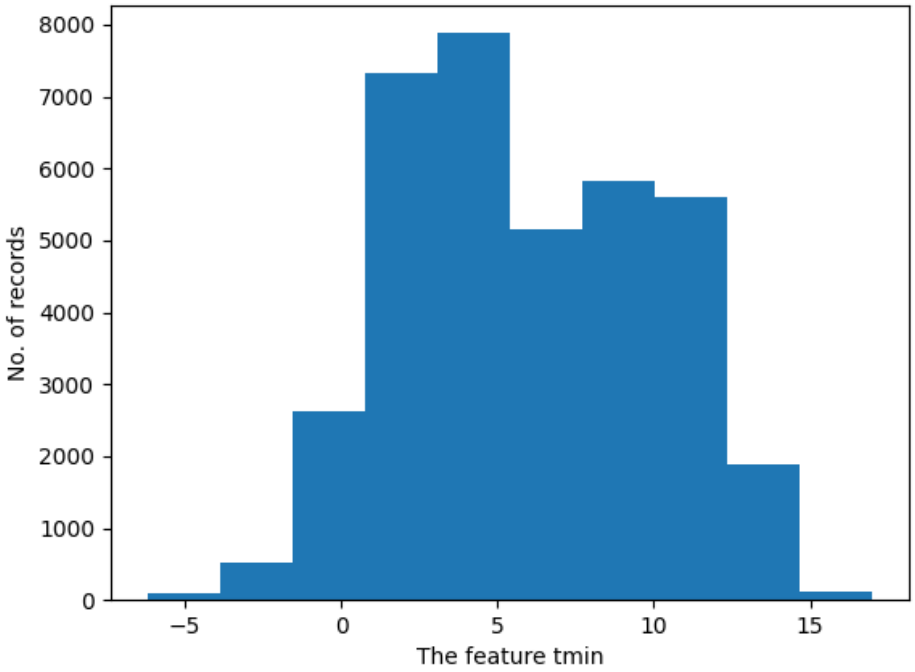


Figure 3.16: Histogram regarding the feature *tmin (minimum temperature)*

the dataset as a csv file with the conversion of the categorical variable *station* to numeric and other pre-processed variables too. On the other hand, for the second version, I've saved the dataset as csv file with all of the pre-processed variables but no conversion of categorical variable *station* to numeric.

3.2 Explanation of models for prediction

As I mentioned before, I will use support vector machine (SVM), Random Forest, long short term memory (LSTM), and stacked long short term memory (Stacked LSTM) for the task of rainfall prediction. The ingredients I've used are the programming language *python*, Python and Sklearn libraries for model definition. The whole process for each of these models is describing as follows:

(a) In case of support vector machines (SVM) and random forest (RF)

From the pre-processed dataset of version one (as mentioned above on the finish-

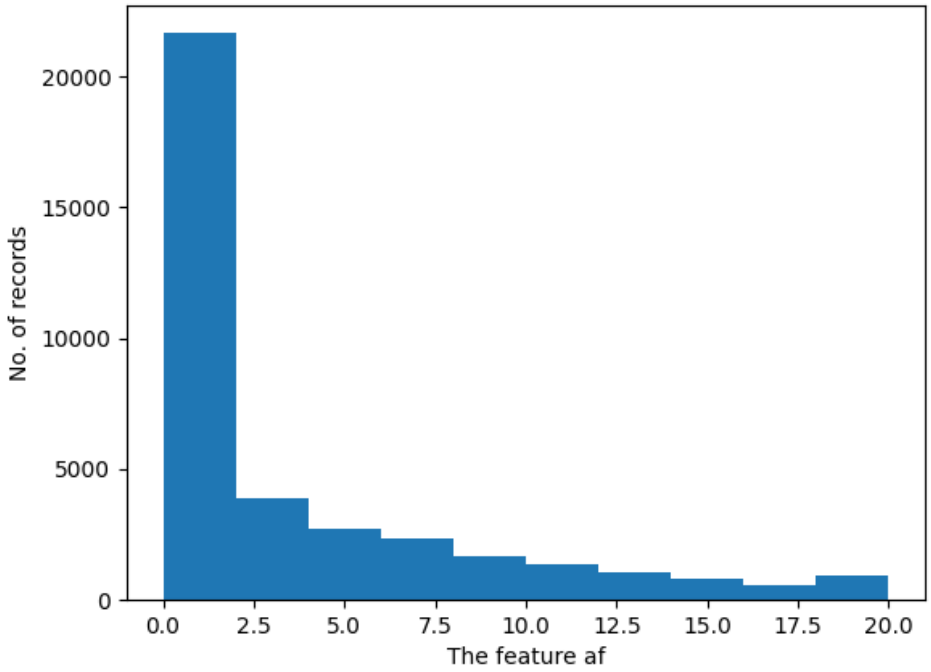


Figure 3.17: Histogram regarding the feature *af (air forst)*

ing touch of pre-processed data), *year* and *month* variables are dropped. After that, this dataset is separated in into feature variables and target or response variables *rain*. Then, I've split the dataset into a training dataset (67%) and a testing dataset (33%). At the next stage, I scaled the training and testing datasets excluding the target variable using *StandardScaler()*.

As a first model, support vector regressor (SVR) of SVM is used with parameters *kernel='rbf', gamma=1, epsilon=0.01* to train the training dataset.

As a second model, random forest (RF) is used with no parameter tuning.

(b) In case of long short term memory (LSTM) and stacked long short term memory (Stacked LSTM)
Here I've used the pre-processed dataset version two (as mentioned above on the finishing touch of pre-processed data). From this pre-processed dataset, I've picked two resampled datasets based on *armagh* and *oxford* weather stations as new indivisual datasets from the feature *station* in the dataset.

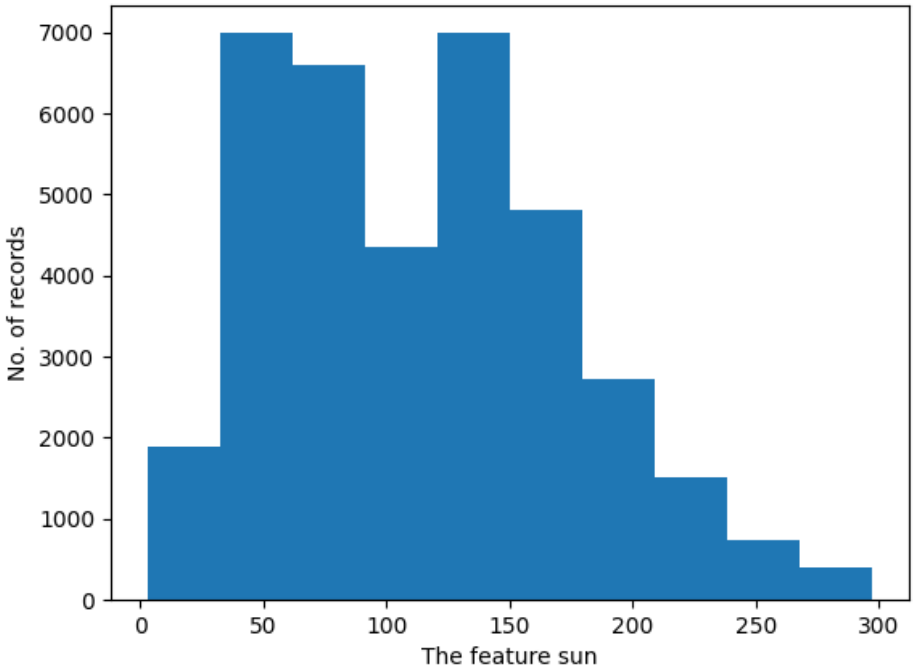


Figure 3.18: Histogram regarding the feature *sun (sunshine)*

For these datasets, based on *armagh* weather station and *oxford* weather station, I've just combined *year* and **month** features and made them to *Datetime* feature as type datetime. From the figures 3.24 (armagh) and 3.25 (oxford), I can see that the *time siries plots* have been plotted by me; for the horizontal axis, there's *Datetime* feature, and for the vertical axis, there's *rain* target or response variable. Regarding *armagh* station, the plot shows that the range of rainfall varies from approximately 40 mm to 150 mm in maximum cases for the dates (year and month) with lots of ups and downs. And regarding *oxford* station, the plot shows that the range of rainfall varies from approximately 20 mm to 90 mm in maximum cases for the dates (year and month) with lots of ups and downs.

For the dataset based on *armagh* weather station, I've followed the further processes as follows;

The features and the target labels are created for the time series prediction variable

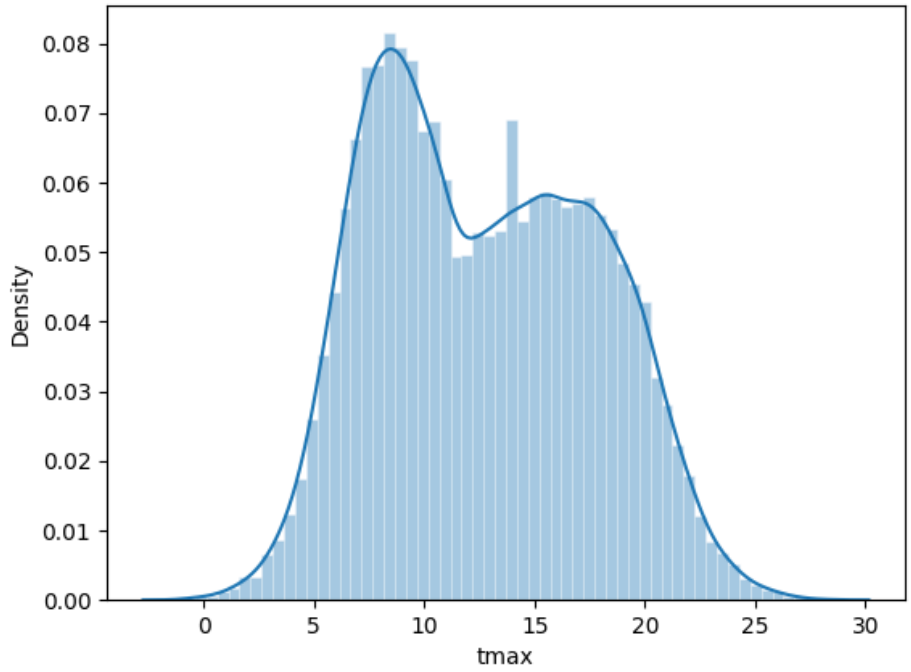


Figure 3.19: Distribution of variable *tmax (maximum temperature)* using histplot

rain (rainfall) with index *Date* of the dataset. The shape of the dataset is *(1910, 100, 1)* for time series features and *(1910,)* where the *timestemps* has been set to 100. Then I've split this dataset into training data (*record range 0 to 1500*), validation data (*record range 1500 to 1800*), and testing data (*record range 1800 to 1910*).

For long short term memory (LSTM) , I've used 1 hidden layer, 32 neurons, *tanh* activation function, and 1 dense layer. While compiling this model, I've set the parameters such as optimizer as *adam* with learning rate equal to 0.0001, loss as *mse*, and metrics as *rmse*. Finally, while fitting the model, I've set the parameters like training features, training labels, validation data, batch size as 5, epocs as 100, and I've saved the best compiled model through epocs. For these hyperparameter tunings, I've taken the idea from there [18].

For the stacked long short term memory (Stacked LSTM) , I've changed the value of *timestemps* to 50 while splitting the dataset into training, validation, and testing has changed regarding this *timestemps or timelags*. I've used 3 hidden layers with

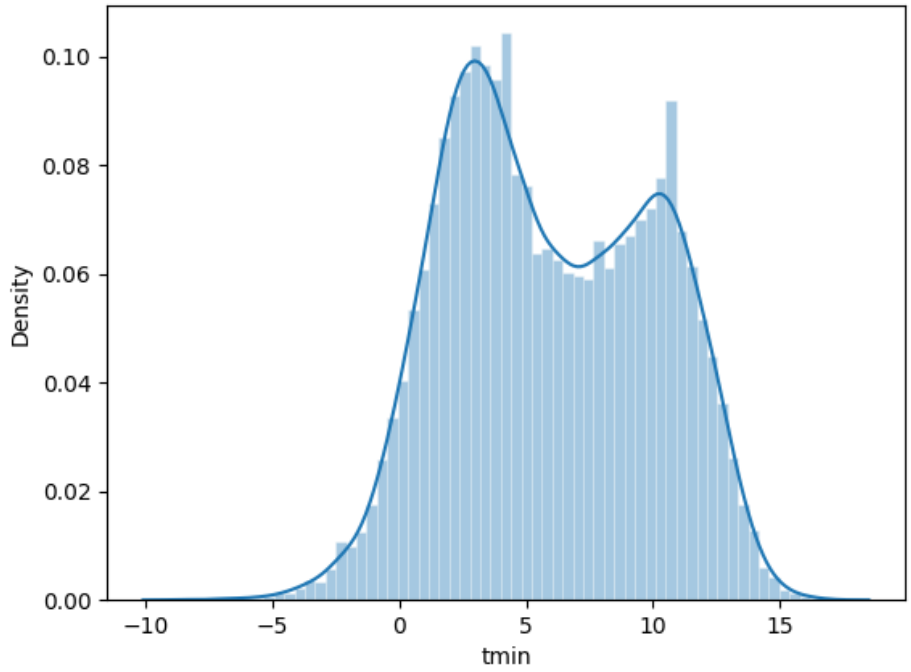


Figure 3.20: Distribution of variable *tmin (minimum temperature)* using histplot

a dropout value of 0.3, 32 neurons for each hidden layer, *relu* activation function for each hidden layer, and 1 dense layer. While compiling this model, I've set the parameters such as optimizer as *adam* with learning rate equal to 0.0001, loss as *mse*, and metrics as *rmse*. Finally, while fitting the model, I've set the parameters like training features, training labels, validation data, batch size as 5, epocs as 100, and I've saved the best compiled model through epocs. For these hyperparameter tunings, I've taken the idea from there [18].

For the dataset based on *oxford* weather station, I've followed the further processes as follows:

The features and target labels are created for the time series prediction variable *rain* with index *Date* of the dataset. The shape of the dataset is *(1910, 100, 1)* for time series features and *(1910,)* where the *timestemps* has been set to 100. Then I've split this dataset into training data (*record range 0 to 1500*), validation data (*record range 1500 to 1800*), and testing data (*record range 1800 to 1910*).

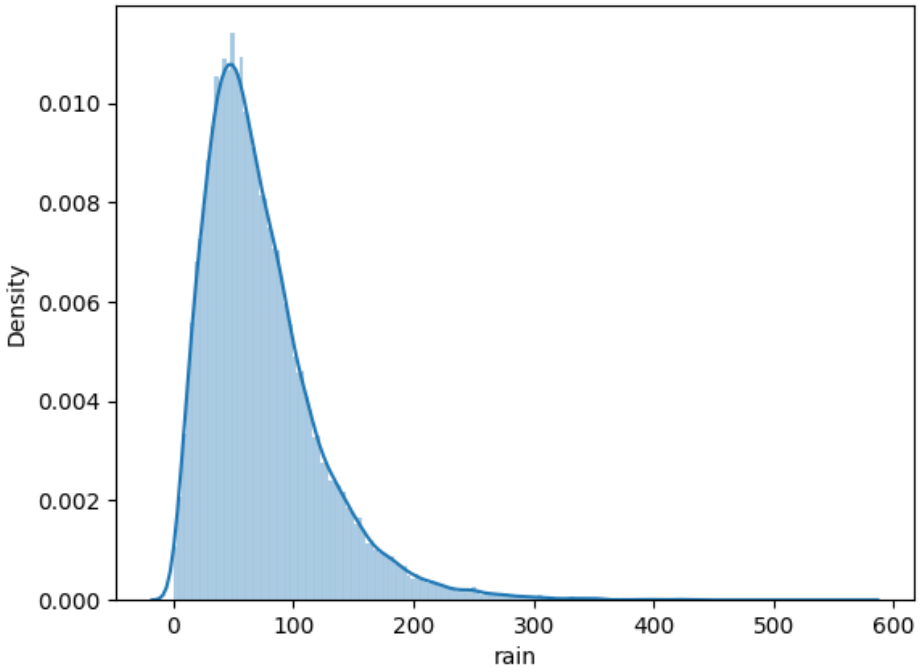


Figure 3.21: Distribution of variable *rain (rainfall)* using histplot

For long short term memory (LSTM), everything remains the same as it was for *armagh* weather station, which was mentioned above.

For the stacked long short term memory (Stacked LSTM), everything remains the same as it was for *armagh* weather station, which was mentioned above.

3.3 Results

The support vector regressor (SVR) was implemented, and the results obtained are discussed in Table 3.1 shows the performance measures of the SVR model on the testing dataset. The results from the table show that the value of each metric does not define a good model or a good prediction. The figure 3.26 shows the prediction distribution of 1000 values, and the figure 3.27 shows the actual distribution of 1000 values. From these figures, It is clearly visible that there are differences between actual values and

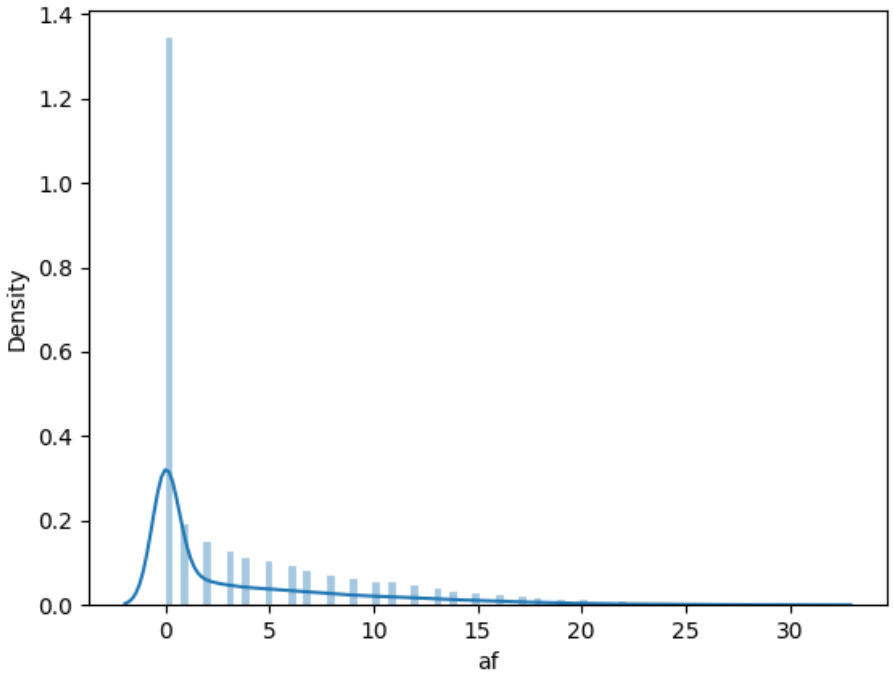


Figure 3.22: Distribution of variable *af (air forst)* using histplot

prediction values. From the prediction distribution plot, It shows that prediction values vary from 50 to 100 in the maximum case. But the actual distribution plot shows that actual values also vary from 50 to 100, where a good amount of values lie on less than 50 and some amount of values lie on greater than 100. So, this is not good prediction by the model, as expected.

Support Vector Regressor's results	
Name of measures	**Values**
Mean Absolute Error (MAE)	28.786
Mean Squard Error (MSE)	1481.058
R-Squared score (R2)	0.270
Adjusted R-Squared score	0.268

Table 3.1: Performance measures of SVR model on the testing dataset

The random forest regressor (RFR) was implemented, and the results obtained are discussed in Table 3.2 shows the performance measures of the RFR model on the testing

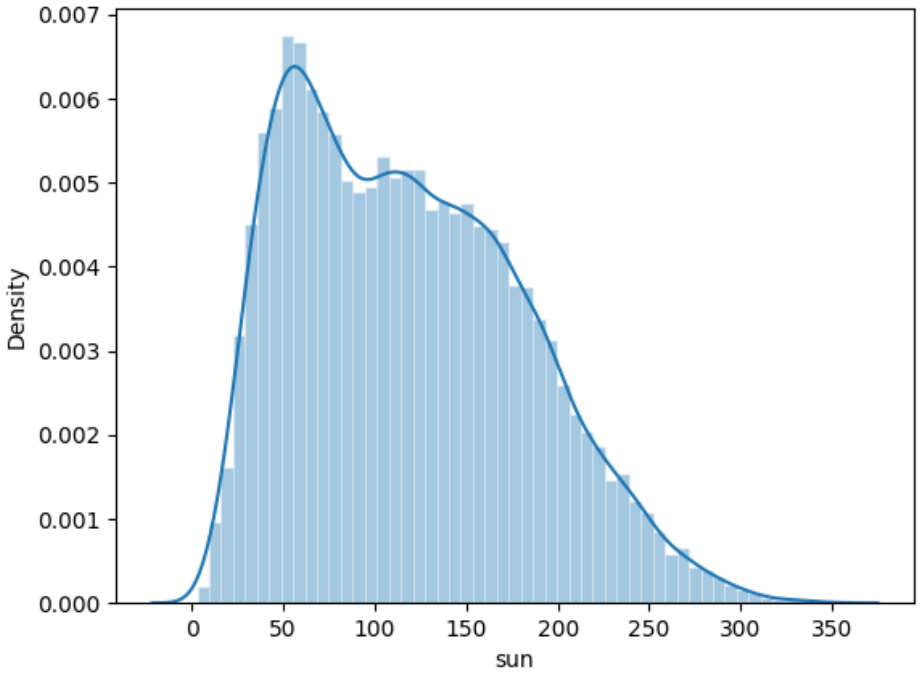


Figure 3.23: Distribution of variable *sun (sunshine)* using histplot

dataset. The results from the table show that the value of each metric does not define a good model or a good prediction. The figure 3.28 shows the prediction distribution of 2000 values, and the figure 3.29 shows the actual distribution of 1000 values. From these figures, It is clearly visible that there are also differences between actual values and prediction values. From the prediction distribution plot, It shows that prediction values vary approximately from 20 to 100 in the maximum case. But the actual distribution plot shows that actual values also vary from 20 to 100, with some values lying on less than 20 and a good number of values on greater than 100. So, this is also not good prediction by the model, as expected.

The tabulated results of support vector regressor (SVR) and Random Forest Regressor showed that the RFR model gives better prediction and performance in comparison to SVR. The model demonstrates the RF model can be chosen for accurate prediction model compared to SVR.

The long short term memory (LSTM) for the datasets based on *armagh* and *oxford* which

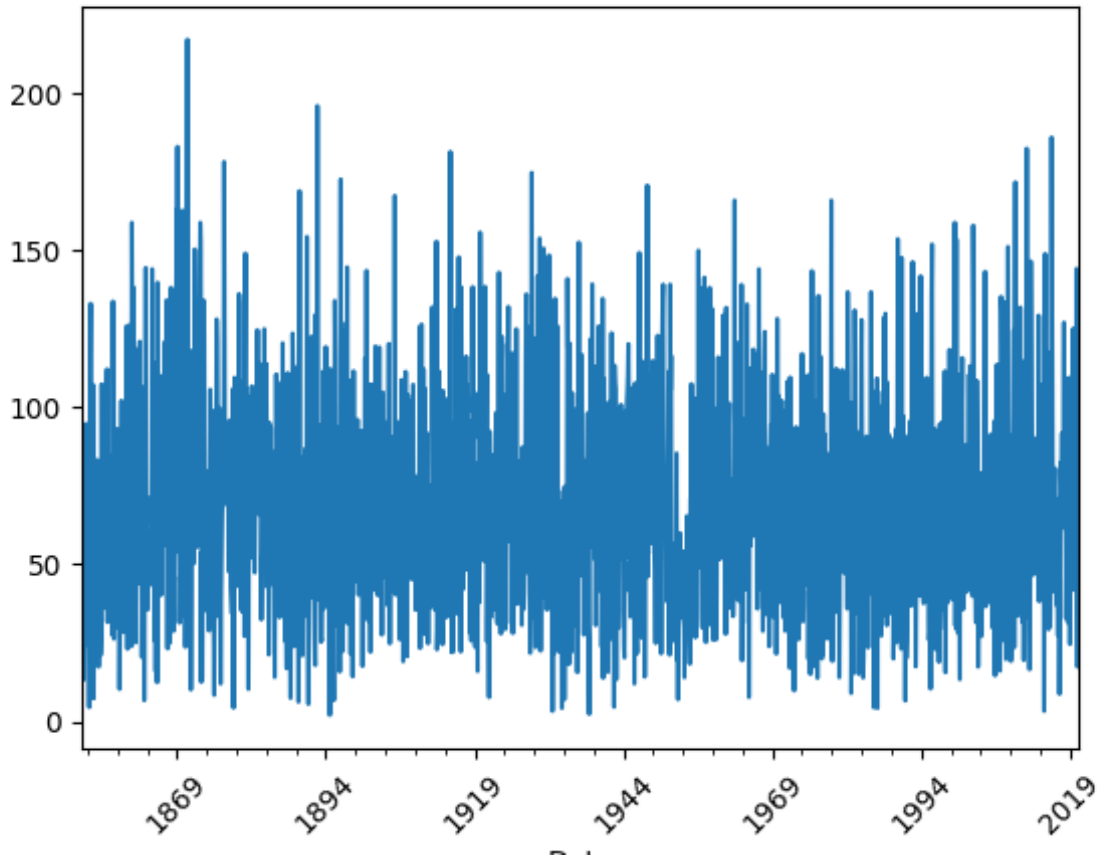


Figure 3.24: Time siries plot of the resampled dataset based on *armagh* weather station

were implemented in the finishing touch part above, and the results obtained are discussed here. The figures 3.30 (*armagh*) and the figure 3.31 (*oxford*) show the plot of predicted values and actual values based on the splitted testing dataset for these two weather station datasets. In the figure, the yellow color plot represents actual values. From the figure, It is clearly visible that there are huge differences between the actual plot and the prediction plot for both these weather station datasets. From the figure, I can see that some predicted values are close to actual values, but most of the predicted values are far away from the actual values. Maximum predicted values fall within the range of approximately 67 to 68 for the *armagh* weather station dataset. Where maximum predicted values fall within the values approximately 54 to 55 for the *oxford* weather station dataset So, the LSTM model does not perform as well as expected with these two weather station datasets.

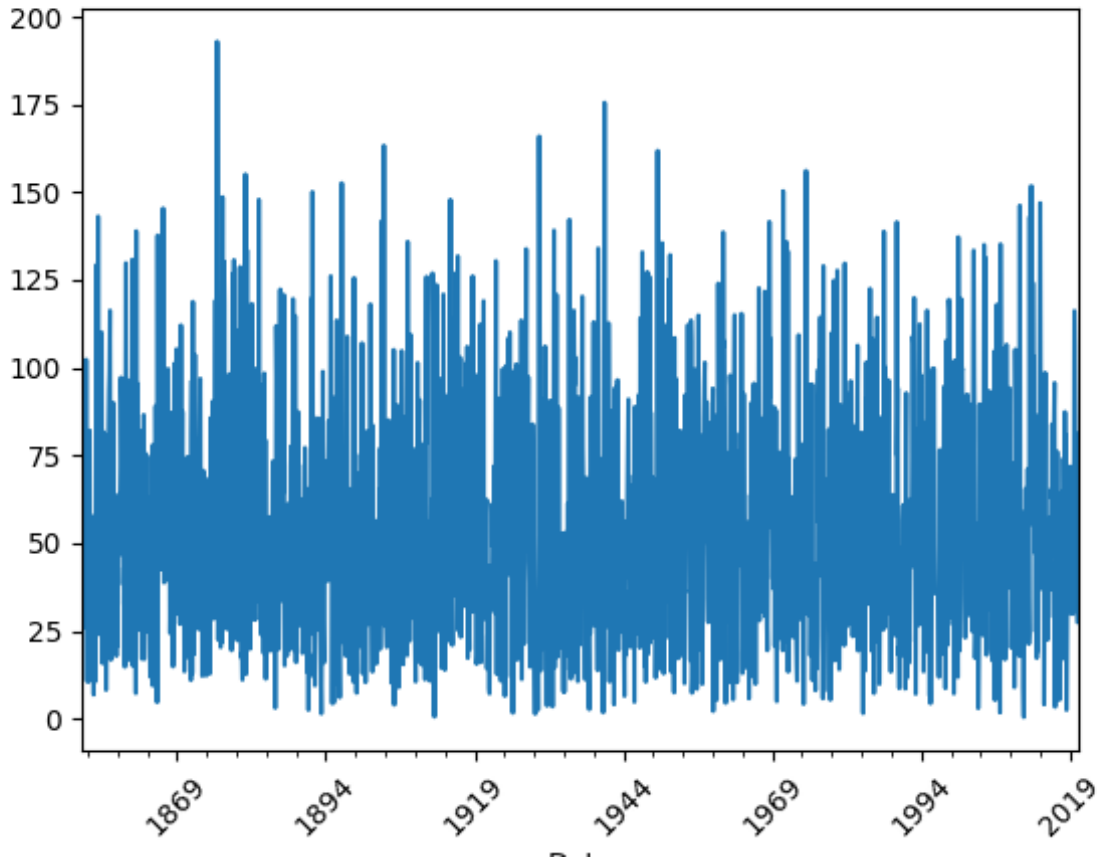


Figure 3.25: Time siries plot of the resampled dataset based on *oxford* weather station

The performance metrics for both these weather station datasets using long short term memory (LSTM) have been shown on the tables 3.7 (*armagh*) and 3.8 (*oxford*). The results are also not satisfactory, as expected for these weather station datasets.

The Stacked long short term memory (Stacked LSTM) for the dataset based on *armagh* and *oxford* were implemented, which was mentioned above in the finishing touch part, and the results obtained are discussed here. The tables 3.5 (*armagh*) and 3.6 (*oxford*), show the first 10 values from the predicted and actual values for these two weather station datasets. The figures 3.32 and 3.33 show the plot of predicted values and actual values based on the splitted testing dataset for these two weather station datasets. In the figure, the yellow color plot represents actual values. From the figures, It is clearly visible that there are huge differences between the actual plot and the prediction plot. From the figures, I can see that some predicted values are close to actual values, but

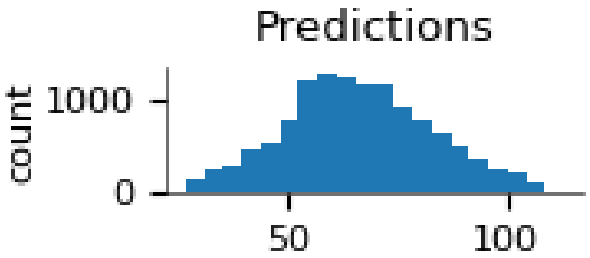


Figure 3.26: SVR prediction distribution of testing dataset

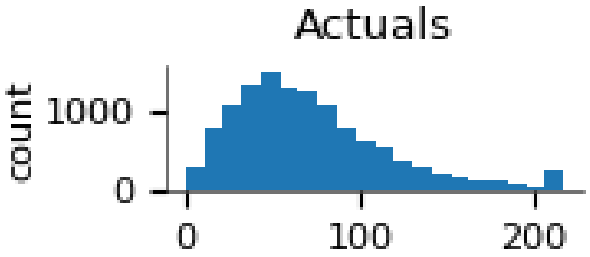


Figure 3.27: SVR actual distribution of testing dataset

most of the predicted values are far away from the actual values. So, the stacked LSTM model does not perform as well as expected for these two weather station datasets, but it performs better than the LSTM model.

The performance metrics for both these weather station datasets using stacked long short term memory (Stacked LSTM) have been shown on the tables 3.7 (*armagh*) and 3.8 (*oxford*). The results are also not satisfactory, as expected for these weather station datasets.

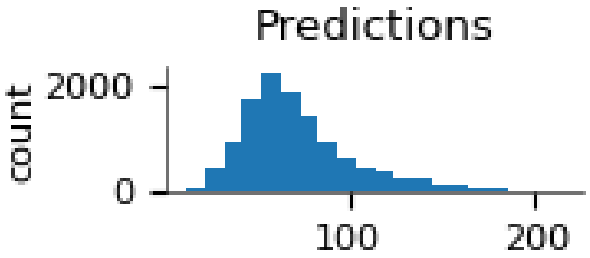


Figure 3.28: RFR prediction distribution of testing dataset

Random Forest Regressor's results	
Name of measures	**Values**
Mean Absolute Error (MAE)	27.486
Mean Squard Error (MSE)	1295.877
R-Squared score (R2)	0.361
Adjusted R-Squared score	0.359

Table 3.2: Performance measures of RFR model on the testing dataset

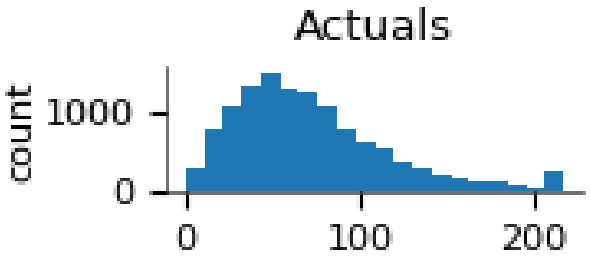


Figure 3.29: RFR actual distribution of testing dataset

In the above, I wrote the results of both long-short-term memory (LSTM) and stacked long-short-term memory (Stacked LSTM) models while using datasets based on *armagh* and *oxford* weather stations (mentioned there in the finishing touch part). For both datasets, the LSTM model performed well for the resampled dataset based on *oxford* weather station, and Stacked LSTM also performed well for the resampled dataset based on *oxford* in terms of the evaluation metrics MAE and RMSE.

The model's programs are run on the Windows operating system, on Google Colab, on an HP laptop with an Intel Core i5 processor and 4GB of RAM. The runtime for the LSTM and Stacked LSTM models is approximately 40 minutes.

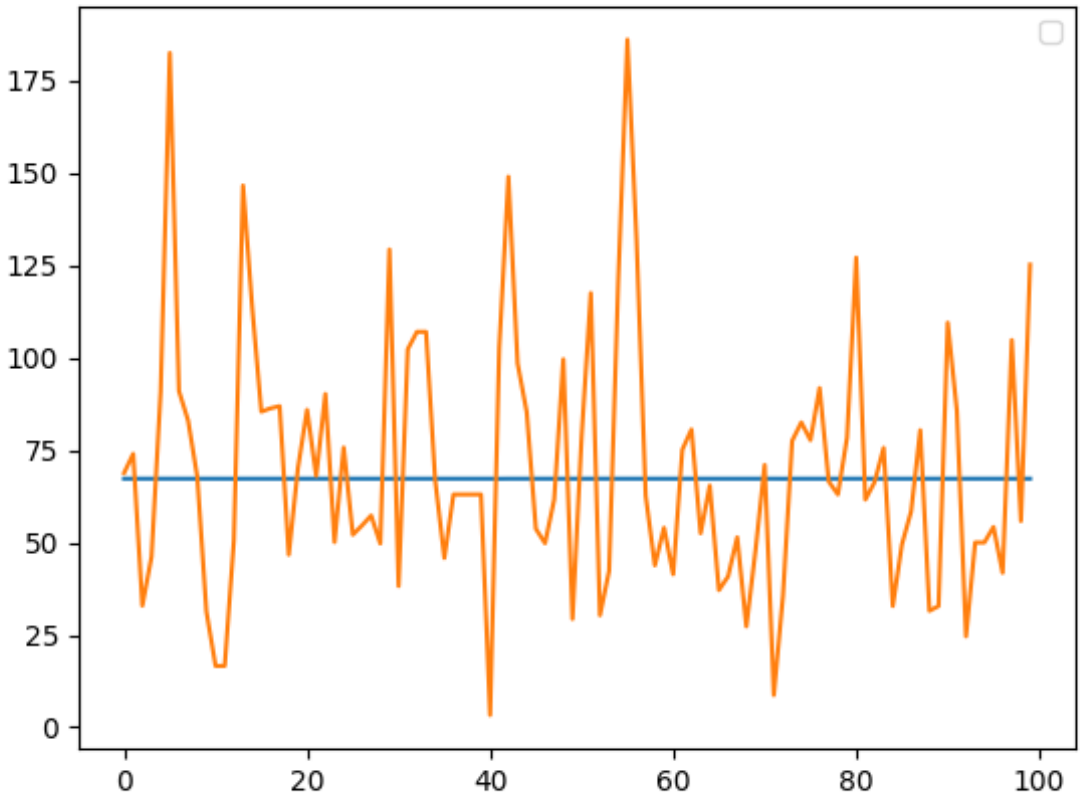


Figure 3.30: Plot of predicted vs actual values based on *armagh* station's dataset using LSTM

LSTM model's results on *armagh* station for testing data	
Name of measures	**Values**
Mean Absolute Error (MAE)	25.851
Root Mean Squard Error (RMSE)	34.211

Table 3.3: Performance measures of LSTM model on the testing dataset (*armagh*)

LSTM model's results on *oxford* station for testing data	
Name of measures	**Values**
Mean Absolute Error (MAE)	24.256
Root Mean Squard Error (RMSE)	32.102

Table 3.4: Performance measures of LSTM model on the testing dataset (*oxford*)

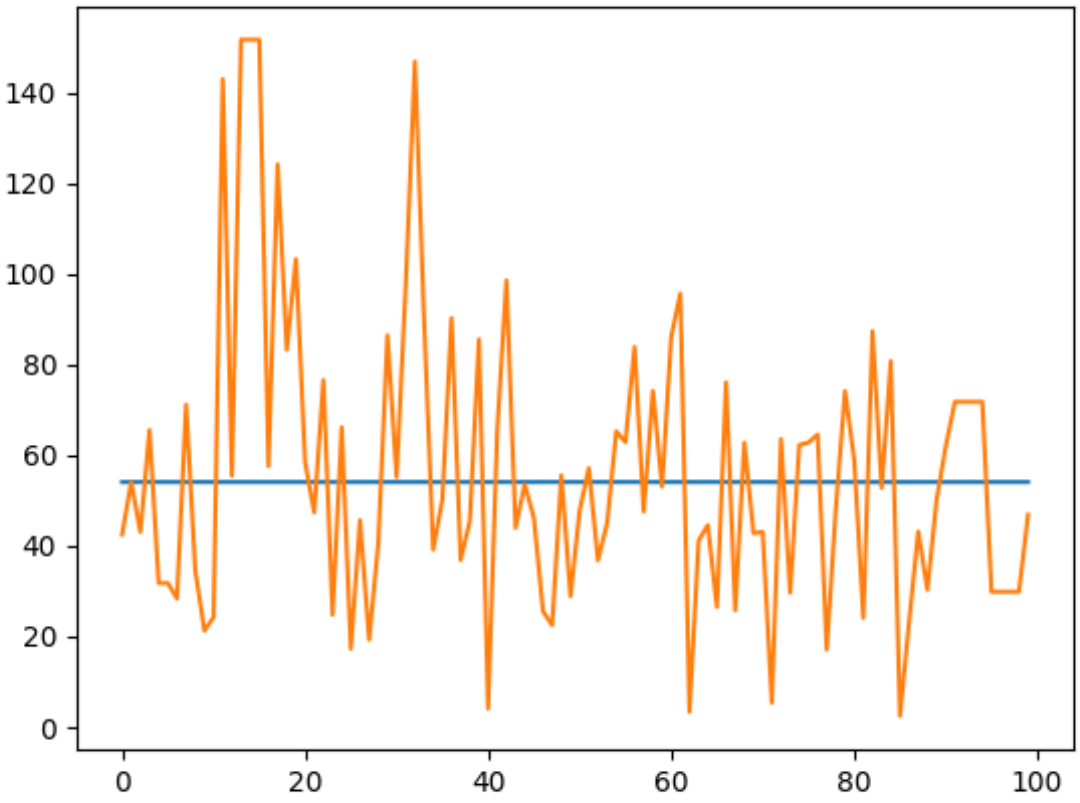


Figure 3.31: Plot of predicted vs actual values based on *oxford* station's dataset using LSTM

Test prediction values	**Actual values**
29.424	66.8
59.189	16.1
33.207	49.8
30.435	135.3
28.756	131.0
92.322	109.8
31.396	25.8
35.792	25.8
42.231	58.5
22.629	80.6

Table 3.5: Predicted vs actual values based *armagh* weather station dataset using Stacked LSTM

Test prediction values	Actual values
53.002	36.9
54.386	1.8
49.788	135.2
61.111	78.8
54.441	105.5
55.283	38.2
53.865	17.7
49.916	68.9
60.594	56.0
51.623	60.6

Table 3.6: Predicted vs actual values based *oxford* weather station dataset using Stacked LSTM

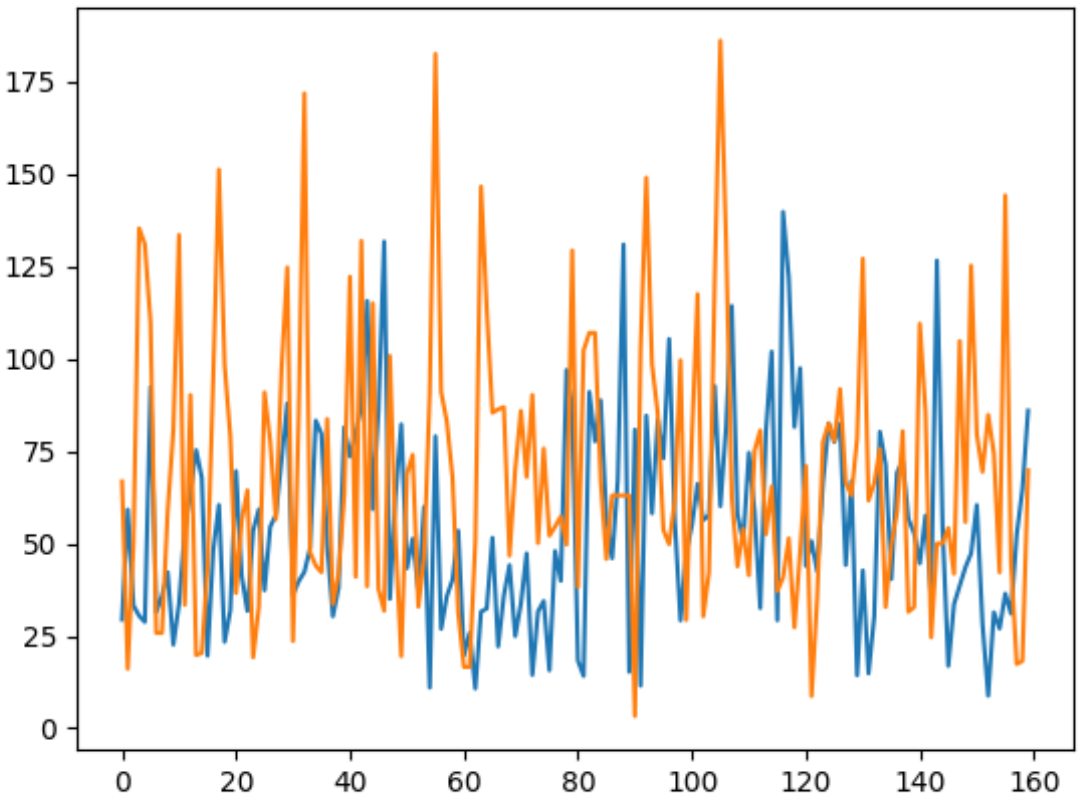


Figure 3.32: Plot of predicted vs actual values based on *armagh* station's dataset using Stacked LSTM

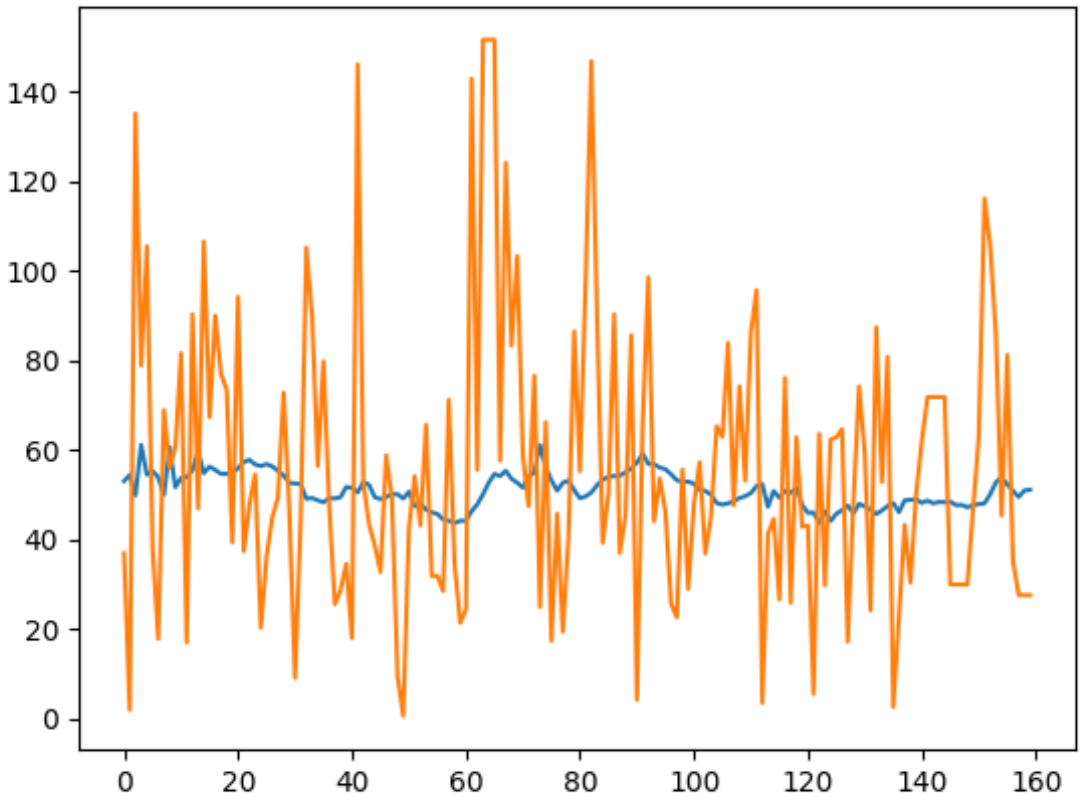


Figure 3.33: Plot of predicted vs actual values based on *oxford* station's dataset using Stacked LSTM

Stacked LSTM model's results on *armagh* station for testing data	
Name of measures	**Values**
Mean Absolute Error (MAE)	38.690
Root Mean Squard Error (RMSE)	48.382

Table 3.7: Performance measures of stacked LSTM model on the testing dataset (*armagh*)

Stacked LSTM model's results on *oxford* station for testing data	
Name of measures	**Values**
Mean Absolute Error (MAE)	24.533
Root Mean Squard Error (RMSE)	32.435

Table 3.8: Performance measures of stacked LSTM model on the testing dataset (*oxford*)

CHAPTER 4

Conclusions

I started my dissertation by selecting the dissertation topic, which is *Predication of rainfall in the UK using machine learning techniques* . Then I started writing about why this topic is important. After that, I started reading previous works related to my chosen topic. I read about eight papers and have written a summary for each. Then I talked about methods and strategies in detail that I've used to complete my task on my chosen topic successfully. After that, I collected a dataset from Kaggle named *UK Met Office Weather Data* with the assistance of my supervisor and started exploring the whole dataset. After exploration of the dataset, I preprocessed it. After that, I explained the machine learning-based models that I've used, saying how these will be implemented and what the hyperparameter tuning will be. After the implementation of the models on the dataset, I discussed the results.

In my first strategy, I implemented support vector regressor (SVR) and random forest regressor (RFR) to predict rainfall on a month-by-month basis. I removed the *year* and *month* features from the dataset and worked with the remaining features like maximum temperature, minimum temperature, air forst amount, and sunlight amount. I found results from performance metrics for SVR, which is mentioned in the table 3.1 and for RFR, which is mentioned in the table 3.2. The evaluation results showed that the SVR model got 28.786 and 1481.058 values for MAE and MSE, respectively, while the RFR model got 27.486 and 1295.877 values for MAE and MSE, respectively. So in terms of

MAE and MSE, the RFR model outperformed the SVR model in rainfall prediction.

In my second strategy, I resampled the main dataset into two individual datasets based on the weather stations *armagh* and *oxford*, then I implemented long short term memory (LSTM) and Stacked LSTM for each of these resampled datasets. From the evaluation results, I found that for both resampled datasets, the LSTM model for *armagh* weather station resampled dataset got 25.851 and 34.211 values for MAE and RMSE, respectively, while the LSTM model for *oxford* weather station resampled dataset got 24.256 and 32.102 values for MAE and RMSE, respectively. So, in terms of MAE and RMSE, the LSTM model for the resampled dataset based on *oxford* weather station outperforms the LSTM model for the resampled dataset based on *armagh* weather station. After that, I used Stacked LSTM for these datasets. With *armagh* dataset, the model got 38.690 and 48.382 values for MAE and RMSE, respectively, whereas with *oxford* dataset, the model got 24.533 and 32.435 values for MAE and RMSE, respectively. So, in terms of MAE and RMSE, the stacked LSTM model for the resampled dataset based on *oxford* weather station outperforms the Stacked LSTM model for the resampled dataset based on *armagh* weather station. On the other hand, the evaluation results from the paper, which is [13], Stacked-LSTM got RMSE values ranging between 0.0375 and 0.0084 and MAE values ranging between 0.0071 and 0.0013 in their study with different cities. In comparison with these measure's values of this model, my Stacked LSTM models evaluation measure's values show that they do not perform well as expected and that their errors are bigger than their errors.

While comparing among the models, which are support vector regressor (SVR), random forest (RF), long short term memory (LSTM), and stacked long short term memory (Stacked LSTM). I've implemented these models to predict total rainfall on a month-by-month basis. From all of the models, RF with all the weather stations in the dataset outperformed all the other models used for the task because this model has the least MAE error.

The limitation of my methodology is that I wanted to train and test LSTM and Stacked

LSTM models with the whole dataset, including all of the weather stations, not only *armagh* and *oxford*. But I couldn't do that because of the complexity of managing the dataset with all the weather stations for the models. Another limitation was that I tried to train the Stacked LSTM with 100 *timestemps/ timelags*, but after some epocs while training, the loss of all evaluation metrics appeared as value *nan*. In future work, it would be interesting to study these limitations. On the other hand, while I was running or executing LSTM and Stacked LSTM models, the system produced unusual results and repeated the same results. Then I had to delete the runtime kernel and restart the kernel. It took a lot of my valuable time.

Appendix 1

A.1 The following codes for exploratory data analysis, SVR model, and RF model

```
# -*- coding: utf-8 -*-
"""Rainfall Prediction.ipynb

Automatically generated by Colaboratory.

Original file is located at the following link;
```

[3]

```
Loading libraries
"""

import numpy as np
import pandas as pd
```

```
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2

"""Downloading the dataset"""

pip install opendatasets --upgrade --quiet

import opendatasets as od

download_url = 'https://www.kaggle.com/datasets/josephw20/uk-met-office

od.download(download_url)

"""Loading the dataset"""

data_filename="/content/uk-met-office-weather-data/MET Office Weather D

df=pd.read_csv(data_filename)

"""Exploratory data analysis"""

pd.set_option('display.max_row', None)

df.head()

df.shape

df.info()

df.describe()
```

```
for features in df.columns:
  print(df[features].unique())

"""Searching for missing values in the dataset


"""

df.isnull().sum()

[f for f in df.columns if df[f].isnull().sum()>0]

sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="viridis")
# plt.savefig('/content/missing.png')

for features in df.columns:
  print(features,'contains: ', df[features].isnull().sum()/len(df)*100,

"""Handling missing values"""

df=df.dropna(subset=['year', 'month'])
df.isnull().sum()

df.columns

cols_fill=['tmax', 'tmin', 'af', 'rain', 'sun']
for i in cols_fill:
  df[i].ffill(inplace=True)
  df[i].bfill(inplace=True)

df.isnull().sum()
```

```
"""checking correlation among features"""

plt.rcParams['figure.figsize']=[7, 4]
corr=df.corr()
sns.heatmap(corr, annot=True, fmt=".3f")
# plt.savefig('/content/corrplot.png')

"""Checking outliers"""

plt.rcParams['figure.figsize']=[8, 4]
sns.boxplot(df[['tmax', 'tmin', 'af', 'rain', 'sun']], orient='v')
# plt.savefig('/content/boxplot.png')

print(max(df['rain']))

plt.rcParams['figure.figsize']=[5, 5]
for i in df[['tmax', 'tmin', 'af', 'rain', 'sun']]:
  sns.distplot(df[i])
  plt.xlabel(i)
  plt.ylabel("Count")
  plt.title(i)
  plt.figure(figsize=(15,15))
  plt.show()
  # plt.savefig(f'/content/{i}.')

"""Handling outliers"""

for i in ['tmax', 'tmin', 'rain', 'sun']:
  upperValue=df[i].mean() + 3* df[i].std()
  lowerValue=df[i].mean() - 3* df[i].std()
  print(i, ":")
```

```
  print(lowerValue), print(upperValue),print(df[i].max()), print(df[i].
  print()
  df.loc[df[i]>=upperValue, i]=upperValue
  df.loc[df[i]<=lowerValue, i]=lowerValue


print()
iqr=df.af.quantile(0.75)-df.af.quantile(0.25)
lower_value=df['af'].quantile(0.25)-(iqr*3)
upper_value=df['af'].quantile(0.75)+(iqr*3)
print('af')
print(lower_value), print(upper_value), print(df['af'].max()), print(df
df.loc[df['af']>=upper_value, 'af']=upper_value
df.loc[df['af']<=lower_value, 'af']=lower_value

df.describe()

df.head()

"""Saving preprocessed dataset 2"""

df.to_csv('/content/uk_weather_data_preprocessed2.csv')

df=pd.read_csv('/content/uk_weather_data_preprocessed2.csv')
df.head()

df.year, df.month=df.year.astype(int), df.month.astype(int)
df['Date']=pd.to_datetime(df.year.astype(str)+'-'+df.month.astype(str))
df['Date2']=pd.to_datetime(df.year.astype(str)+'-'+df.month.astype(str)

"""```
Exploratory analysis of the feature 'year'.
```

```
Questions:
how many unique values in the year feature?
how many value count for each of the unique year?
what is the relation between year and station feature?
```
"""

print(len(df.year.unique()))
df.year.unique()

fig = plt.subplots(figsize=(20, 5))
sns.lineplot(x=df.year, y=df.year.value_counts(), data=df, marker='*',
sns.set_theme(style='dark', font_scale=3)
# plt.savefig('/content/year_value_counts.png')

df.year.value_counts()

plt.rcParams['figure.figsize'] = [30, 20]
sns.scatterplot(data=df, x="year", y="station")
# plt.savefig('/content/stationvsyear.png')

"""Time siries plotting

```
Exploratory data analysis of the target variable on time series plot.

Questions:
What kind of trends are the time series plot following?
Is there any seasonality present?
```
"""

```
df.index=df.Date
df.head()

df.rain.plot(figsize=(12, 4), xlim=['2000-01-01', '2020-12-01'], c='bla
plt.ylabel('rainfall in mm')
# plt.savefig('/content/ts2000.png')

df.year.min()

df.rain.plot(figsize=(12, 4), xlim=['1975-01-01', '1999-12-01'], c='gre
plt.ylabel('rainfall in mm')
# plt.savefig('/content/ts1975.png')

df.rain.plot(figsize=(12, 4), xlim=['1935-01-01', '1974-12-01'], c='blu
plt.ylabel('rainfall in mm')
# plt.savefig('/content/ts1935.png')

df.rain.plot(figsize=(12, 4), xlim=['1901-01-01', '1934-12-01'], c='bro
plt.ylabel('rainfall in mm')
# plt.savefig('/content/ts1901.png')

df.rain.plot(figsize=(12, 4), xlim=['1853-01-01', '1900-12-01'], c='red
plt.ylabel('rainfall in mm')
# plt.savefig('/content/ts1853.png')

"""```
Exploratory data analysis of the feature 'rainfall'.


Questions:
How the records look like for rainfall in a month?
```

```
In which month the rate of rainfall has got highest and lowest?
For which weather stations the rainfall was maximum regarding month cou
For which weather stations the ranfall was minimun regarding month coun
```
```
"""

df.rain.plot(kind='hist')
plt.ylabel('No. of records for stations in various year with month')
plt.xlabel('Rainfall amount in mm')
# plt.savefig('/content/histrain.png')

df.rain.describe()

hirec=df.iloc[np.where(df['rain']>=217.23)]
hirec['month'].value_counts()

plt.bar(hirec.station.unique(), hirec.station.value_counts(), color=['g
plt.xticks(rotation=45)
plt.xlabel('Different weather stations in UK')
plt.ylabel('Counts of highest rainfall in a month of years')
# plt.savefig('/content/maxrainsta.png')

hirec=df.iloc[np.where(df['rain']<=0)]
hirec['month'].value_counts()

plt.bar(hirec.station.unique(), hirec.station.value_counts(), width=0.2
plt.xticks(rotation=45)
plt.xlabel('Different weather stations in UK')
plt.ylabel('Counts of lowest rainfall in a month of years')
# plt.savefig('/content/minrainsta.png')

"""```
```

```
Exploratory data analysis of the feature 'station'.

Questions:
How many unique weather stations are in the dataset?
What are percentages of stations used for different dates?
What are the stations have got rainfall less equal 10 mm?
What are the stations have got rainfall greater equal 200 mm?

What was the condition of rainfall after the year 2000?
What was the condition of rainfall from year 1975 to 1999?
What was the condition of rainfall before the year 1975?
```
"""

print(len(df.station.unique()))

plt.pie(df.station.value_counts(), labels=df.station.unique(), autopct=
plt.savefig('/content/stationpercen.png')

pd.set_option('display.max_row', None)
df[['station', 'Date2', 'rain']]

rain10=df.loc[df.rain<=10]
rain200=df.loc[df.rain>=200]
print(len(rain10.station.unique()))
print(len(rain200.station.unique()))

rain2000=df.iloc[np.where(df.year>=2000)]
rain2000['rain'].describe()

rain7599=df.loc[(df['year']>=1975) & (df['year']<=1999)]
rain7599['rain'].describe()

```
rain1974=df.loc[(df['year']<=1975)]
rain1974['rain'].describe()

"""Observing the feature variables 'tmax', 'tmin', 'af', and 'sun' usin

for i in ['tmax', 'tmin', 'af', 'sun']:
  plt.hist(df[i])
  plt.xlabel('The feature'+ ' '+i)
  plt.ylabel('No. of records')
  # plt.savefig('/content/hist'+i+'.png')
  plt.show()

"""Converting categorical variable 'station' to numeric"""

df.reset_index(drop=True, inplace=True)
df=pd.get_dummies(df, columns=['station'])
df.head()

"""Saving uk_weather_data_preprocessed1"""

df.to_csv('/content/uk_weather_data_preprocessed1.csv')

"""Model training and prediction part for pre-processed dataset version

pdf=pd.read_csv('/content/uk_weather_data_preprocessed1.csv')
pdf.head()

pdf.columns

pdf_cols=['tmax', 'tmin', 'af', 'sun', 'station_aberporth', 'station_ar
       'station_ballypatrick', 'station_bradford', 'station_braemar',
```

```
        'station_camborne', 'station_cambridge', 'station_cardiff',
        'station_chivenor', 'station_cwmystwyth', 'station_dunstaffnage'
        'station_durham', 'station_eastbourne', 'station_eskdalemuir',
        'station_heathrow', 'station_hurn', 'station_lerwick',
        'station_leuchars', 'station_lowestoft', 'station_manston',
        'station_nairn', 'station_newtonrigg', 'station_oxford',
        'station_paisley', 'station_ringway', 'station_rossonwye',
        'station_shawbury', 'station_sheffield', 'station_southampton',
        'station_stornoway', 'station_tiree', 'station_valley',
        'station_waddington', 'station_whitby', 'station_wickairport',
        'station_yeovilton']
x=pdf[pdf_cols]
y=pdf['rain']
y

x.head()

"""Splitting data into train and test and Scaling"""

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.33,

scaler=StandardScaler()
scaler.fit(X_train)
scaled_trainx=scaler.transform(X_train)
scaled_testx=scaler.transform(X_test)

"""Using first model support vector regressor (SVR)"""

from sklearn.svm import SVR
from sklearn.model_selection import GridSearchCV
```

```
# parameters={
#             'kernel': ['rbf', 'sigmoid', 'poly'],
#             'C':[1.5, 10],'gamma': [1e-7, 1e-4],'epsilon':[0.1,0.2,0.
#             }

# svrModel=GridSearchCV(SVR(), parameters)
# svrModel.fit(scaledsvr_trainx, y_train)
# svrModel.best_params_


svr=SVR(kernel='rbf', gamma=1, epsilon=0.01)
svrModel=svr.fit(scaled_trainx, y_train)

svrypred=svrModel.predict(scaled_testx)

print(mean_absolute_error(y_test, svrypred))
print(mean_squared_error(y_test, svrypred))
print(r2_score(y_test, svrypred))

Adj_r2 = 1 - (1-r2_score(y_test, svrypred)) * (len(y_test)-1)/(len(y_te
print(Adj_r2)

plt.rcParams['figure.figsize']=[10, 5]
plt.plot(y_test, label='y_test')
plt.plot(svrypred, label='svrypred')
plt.legend()

test_result = pd.DataFrame(data={'Predictions':svrypred, 'Actuals':y_te
test_result
```

```
"""Using second model Random forest"""

from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfrModel=rfr.fit(scaled_trainx, y_train)

rfrypred=rfrModel.predict(scaled_testx)

print(mean_absolute_error(y_test, rfrypred))
print(mean_squared_error(y_test, rfrypred))
print(r2_score(y_test, rfrypred))

Adj_r2 = 1 - (1-r2_score(y_test, rfrypred)) * (len(y_test)-1)/(len(y_te
print(Adj_r2)

plt.rcParams['figure.figsize']=[10, 5]
plt.plot(y_test, label='y_test')
plt.plot(rfrypred, label='rfrypred')
plt.legend()

rfrtest_result = pd.DataFrame(data={'Predictions':rfrypred, 'Actuals':y
rfrtest_result
```

APPENDIX B

Appendix 2

B.1 The following codes for LSTM model and Stacked LSTM model

Make sure the pre-processed dataset named *uk_weather_data_preprocessed2.csv* has been successfully imported, which was created in the previous notebook file. The codes of the file have been put in Appendix 1. In this portion of the code, I made some comments and uncomments for the four variations of LSTM models to train the models and reduce the code length.

```
# -*- coding: utf-8 -*-
"""rainfall prediction stacked lstm.ipynb

Automatically generated by Colaboratory.

Original file is located at the following link;
```

[4]

```
importing libraries
"""

import tensorflow as tf
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score as r2s
from math import sqrt

"""Loading pre processed dataset version 2"""

df1=pd.read_csv('/content/uk_weather_data_preprocessed2.csv')
df1.head()

df1.columns

df1_cols=['year', 'month', 'tmax', 'tmin', 'af', 'rain', 'sun',
       'station']
df1=df1[df1_cols]
df1.head()

"""Resampling dataset based on armagh weather station as df1 and oxford

df1=df1.loc[df1['station']=='armagh']
df1.drop(['station'], axis=1, inplace=True)
df1['year']=df1['year'].astype(int)
df1['month']=df1['month'].astype(int)
```

```
df1["Date"] = pd.to_datetime(df1["year"].astype(str) + "/" + df1["month
df1.head()

# df2=df1.loc[df1['station']=='oxford']
# df2.drop(['station'], axis=1, inplace=True)
# df2['year']=df2['year'].astype(int)
# df2['month']=df2['month'].astype(int)
# df2["Date"] = pd.to_datetime(df2["year"].astype(str) + "/" + df2["mon
# df2.head()

"""Setting index to Date feature"""

df1.index = pd.to_datetime(df1['Date'])
df1.head()



# df2.index = pd.to_datetime(df2['Date'])
# df2.head()

"""Taking 'rain' variable to make independent features and target varia

rainfall=df1['rain']
rainfall.plot()
plt.xticks(rotation=45)



# rainfall=df2['rain']
# rainfall.plot()
# plt.xticks(rotation=45)

# plt.savefig('/content/timesiriesoxford.png')
```

```
rainfall

"""Creating features and label from the time siries prediction variable

# [[[1], [2], [3], [4], [5]]] [6]
# [[[2], [3], [4], [5], [6]]] [7]
# [[[3], [4], [5], [6], [7]]] [8]

def df_to_X_y(df, timestemps):
  df_as_np = df.to_numpy()
  X = []
  y = []
  for i in range(len(df_as_np)-timestemps):
    row = [[a] for a in df_as_np[i:i+timestemps]]
    X.append(row)
    label = df_as_np[i+timestemps]
    y.append(label)
  return np.array(X), np.array(y)

timestemps=50
X1, y1 = df_to_X_y(rainfall, timestemps)
X1.shape, y1.shape

X1

y1

X_train1, y_train1 = X1[:1500], y1[:1500]
X_val1, y_val1 = X1[1500:1800], y1[1500:1800]
X_test1, y_test1 = X1[1800:], y1[1800:]
X_train1.shape, y_train1.shape, X_val1.shape, y_val1.shape, X_test1.sha
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam

model1 = Sequential()

model1.add(LSTM(32, activation='tanh', input_shape=(X_train1.shape[1],

model1.add(Dense(1))

model1.summary()

cp1 = ModelCheckpoint('model1/', save_best_only=True)
model1.compile(loss=MeanSquaredError(), optimizer=Adam(learning_rate=0.

model1.fit(X_train1, y_train1, validation_data=(X_val1, y_val1), epochs

from tensorflow.keras.models import load_model
model1 = load_model('model1/')

train_predictions = model1.predict(X_train1).flatten()
train_results = pd.DataFrame(data={'Train Predictions':train_prediction
train_results[:30]

import matplotlib.pyplot as plt
plt.plot(train_results['Train Predictions'][100:200])
plt.plot(train_results['Actuals'][100:200])

val_predictions = model1.predict(X_val1).flatten()
```

```
val_results = pd.DataFrame(data={'Val Predictions':val_predictions, 'Ac
val_results

plt.plot(val_results['Val Predictions'][:100])
plt.plot(val_results['Actuals'][:100])

test_predictions = model1.predict(X_test1).flatten()
test_results = pd.DataFrame(data={'Test Predictions':test_predictions,
test_results[:20]

test_results.iloc[:, 0]

plt.plot(test_results['Test Predictions'][:100])
plt.plot(test_results['Actuals'][:100])
plt.savefig('/content/olstmtestresfig.png')

"""Performance measures"""

mae1=mae(test_results.iloc[:, 1],test_results.iloc[:, 0])
mse1=mse(test_results.iloc[:, 1],test_results.iloc[:, 0])
print(mae1)
# print(mse1)
rmse1=sqrt(mse1)
print(rmse1)
r2_score=r2s(test_results.iloc[:, 1],test_results.iloc[:, 0])
print(r2_score)

"""Using Stacked LSTM"""

model=Sequential()
model.add(LSTM(32,activation='relu',input_shape=(X_train1.shape[1],X_tr
model.add(Dropout(0.3))
```

```
model.add(LSTM(32,activation='relu',return_sequences=True))
model.add(Dropout(0.3))

model.add(LSTM(32,activation='relu',return_sequences=False))
model.add(Dropout(0.3))

model.add(Dense(1))

model.summary()

cp = ModelCheckpoint('model/', save_best_only=True)
model.compile(loss=MeanSquaredError(), optimizer=Adam(learning_rate=0.0

model.fit(X_train1, y_train1, validation_data=(X_val1, y_val1), epochs=

from tensorflow.keras.models import load_model
model = load_model('model/')

train_predictions = model.predict(X_train1).flatten()
train_results = pd.DataFrame(data={'Train Predictions':train_prediction
train_results

import matplotlib.pyplot as plt
plt.plot(train_results['Train Predictions'][:100])
plt.plot(train_results['Actuals'][:100])

val_predictions = model.predict(X_val1).flatten()
val_results = pd.DataFrame(data={'Val Predictions':val_predictions, 'Ac
val_results

plt.plot(val_results['Val Predictions'][:100])
```

```
plt.plot(val_results['Actuals'][:100])

test_predictions = model.predict(X_test1).flatten()
test_results = pd.DataFrame(data={'Test Predictions':test_predictions,
test_results[:15]

plt.plot(test_results['Test Predictions'][:])
plt.plot(test_results['Actuals'][:])
plt.savefig('/content/oxlstmtestresfig.png')

"""Performance measures"""

mae1=mae(test_results.iloc[:, 1],test_results.iloc[:, 0])
mse1=mse(test_results.iloc[:, 1],test_results.iloc[:, 0])
print(mae1)
# print(mse1)
rmse1=sqrt(mse1)
print(rmse1)
r2_score=r2s(test_results.iloc[:, 1],test_results.iloc[:, 0])
print(r2_score)
```

References

[1] Climate change explained. https://www.gov.uk/guidance/climate-change-explained#full-publication-update-history. last updated: 20 June 2023.

[2] Colorado climate center. http://ccc.atmos.colostate.edu/.

[3] Jupyter notebook coding file 1 (exploratory data analysis, svr, rf). https://colab.research.google.com/drive/1LBdy9ET1RmJ4aGjaNAFEGLRejKZt_Ltt.

[4] Jupyter notebook coding file 2 (lstm, stacked lstm). https://colab.research.google.com/drive/1WSbff9DOgSVgHEMg1dTtT6ftl9G_Yheu.

[5] Lstm structure details. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[6] Met office. https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data.

[7] R2 score and adjusted r2. https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

[8] Random forest. https://www.geeksforgeeks.org/random-forest-regression-in-python/.

[9] Reinforcement learning. https://en.wikipedia.org/wiki/Reinforcement_learning.

[10] Support vector machine. https://www.geeksforgeeks.org/support-vector-machine-algorithm/.

[11] Uk met office weather data. https://www.kaggle.com/datasets/josephw20/uk-met-office-weather-data.

[12] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. Machine learning from theory to algorithms: an overview. In *Journal of physics: conference series*, volume 1142, page 012012. IOP Publishing, 2018.

[13] Ari Yair Barrera-Animas, Lukumon O Oyedele, Muhammad Bilal, Taofeek Dolapo Akinosho, Juan Manuel Davila Delgado, and Lukman Adewale Akanbi. Rainfall prediction: A comparative analysis of modern machine learning algorithms for time-series forecasting. *Machine Learning with Applications*, 7:100204, 2022.

[14] Bogdan Bochenek and Zbigniew Ustrnul. Machine learning in weather prediction and climate analysesâapplications and perspectives. *Atmosphere*, 13(2):180, 2022.

[15] Jenny Cifuentes, Geovanny Marulanda, Antonio Bello, and Javier Reneses. Air temperature forecasting using machine learning techniques: a review. *Energies*, 13(16):4215, 2020.

[16] Rochelle Schneider dos Santos. Estimating spatio-temporal air temperature in london (uk) using machine learning and earth observation satellite data. *International Journal of Applied Earth Observation and Geoinformation*, 88:102066, 2020.

[17] MA Ghorbani, R Khatibi, MH FazeliFard, L Naghipour, and O Makarynskyy. Short-term wind speed predictions with machine learning techniques. *Meteorology and Atmospheric Physics*, 128:57–72, 2016.

[18] V Bharat Kumar, V Mallikarjuna Nookesh, B Satya Saketh, S Syama, and J Ramprabhakar. Wind speed prediction using deep learning-lstm and gru. In *2021 2nd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 602–607. IEEE, 2021.

[19] VP Tharun, Ramya Prakash, and S Renuga Devi. Prediction of rainfall using data mining techniques. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1507–1512. IEEE, 2018.