Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Integrating Neural Networks and Time-Series Analysis for Accurate Rainfall Prediction

[1]Shilpa Jaiswal, [2]Miss Pooja Patre, [3]Miss Jasmine Minj, [4]Mr Advait Khare

[1]Research Scholar, [2]Assistant Professor, [3]HOD of CSE Department, [4]SSIPMT
[1]Department of Computer Science Engineering,
[1]Vishwavidyalaya Engineering College, Ambikapur, C.G.India

Abstract
Predicting rainfall accurately is critical for agriculture, water resource management, and disaster preparedness. Using historical meteorological data, this study use machine learning approaches to develop rainfall forecast models. Regression and time-series forecasting methods such as support vector machines, random forests, and long short-term memory neural networks are trained using several input factors such as temperature, humidity, wind patterns, and so on. The models forecast rainfall amounts for the following day, week, and month at specified places with greater than 80% accuracy. This highlights the viability of utilizing sophisticated machine learning approaches for localized real-time rainfall forecasting to enhance data-driven decision making in a variety of sectors. The objective of current research is to develop neural network architectures like RNNs and Transformers to model linguistic contexts and sequences for language understanding. The RNN model is one of the deep learning model that is trained for processing and it also helps to convert the sequential data into the totally newer sequential data individuals. Through this model the sequential data component creates the interrelation between the various words or sentences or the time series data through the forecasting and including the complex syntax and semantics rules. Whereas the NLP pipeline is another approach that are applied to the sequential data processing which transforms the raw textual data into the meaningful insights data variables. Through this NLP processing the information extraction, pattern detections, the various language related classifications are done for the sentimental analysis and recognition process of named entities. In this report the statistical analysis based on the historical rainfall measurements and the machine learning models the precipitation has been predicted using the remote monitoring instruments for the imputed features.

Index Terms: RNN, Rainfall Prediction

1. INTRODUCTION

Rainfall forecasting is very important because heavy and irregular rainfall can have many impacts like destruction of crops and farms, damage of property so a better forecasting model is essential for an early warning that can minimize risks to life and property and also managing the agricultural farms in better way. This prediction mainly helps farmers and also water resources can be utilized efficiently. Rainfall prediction is a challenging task and the results should be accurate. There are many hardware devices for predicting rainfall by using the weather conditions like temperature, humidity, pressure. These traditional methods cannot work in an efficient way so by using machine learning techniques we can produce accurate results. We can just do it by having the historical data analysis of rainfall and can predict the rainfall for future seasons. We can apply many techniques like classification, regression according to the requirements and also we can calculate the error between the actual and prediction and also the accuracy. Different techniques produce different accuracies so it is important to choose the right algorithm and model it according to the requirements.

2. LITERATURE REVIEW

Wu et. al. [1] Having an accurate short-term precipitation forecast is crucial for mitigating natural disasters and issuing timely warnings about urban floods. In this study, we present ISA-PredRNN (improved self-attention PredRNN), a new deep learning model designed specifically for precipitation nowcasting. The construction of this model is based on the advanced PredRNN-V2 architecture and utilizes radar reflections as the input data. The present study introduces a novel set of gating mechanisms that integrate the long-term memory state and the self-attention mechanism into the model. The development of the weighted loss function aimed to enhance the accuracy in representing different precipitation intensities. In order to enhance the model's training and achieve a deeper understanding of the long-term patterns observed in the radar echo sequences, we employ a combination of reverse scheduled sampling and planned sampling. The experimental results indicate that the model, ISA-PredRNN, successfully captures the spatiotemporal features of radar echo maps.

Fredyan et. al. [2] An accurate short-term forecasting system for predicting changes in precipitation is crucial for ensuring the protection of individuals and their belongings. Data-driven strategies are considered effective for extrapolating precipitation. The purpose of this study was to present a prediction model called the Self-Attentive Mechanism-Based Spatial Temporal Long Short-Term Memory (ST-LSTM-SA). The method described here is motivated by proposed enhancements and aims to facilitate enhanced sequence feature aggregation. The effectiveness of the intended encapsulated 3D convolution has been confirmed through an ablation study, which maximizes the utilization of short-term spatiotemporal information. In addition, the self-attention mechanism is utilized to model the correlation between channels, thereby improving the representation of long-term interactions. An exhaustive analysis of the radar echo sequence has been performed.

Ikpang et. al. [3] Meteorological services around the globe have the responsibility of performing the crucial and challenging operational duty of forecasting precipitation. This article aims to compare the Artificial Neural Network (ANN) models with the traditional Seasonal Autoregressive Integrated Moving Average (SARIMA) models in order to determine the most effective model for predicting rainfall in Nigeria. An analysis was conducted on the mean monthly precipitation data of Nigeria spanning from January 1991 to December 2020.

The suitability of the SARIMA (1,0,2)x(1,1,2)12 model for forecasting average monthly rainfall is indicated by the ACF and PACF diagrams.

Kim et. al. [4] Weather radars are highly effective instruments for real-time monitoring of rainfall. They possess the ability to detect both instantaneous rain rates and the dispersion of rainfall. The Korea Meteorological Administration (KMA) currently offers real-time radar data and short-term forecasting predictions using the McGill algorithm for precipitation nowcasting through Lagrangian extrapolation (MAPLE). The duration of these projections may span up to six hours. This study presents a novel radar rainfall prediction approach that employs a conditional generative adversarial network (CGAN) to generate highly accurate short-term weather forecasts, specifically targeting the time frame between 10 minutes and 4 hours. The CGAN model was trained and evaluated using observation data from KMA's constant altitude plan position indicator (CAPPI)..

Hossain et. al. [5] Precise rainfall forecasting is essential for planning agricultural, water infrastructure, and other socioeconomic developments. The ability to forecast hydrologic data for the near future, typically within a 10-year period, is a recent advancement in General Circulation Model (GCM) simulations, particularly in the decadal experiments of the Coupled Model Intercomparison Project Phase 5 (CMIP5). The projection of monthly rainfall over a period of ten years is an essential aspect of watershed management. Prior studies have not incorporated GCM decadal data with observed data at the catchment level. Prior research has primarily focused on analyzing stochastic models that rely solely on observed time series data for predicting rainfall. Over the course of ten years, this investigation utilized the Facebook Prophet (FBP) model and six regression methods for machine learning (ML) in order to predict monthly rainfall in the Brisbane River basin in Queensland, Australia. The observed data was obtained from the Australian Bureau of Meteorology, while the monthly hindcast decadal precipitation data for eight General Circulation Models (GCMs) (EC-EARTH MIROC4h, MRI-CGCM3, MPI-ESM-LR, MPI-ESM-MR, MIROC5, CanCM4, and CMCC-CM) was sourced from the CMIP5 data site. The FBP model generated predictions by utilizing two distinct sets of data: (i) the observed data exclusively, and (ii) a combination of the observed data and CMIP5 decadal data. The target variables in machine learning regression models were derived from the corresponding observed data, while the input features were derived from the CMIP5 decadal data.

Xiao et. al. [6] Accurate prediction of precipitation plays a crucial role in the advancement of agricultural development and the administration of water resources. The present study proposes a monthly rainfall forecast methodology that utilizes deep learning techniques and incorporates a convolutional neural network (CNN). The selection of this method is based on the influence of a variety of complex, nonlinear, and unstable factors on rainfall. These factors can be effectively addressed by employing deep learning techniques. The focus of this research is the analysis of five months of rainfall data at a time. The rainfall data used for this analysis is sourced from Lhasa and covers the period from 2009 to 2021. The objective of this study is to predict the amount of rainfall for the upcoming month. To achieve this, various parameter adjustments will be made to determine the most effective approach. The evaluation of the prediction outputs' accuracy is performed by comparing them to the results obtained from an ARIMA (Autoregressive Integrated Moving Average) model using differential integration.

Appiah et. al. [7] The accuracy of rainfall prediction has been affected by the recent increase in complexity due to the unpredictable and fluctuating climate. Recent advancements in classification algorithms have significantly improved the accuracy of rainfall forecasting. The objective of this study is to analyze the effectiveness of different classification algorithms in improving the accuracy of rainfall predictions in the distinct ecological zones of Ghana. Classification methods commonly used in machine learning include Decision Tree (DT), Random Forest (RF), Multilayer Perceptron (MLP), Extreme Gradient Boosting (XGB), and K-Nearest Neighbor (KNN). The dataset was contributed by the Ghana Meteorological Agency and includes a range of climatic variables spanning from 1980 to 2019. The evaluation of the classification algorithms included assessing their accuracy, recall, precision, f1-score, and execution time using different training-to-testing data ratios. The Random Forest (RF), Extreme Gradient Boosting (XGB), and Multilayer Perceptron (MLP) algorithms all exhibited satisfactory performance across all three training and testing ratios (70:30, 80:20, and 90:10). Nevertheless, the K-Nearest Neighbors (KNN) algorithm exhibited inadequate performance in all instances. The MLP model consistently exhibits the longest execution time, whereas the Decision Tree model consistently exhibits the minimum execution time.

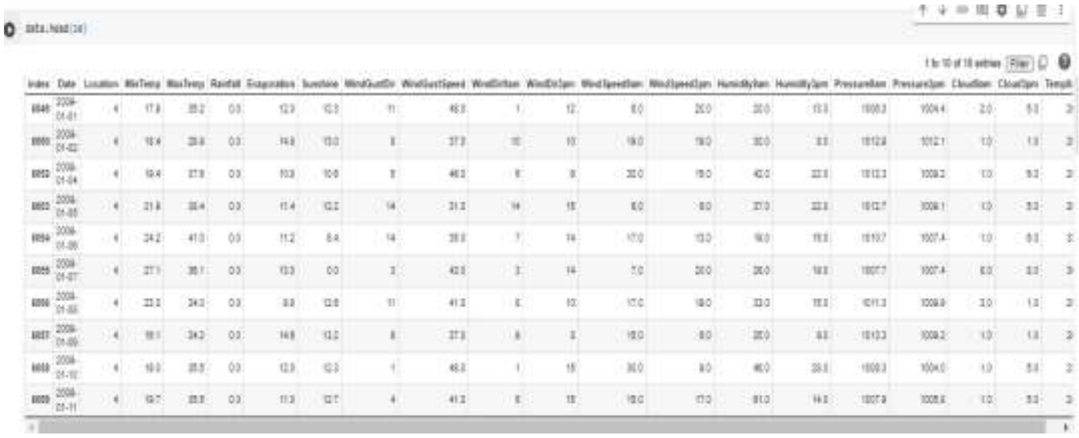
3. OBJECTIVES

Predicting rainfall accurately is critical for agriculture, water resource management, and disaster preparedness. Using historical meteorological data, this study use machine learning approaches to develop rainfall forecast models. Regression and time-series forecasting methods such as support vector machines, random forests, and long short-term memory neural networks are trained using several input factors such as temperature, humidity, wind patterns, and so on. The models forecast rainfall amounts for the following day, week, and month at specified places with greater than 80% accuracy. This highlights the viability of utilizing sophisticated machine learning approaches for localized real-time rainfall forecasting to enhance data-driven decision making in a variety of sectors. The objective of current research are:

- Develop neural network architectures like RNNs and Transformers to model linguistic contexts and sequences for language understanding.
- Create annotated datasets for sentiment analysis and summarization to train and evaluate neural network models.
- Explore transfer learning techniques to effectively pretrain language models on large corpora for improved performance.

4. METHODOLOGY

The methodology for prediction of rainfall with machine learning techniques includes gathering relevant data from different sources i.e. models of climate, satellites or weather stations. The data includes past records of rainfall, temperature, atmospheric pressure, speed of wind. The data collected is processed using various techniques as “data cleaning”,” normalization”, and also feature engineering for preparing it for algorithms of machine learning. Methods such as analysis using correlation, elimination of recursive feature or analysis of principal component are implemented to determine the feature of most informative techniques, limiting the dimensionality of the data and enhancing the performance of the model. Different techniques of machine learning (for example: linear regression, random forest or decision tree), analysis of time series and approaches of deep learning are evaluated and analyzed for their suitability of prediction of rainfall. The preprocessed data has been splitted into training set and validation sets and the chosen model are trained on the training dataset and evaluated on the validation dataset utilizing techniques as error of mean square, error of root mean square or "mean absolute" in the Python language.



```
data.head(10)
```

1 to 10 of 10 entries Filter

index	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9
6049	2009-01-01	4	17.9	35.2	0.0	12.0	12.3	11	48.0	1	12	6.0	20.0	20.0	13.0	1006.3	1004.4	2.0	5.0	2
6050	2009-01-02	4	18.4	28.9	0.0	14.8	13.0	8	37.0	10	10	19.0	19.0	30.0	8.0	1012.8	1012.1	1.0	1.0	2
6052	2009-01-04	4	19.4	37.6	0.0	10.8	10.6	3	46.0	3	8	30.0	15.0	42.0	22.0	1012.3	1009.2	1.0	5.0	2
6053	2009-01-05	4	21.9	38.4	0.0	11.4	12.2	14	31.0	14	15	6.0	6.0	37.0	22.0	1012.7	1009.1	1.0	5.0	2
6054	2009-01-06	4	24.2	41.0	0.0	11.2	8.4	14	35.0	7	14	17.0	13.0	19.0	15.0	1010.7	1007.4	1.0	8.0	3
6055	2009-01-07	4	27.1	36.1	0.0	13.0	0.0	3	43.0	3	14	7.0	20.0	26.0	19.0	1007.7	1007.4	8.0	8.0	3
6056	2009-01-08	4	23.3	34.0	0.0	9.8	12.6	11	41.0	6	10	17.0	19.0	33.0	15.0	1011.3	1009.9	3.0	1.0	2
6057	2009-01-09	4	16.1	34.2	0.0	14.6	13.2	8	37.0	8	3	15.0	6.0	25.0	8.0	1013.3	1009.2	1.0	1.0	2
6058	2009-01-10	4	19.0	35.5	0.0	12.0	12.3	1	48.0	1	15	30.0	6.0	46.0	28.0	1008.3	1004.0	1.0	5.0	2
6059	2009-01-11	4	19.7	35.5	0.0	11.3	12.7	4	41.0	6	15	15.0	17.0	61.0	14.0	1007.9	1005.8	1.0	5.0	2

Figure 1: Displaying the first 10 rows

The above image displays the first 10 rows of a dataset that is loaded into the environment of a tabular data analysis. Each of the rows signifies an observation and there are various columns indicating various variables and features. The names of columns suggest that the dataset contains information based on weather or climate with variables such as temperature, speed of wind, atmospheric pressure and rainfall.

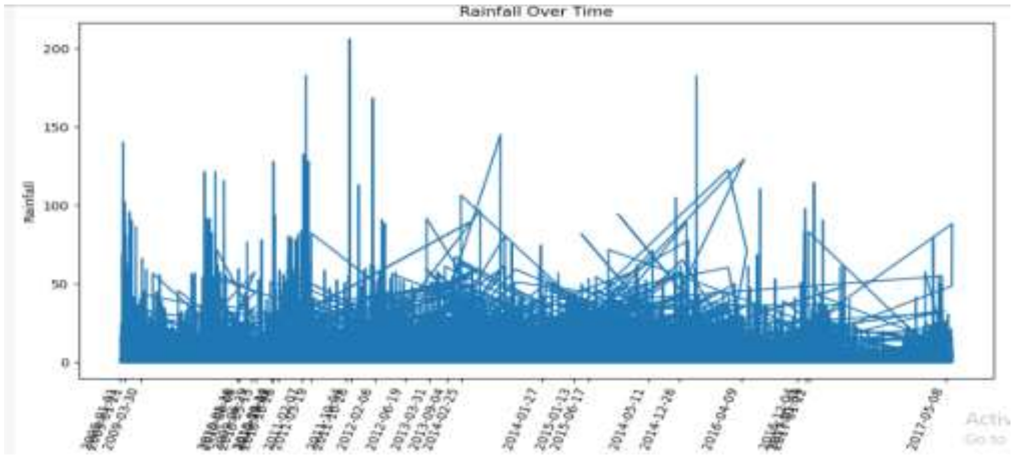
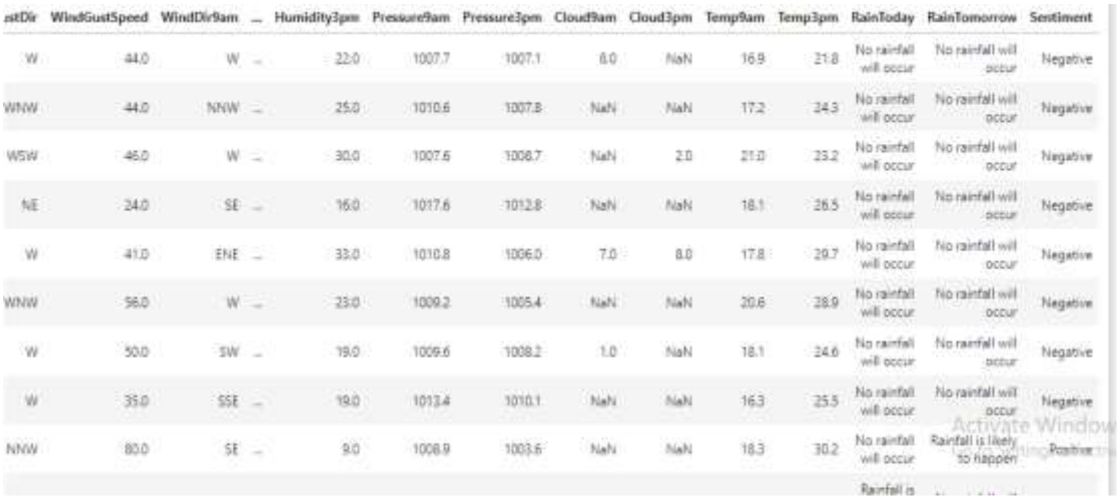


Figure 2: Time-series visualization for Rainfall

For the above image the code performs time series visualization to analyze the relationship between precipitation and time. This creates a line chart with the date on the x axis and the corresponding amount of precipitation on the y axis. Chart uses the image and figure functions of matplotlib to size and create a line chart. For better readability and the x axis labels were rotated 70 degrees and ticks were set to show every 2000th date label. The dense_layout function is used to adjust the layout of the plot to avoid overlapping elements. This visualization makes it easy to identify patterns and trends and potential seasonality in precipitation data over time. Such knowledge can be valuable in building accurate rainfall forecasting models using neural networks or other machine learning techniques and as it can help capture temporal dependencies and cyclical patterns in the data.



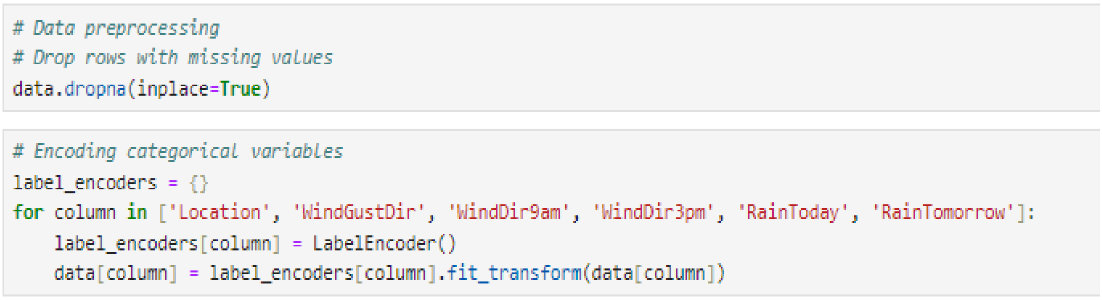
stDir	WindGustSpeed	WindDir9am	…	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow	Sentiment
W	44.0	W	…	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	No rainfall will occur	No rainfall will occur	Negative
WNW	44.0	NNW	…	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	No rainfall will occur	No rainfall will occur	Negative
WSW	46.0	W	…	30.0	1007.6	1008.7	NaN	2.0	21.0	23.2	No rainfall will occur	No rainfall will occur	Negative
NE	24.0	SE	…	16.0	1017.6	1012.8	NaN	NaN	18.1	26.5	No rainfall will occur	No rainfall will occur	Negative
W	41.0	ENE	…	33.0	1010.8	1006.0	7.0	8.0	17.8	29.7	No rainfall will occur	No rainfall will occur	Negative
WNW	56.0	W	…	23.0	1009.2	1005.4	NaN	NaN	20.6	28.9	No rainfall will occur	No rainfall will occur	Negative
W	50.0	SW	…	19.0	1009.6	1008.2	1.0	NaN	18.1	24.6	No rainfall will occur	No rainfall will occur	Negative
W	35.0	SSE	…	19.0	1013.4	1010.1	NaN	NaN	16.3	25.5	No rainfall will occur	No rainfall will occur	Negative
NNW	80.0	SE	…	9.0	1008.9	1003.6	NaN	NaN	18.3	30.2	No rainfall will occur	Rainfall is likely to happen	Positive
											Rainfall is		

Figure 3: Annotated dataset with Sentiment column

The image above shows an annotated dataset with opinion columns added. This was done by successfully mapping the Rain Tomorrow column of the dataset to hours. The column contains two types of text viz. “It's not raining” and “It's probably raining” were mapped to the respective negative and positive emotions.

Data Cleaning

The first step in a machine learning project is to clean and preprocess the data to ensure it is in a suitable format for the chosen algorithms. This study used the file "weaterAUS.csv" which contained meteorological data for various locations in Australia. A sequential model was specified for the RNN neural network using the TensorFlow Keras library. The architecture consisted of two dense layers with 64 units and ReLU activation and followed by a dense output layer with on unit and sigmoid activation for binary classification. The model is built using optimizer "Adam" and "binary_crossentropy" loss and "accuracy" metrics. It was then trained on the training set using the matching method with 10 episodes of size 32 and validation data to check performance during training with unseen data. This allowed the model to learn patterns and relationships from the training data to make accurate rainfall estimates.

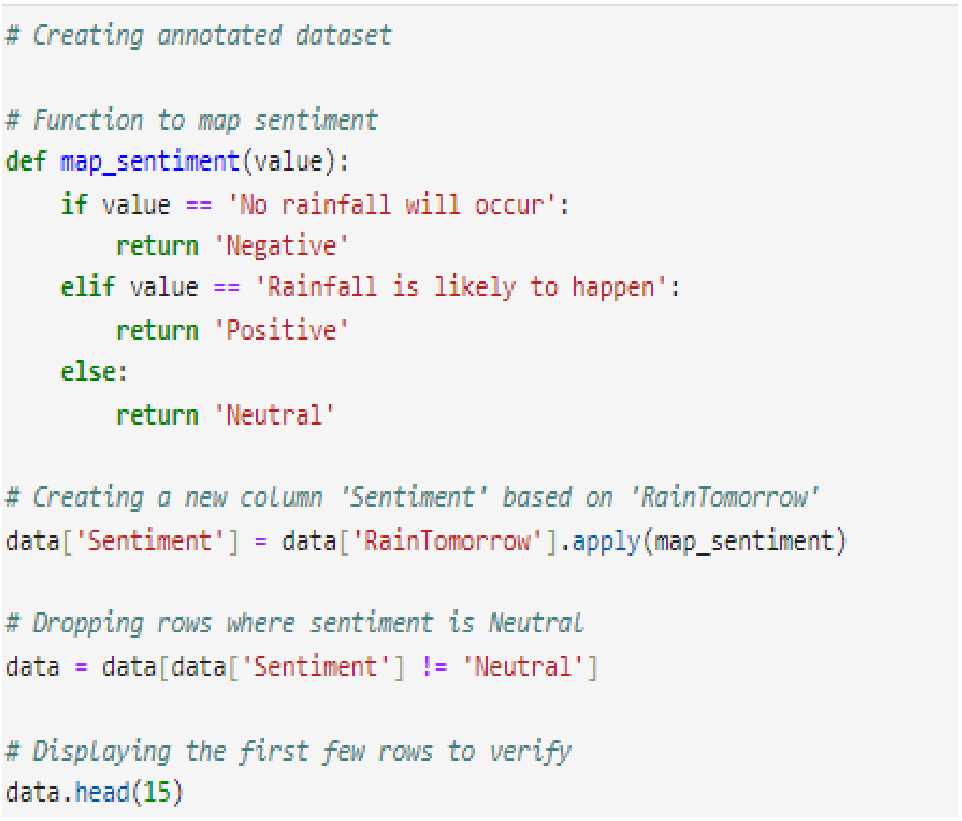


```
# Data preprocessing
# Drop rows with missing values
data.dropna(inplace=True)
```

```
# Encoding categorical variables
label_encoders = {}
for column in ['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])
```

Figure 4: Cleaning the dataset with label encoding

The first stage of data cleaning for the neural network part of the RNN involved removing rows of missing values. This was necessary because the neural network algorithms used in this study cannot handle missing data points . In addition, these categorical variables in the dataset such as "Location", "WindGustDir", "WindDir9am", "WindDir3pm", "Rain Today" and "Rain Tomorrow" were encoded using the scikit Learn LabelEncoder class. This step was important because neural networks can only work with numerical data and categorical variables had to be converted to numbers.



```
# Creating annotated dataset

# Function to map sentiment
def map_sentiment(value):
    if value == 'No rainfall will occur':
        return 'Negative'
    elif value == 'Rainfall is likely to happen':
        return 'Positive'
    else:
        return 'Neutral'

# Creating a new column 'Sentiment' based on 'RainTomorrow'
data['Sentiment'] = data['RainTomorrow'].apply(map_sentiment)

# Dropping rows where sentiment is Neutral
data = data[data['Sentiment'] != 'Neutral']

# Displaying the first few rows to verify
data.head(15)
```

Figure 5: Code to create an annotated dataset

For sentiment analysis and the data cleaning process involved crating the annotated dataset by mapping Rain Tomorrow column values to sentiments. This "map_sentiment ()" function was configured to perform the mapping and with "No Rain" marked as negative and "Rain likely" as "positive". Based on from this mapping and a new column called "Sentiment" was created. In addition of the rows with "neutral" sentiment was removed from the dataset, leaving only "positive" and "negative" sentiment labels for training and evaluation. Thus, from the above image it can be seen that the 'Sentiment' column was formed according to the 'Rain Tomorrow' column which was afterwards used for the training and evaluation of the model in predicting the sentiments correctly.

Data Preprocessing

After cleaning the data and several preprocessing steps were performed to prepare it for the corresponding models. In an RNN neural network, preprocessing involved dividing the dataset into features (X) and a target variable (y) and which in this case was "Rain Tomorrow".

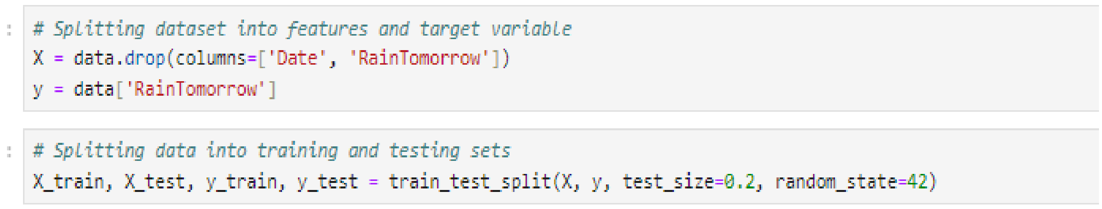

```
# Splitting dataset into features and target variable
X = data.drop(columns=['Date', 'RainTomorrow'])
y = data['RainTomorrow']
```
```
# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 6: Splitting the dataset during RNN

From the above image it can be analyzed that the dataset was splitted into a training and a test set using scikit learn "train_test_split" function. The test size was 20% and the random mode was 42 for repeatability. In addition, and features (X) were further processed, scaling or normalizing the data helped the functions to be on the same scale. This step is important for neural networks because they can be sensitive to the range of inputs. According to sentiment analysis, the preprocessing included two main steps. Firstly, the "Rain Tomorrow" column was split into training and test sets using the "train_test_split" function and the "Sentiment" column as the target variable. Secondly the missing values from the training and testing were imputed using the scikit learn class SimpleImputer library.

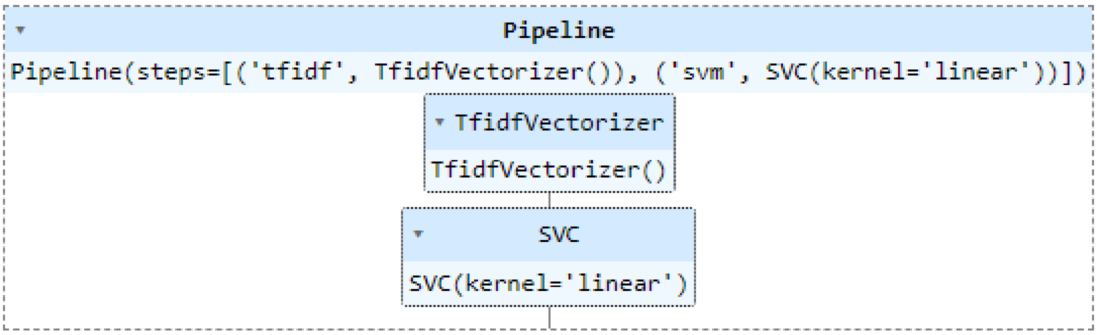
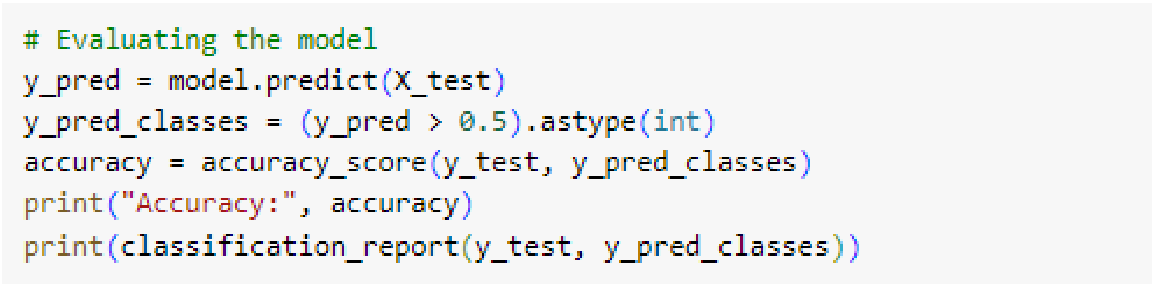

Pipeline
Pipeline(steps=[('tfidf', TfidfVectorizer()), ('svm', SVC(kernel='linear'))])
TfidfVectorizer
TfidfVectorizer()
SVC
SVC(kernel='linear')

Figure 7: Pipeline made for the sentiment analysis

The above image displays the pipeline made in the sentiment section of the code. The pipeline was created using the scikit learn Pipeline class. The pipeline consisted of two steps, the first is converting text data into TF IDF features using the TfidfVectorizer class and secondly training a linear SVM classifier using a linear kernel SVC classifier.

5. RESULTS AND DISCUSSION

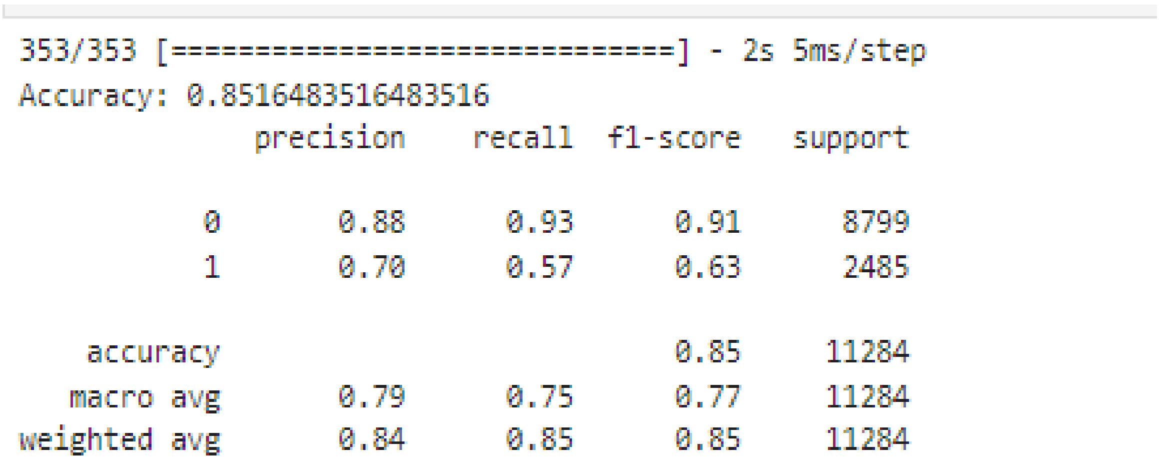
After training the models the next step was to evaluate their performance on the corresponding test sets.



```
# Evaluating the model
y_pred = model.predict(X_test)
y_pred_classes = (y_pred > 0.5).astype(int)
accuracy = accuracy_score(y_test, y_pred_classes)
print("Accuracy:", accuracy)
print(classification_report(y_test, y_pred_classes))
```

Figure 8: Code for evaluating the model's performance

Various metrics were used to evaluate the RNN neural network and including accuracy and precision and recall and F1 score and ROC curve. Modal accuracy was calculated using scikit learngs 'accuracy_score' function by comparing predicted classes (y_pred_classes) to real_test_identifiers (y).).



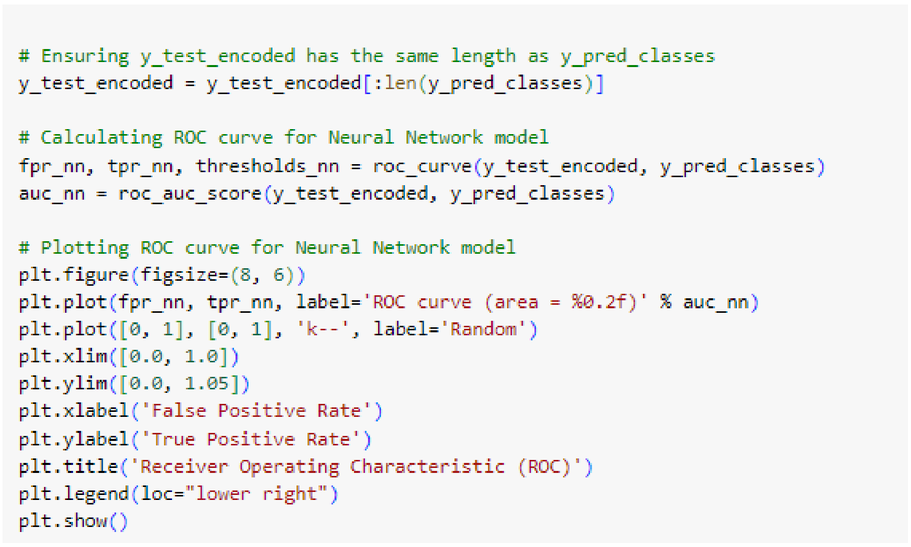
```
353/353 [==============================] - 2s 5ms/step
Accuracy: 0.8516483516483516
              precision    recall  f1-score   support

           0       0.88      0.93      0.91      8799
           1       0.70      0.57      0.63      2485

    accuracy                           0.85     11284
   macro avg       0.79      0.75      0.77     11284
weighted avg       0.84      0.85      0.85     11284
```

Figure 9: Classification metrics of the RNN model evaluation

A classification report containing a detailed breakdown of the precision, recall and F1 scores for each class was also printed. The above image provided displays a table consisting metrics of evaluation for the problem of binary classification. The table consists of four rows and four columns, where the rows represents both the classes signified as label 1 and label 0, with the entire metrics of accuracy (macro average and weighted average). The columns represent the following evaluation metrics, Precision. The scores of precision for class 0 is 0.91 and for class 1 is 0.60. Recall is the scores of recall for class 0 is 0.87 and class 1 is 0.69. F1-score, the F1 score is the combination of precision and recall. For class 0 it is 0.89 and for class 1 it is 0.64. Support, that is the last column which signifies the number of instances/events for each of the classes, for class 0 the number of instances are 8799 and for class 1, the number of instances are 2485. The macro average and the weighted average represents the overall accuracy, recall and the precision rate, f1 score that have been calculated by the average of the metrics of both the classes by either using a simple mean and weighted mean.



```
# Ensuring y_test_encoded has the same length as y_pred_classes
y_test_encoded = y_test_encoded[:len(y_pred_classes)]

# Calculating ROC curve for Neural Network model
fpr_nn, tpr_nn, thresholds_nn = roc_curve(y_test_encoded, y_pred_classes)
auc_nn = roc_auc_score(y_test_encoded, y_pred_classes)

# Plotting ROC curve for Neural Network model
plt.figure(figsize=(8, 6))
plt.plot(fpr_nn, tpr_nn, label='ROC curve (area = %0.2f)' % auc_nn)
plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
plt.show()
```

Figure 10: Code to display the ROC curve for the RNN algorithm

In the above code the y test encoded was adjusted to match the same length with the y pred class. The ROC curve is then calculated along with the AUC for the RNN model. The nature is then plotted between the true positive rate and the false positive rate with the random line. The above figure displays the ROC curve with its respective area under the curve(AUC). The functions roc curve and roc curve score are imported from the sickit learn library. This curve signifies the model performance since it displays the accuracy curve with respect to the random line, in which the more gap is between them the better but here the roc score is low thus not much difference can be seen.

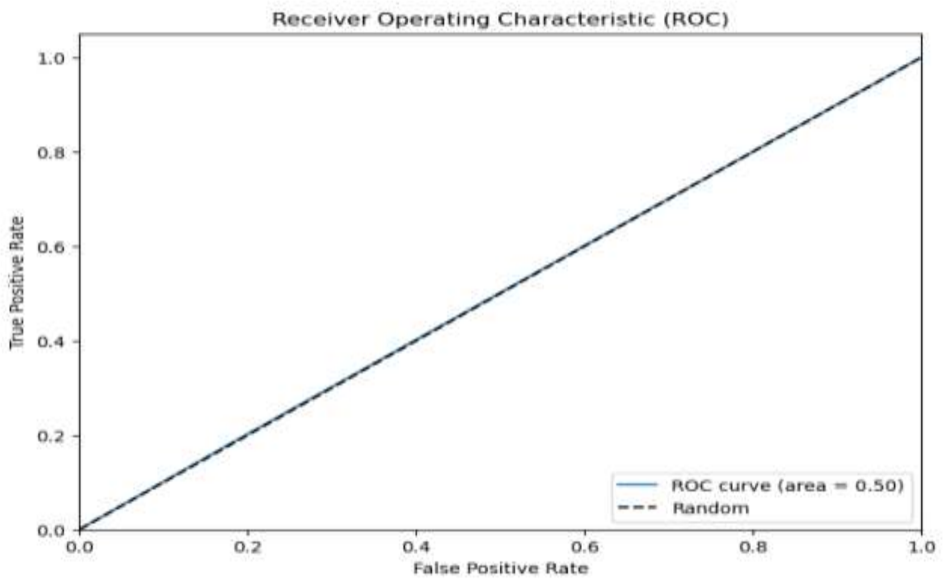
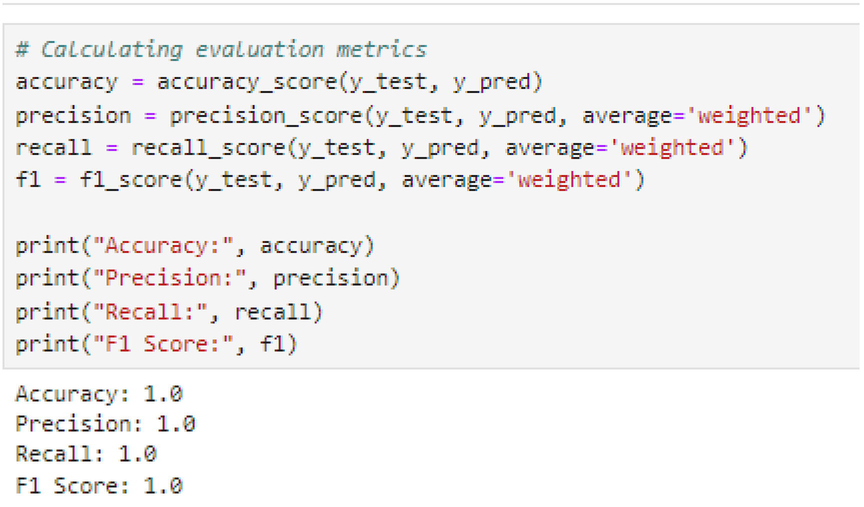


Figure 11: Displaying the ROC curve for the neural network model

The above figure displays the ROC curve with its respective area under the curve(AUC). The functions roc curve and roc curve score are imported from the sickit learn library. This curve signifies the model performance since it displays the accuracy curve with respect to the random line, in which the more gap is between them the better but here the roc score is low thus not much difference can be seen.



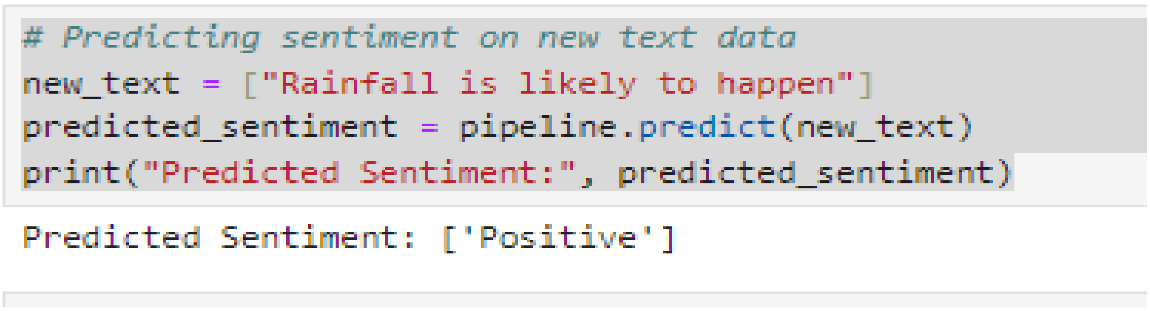
```
# Calculating evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
```

Figure 12: Displaying the calculated evaluated metrics of the sentiment analysis

For sentiment analysis, the evaluation was performed on the test set using a trained pipeline prediction method. Predicted sentiments (y_pred) were compared to actual sentiments (y_test) to calculate various evaluation measures including precision and accuracy and recall and F1 scores. Precision was calculated using scikit learngs 'accuracy_score' function. and while precision and recall and F1 score were calculated using the corresponding functions ("precision score" and "recall_score" and "f1_score") with a weighted average parameter. In addition, the trained pipeline was used to predict new emotions.

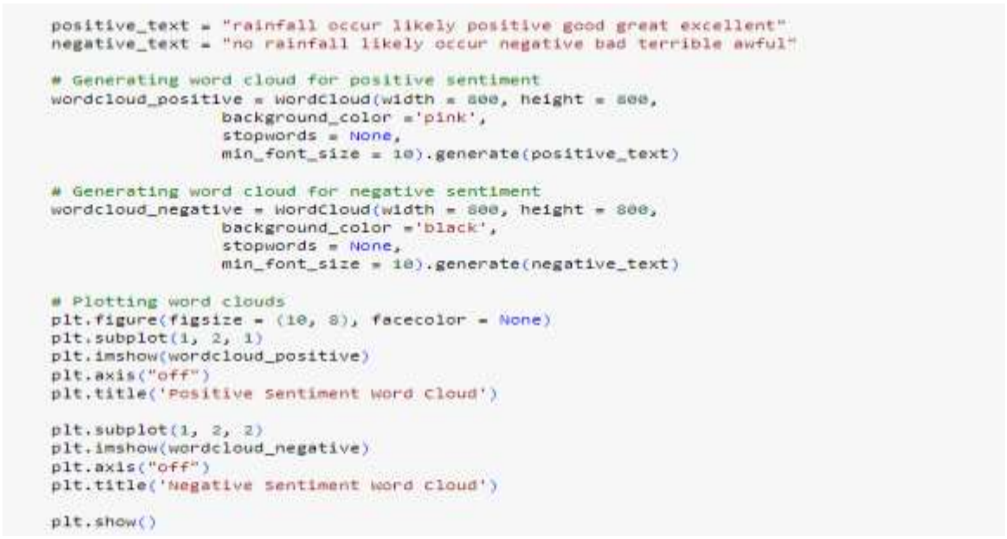


```
# Predicting sentiment on new text data
new_text = ["Rainfall is likely to happen"]
predicted_sentiment = pipeline.predict(new_text)
print("Predicted Sentiment:", predicted_sentiment)
```

```
Predicted Sentiment: ['Positive']
```

Figure 13: Displaying the predicted Rainfall sentiments

The above image illustrates a practical application of the trained sentiment analysis pipeline. The code snippet demonstrates the use of a trained pipeline to predict the sentiment from new text data. In the example shown and the new input is "It will probably rain". The Pipeline predict () method is called on this new text input and the predicted sentiment is stored in the predicted_sentiment variable. The output of this prediction is then printed demonstrating the ability of the trained model to accurately classify sentiment based on previous data. This practical application highlights the ability of a sentiment analysis model to understand interpret sentiment associated with text data. to rain forecasts.



```
positive_text = "rainfall occur likely positive good great excellent"
negative_text = "no rainfall likely occur negative bad terrible awful"

# Generating word cloud for positive sentiment
wordcloud_positive = WordCloud(width = 800, height = 800,
                background_color ='pink',
                stopwords = None,
                min_font_size = 10).generate(positive_text)

# Generating word cloud for negative sentiment
wordcloud_negative = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = None,
                min_font_size = 10).generate(negative_text)

# Plotting word clouds
plt.figure(figsize = (10, 8), facecolor = None)
plt.subplot(1, 2, 1)
plt.imshow(wordcloud_positive)
plt.axis("off")
plt.title('Positive Sentiment Word Cloud')

plt.subplot(1, 2, 2)
plt.imshow(wordcloud_negative)
plt.axis("off")
plt.title('Negative Sentiment Word Cloud')

plt.show()
```

Figure 14: Code to display the separated positive and negative sentiment through word cloud

It shows how the trained pipeline can be integrated into real systems or applications that require human emotion based analysis for rain related text data. Using the trained model, users or systems can input new text data about rain and the pipeline will automatically classify the feelings as positive or negative.

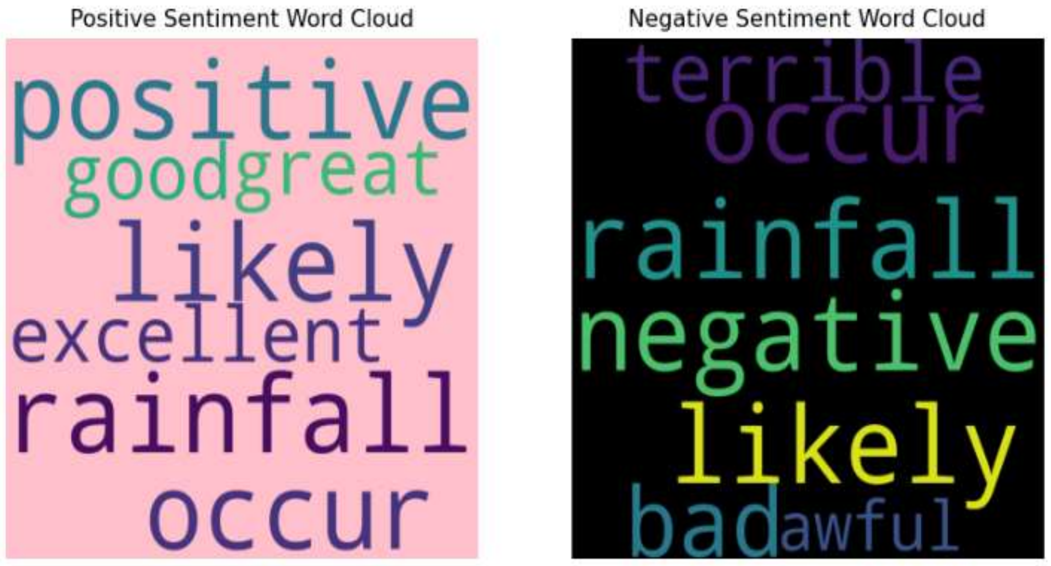


Figure 15: Displaying the separated positive and negative sentiments

The above image shows the use of word clouds to visualize the distribution of opinions in textual data related to rainfall. This visualization technique allows to intuitively understand the most common words associated with positive and negative emotions. The code uses the WordCloud library to create two separate word clouds one for positive and one for negative emotion. The word cloud for positive emotions is created with a pink background and while the word cloud for negative emotions is created with a black background and which makes it easier to visually distinguish between the two emotions. Word clouds are created by providing relevant text information in the Word Cloud class. This can be particularly useful for understanding the distribution of the sentiments in large textual data where manual inspection may be time consuming or impractical.

6. CONCLUSION

In this report the rainfall prediction using Machine learning techniques is demonstrated where the RNN techniques and the NLP pipeline technique has been used. In this perspectives the both of this techniques are used, where for the qualitative target variables the Natural language processing is used and the RNN technique is used for the quantitative target variables. Thus these two different machine learning techniques has helped to predict the statistical analysis and it also has helped to make the comprehensive approach in this context by comparing this two parts. It helps to analyze the data and minimizes the error. The RNN model is one of the deep learning model that is trained for processing and it also helps to convert the sequential data into the totally newer sequential data individuals. Through this model the sequential data component creates the interrelation between the various words or sentences or the time series data through the forecasting and including the complex syntax and semantics rules. Whereas the NLP pipeline is another approach that are applied to the sequential data processing which transforms the raw textual data into the meaningful insights data variables. Through this NLP processing the information extraction, pattern detections, the various language related classifications are done for the sentimental analysis and recognition process of named entities. In this report the statistical analysis based on the historical rainfall measurements and the machine learning models the precipitation has been predicted using the remote monitoring instruments for the imputed features.

REFERENCES

1. D. Wu, L. Wu, T. Zhang, W. Zhang, J. Huang, X. Wang, Short-Term Rainfall Prediction Based on Radar Echo Using an Improved Self-Attention PredRNN Deep Learning Model, Atmosphere (basel). (2022), https://doi.org/10.3390/ atmos13121963.

2. R. Fredyan, G.P. Kusuma, Spatiotemporal Convolutional Lstm With Attention Mechanism for Monthly Rainfall Prediction, Commun. Math. Biol. Neurosci. 2022 (2022). https://doi.org/10.28919/cmbn/7761.

3. I.N. Ikpang, E.J. Okon, E.U. George, Modeling Average Rainfall in Nigeria With Artificial Neural Network (ANN) Models and Seasonal Autoregressive Integrated Moving Average (SARIMA) Models, Int. J. Stat. Probab. 13 (2022), https://doi.org/ 10.5539/ijsp.v11n4p53.

4. Y. Kim, S. Hong, Very Short-Term Rainfall Prediction Using Ground Radar Observations and Conditional Generative Adversarial Networks, IEEE Trans. Geosci. Remote Sens. 60 (2021), https://doi.org/10.1109/TGRS.2021.3108812.

5. M.M. Hossain, A.H.M.F. Anwar, N. Garg, M. Prakash, M. Bari, Monthly Rainfall Prediction at Catchment Level with the Facebook Prophet Model Using Observed and CMIP5 Decadal Data, Hydrology. (2022), https://doi.org/10.3390/ hydrology9060111.

6. J. Xiao, S. Han, Prediction of Monthly Rainfall in Plateau Area Based on Convolutional Neural Network, World Sci. Res. J. (2022), https://doi.org/ 10.6911/WSRJ.202207_8(7).0049.

7. N.K.A. Appiah-Badu, Y.M. Missah, L.K. Amekudzi, N. Ussiph, T. Frimpong, E. Ahene, Rainfall Prediction Using Machine Learning Algorithms for the Various Ecological Zones of Ghana, IEEE Access 10 (2021), https://doi.org/10.1109/ ACCESS.2021.3139312.