

INFO 550: Artificial Intelligence

Final Project

Submitted by: Shashwat Shrivastava

Github Repository link: <https://github.com/shashwatstva/INFO-550-AI-Final-Project.git>

Comparative Analysis of Weather Prediction Models Using Bayesian Networks and RNN LSTM

I. Project Overview

This project aims to compare the performance of two distinct approaches for weather prediction: Bayesian Networks utilizing Maximum Likelihood Estimation (MLE) and Variable Elimination, and Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) architecture. Two datasets were employed for training and evaluation, one comprising 500 records and the other encompassing the entire dataset of 1461 records. The study evaluates and contrasts the predictive capabilities of these models across the varying dataset sizes.

II. AI Concepts

Bayesian Networks:

Bayesian networks are probabilistic graphical models representing dependencies between variables using a directed acyclic graph (DAG). They consist of nodes representing variables, edges denoting probabilistic dependencies, and conditional probability distributions (CPDs) specifying probabilities. These networks facilitate efficient probabilistic reasoning and inference by propagating evidence through the graph.

In our Bayesian model approach, we utilize Bayesian networks to capture the complex relationships between weather variables and make predictions based on observed evidence.

Maximum Likelihood Estimation (MLE):

Maximum Likelihood Estimation (MLE) is a statistical method used to estimate the parameters of a probability distribution by maximizing the likelihood function. In our Bayesian model implementation, we employ MLE to learn the parameters of the Bayesian network from the training data. This involves estimating the conditional probabilities of each variable given its parent variables in the network structure.

Probabilistic Inference Method- Variable Elimination

Probabilistic inference is the process of computing the posterior probability distribution of target variables given observed evidence in a Bayesian network. We utilize probabilistic inference techniques, such as Variable Elimination, to make predictions about weather classes based on the learned Bayesian network model.

Variable Elimination: Variable elimination is a method used for performing probabilistic inference in Bayesian networks. It systematically eliminates variables from the network to compute the posterior probability distribution of target variables given observed evidence.

Recurrent Neural Networks (RNNs):

Recurrent Neural Networks (RNNs) are neural networks designed to handle sequential data by maintaining internal state or memory. They process data iteratively, retaining information about previous time steps through directed cycles.

Long Short-Term Memory (LSTM) networks, is a type of recurrent neural network (RNN) architecture that's particularly effective for processing and modeling sequential data. LSTM networks are designed to address the vanishing gradient problem, which is a common issue with traditional RNNs that makes it difficult for them to learn and retain information over long sequences.

With specialized architectures like Long Short-Term Memory (LSTM) networks, RNNs can effectively capture long-term dependencies in sequential data, allowing us to make accurate predictions about future weather events based on historical patterns.

Deep Learning Methods (Activation Functions, Loss Functions):

1. **Activation Functions:** Activation functions introduce non-linearity into neural networks, enabling them to learn complex patterns in the data.

Activation functions are applied to the output of each neuron in the network's hidden layers, transforming the input into the neuron's output signal. This transformed output is then passed on to the next layer of the network for further processing.

- **ReLU (Rectified Linear Unit):** ReLU is an activation function commonly used in neural networks to introduce non-linearity. It computes the output as the maximum of zero and the input value, helping neural networks learn complex patterns in the data.

Rectified Linear Unit (ReLU) activation function helps our RNNs model nonlinear relationships between weather variables.

2. Loss functions

Loss functions are a critical component of deep learning algorithms, as they quantify the difference between the predicted output of the model and the true labels or targets. The goal during training is to minimize this loss function, which effectively means minimizing the discrepancy between the model's predictions and the ground truth.

They quantify how well our model is performing during training by measuring the difference between predicted and actual outcomes.

- **Sparse Categorical Crossentropy:** Sparse categorical crossentropy is a loss function utilized in multi-class classification tasks where the target labels are integers. Unlike one-hot encoded vectors, it calculates the cross-entropy between the true distribution and the predicted distribution of class probabilities.

Encoding Techniques:

Label encoder and one-hot encoder are encoding techniques used to convert categorical variables into numerical format suitable for machine learning algorithms. Label encoder assigns a unique integer to each category, while one-hot encoder creates binary vectors where each element represents a category, facilitating efficient learning from categorical data.

III. Code Description

A. Dataset Description

Weather prediction dataset is used in building the models.

- **Data Source: Weather Prediction Dataset**
<https://www.kaggle.com/datasets/ananthr1/weather-prediction/data>
- The dataset contains 1461 records and 6 columns
 - 1) Date
 - 2) Precipitation
 - 3) temp_max
 - 4) temp_min
 - 5) wind
 - 6) weather
- The dataset does not contain any missing values.

```
C:\Users\shash\AppData\Local\Programs\Python\Python311\python.exe "C:\AI 550\Scripts\preprocessing.py"
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1461 entries, 0 to 1460
Data columns (total 6 columns):
#   Column          Non-Null Count  Dtype
---  -
0   date            1461 non-null   object
1   precipitation    1461 non-null   float64
2   temp_max        1461 non-null   float64
3   temp_min        1461 non-null   float64
4   wind            1461 non-null   float64
5   weather         1461 non-null   object
dtypes: float64(4), object(2)
memory usage: 68.6+ KB

Process finished with exit code 0
|
```

Figure: Figure showing the information related to the original dataset- seattle-weather.csv

B. Libraries used:

- **time:** Used for measuring execution time.
- **numpy (np):** Used for numerical computations and array operations.
- **pandas (pd):** Used for data manipulation and analysis.
- **sklearn:** Used for machine learning tasks such as data preprocessing, model evaluation, and model selection.
- **tensorflow (tf):** Used for building and training neural network models.
- **pgmpy:** Used for constructing, training, and inference with Bayesian networks.
- **tensorflow (tf):** Used for building and training the RNN model.
- **sklearn.preprocessing:** Utilized for data preprocessing tasks such as label encoding and feature scaling.
- **sklearn.model_selection:** Used for splitting the dataset into training and testing sets.
- **networkx: (Commented out)** Used for the creation, manipulation, and study of complex networks or graphs.

Please Note that pgmpy is an external library which is used for constructing, training, and performing inference with Bayesian networks.

Note2: networkx was used to generate the Bayesian Network graph. It is commented out in the python file.

C. Files Present in the folder

1. **seattle-weather.csv:** Raw weather data downloaded from the data source.
2. **Preprocessing.py:** This script takes seattle-weather.csv as an input. Preprocessing is done. This script generates a random sample of 500 records from the raw data and saves it as a .csv file- **subset.csv (Dataset 1)**. The original data is saved as **alldata.csv**.
3. **subset.csv:** Output of preprocessing.py. It is used as Dataset 1 having 500 records. This is used for training both the models.
4. **alldata.csv:** Output of preprocessing.py. It is used as Dataset 2 having all records - 1461. This is used for training both the models.
5. **RNN_subset.py and RNN_alldata.py:** Both files follow the same steps for RNN LSTM model. RNN_subset.py is used for dataset 1 (500 rows) and the other is for dataset 2 (all data- 1461 records).
6. **Bayesian_subset.py and Bayesian_alldata.py:** Both files follow the same steps for Bayesian Network model. Bayesian_subset.py is used for dataset 1 (500 rows) and the other is for dataset 2 (all data- 1461 records).
7. **bn_subset_graph.png and bn_all_graph.png:** Graph generated by Bayesian Model
8. **RNN_subset_architecture.png and RNN_alldata_architecture.png:** RNN architecture generated by RNN Model.

D. Specific Models Code

1. Bayesian Network Code:

There are two files - **Bayesian_subset.py** and **Bayesian_alldata.py**. Both files follow the same steps. **Bayesian_subset.py** is used for dataset 1 (500 rows) and the other is for dataset 2 (all data- 1461 records).

The code structure is as follows:

- **Data Preprocessing:**
 - I. Loading the dataset from a CSV file.
 - II. Converting the 'date' column to datetime format.
 - III. Dropping rows with missing values.
- **Model Construction:**
 - I. Defining the structure of the Bayesian network.
 - II. The Bayesian network structure is specified with nodes representing variables and edges representing dependencies between them.
- **Model Training:**
 - I. Splitting the dataset into training and testing sets using `train_test_split` function.
 - II. Learning the parameters of the Bayesian network using Maximum Likelihood Estimation (MLE).
 - III. The 'fit' method of the Bayesian network model is called with training data and MLE as the estimator.
- **Prediction:**
 - I. Extracting the relevant columns for prediction from the test data.
 - II. Converting the extracted data into a dictionary format.
 - III. Performing probabilistic inference to make predictions using Variable Elimination method.
 - IV. Computing prediction time complexity.
 - V. Evaluating the Bayesian Network model:
 - 1) Calculating accuracy, log loss, and classification report metrics.
 - 2) Metrics include `accuracy_score`, `log_loss`, and `classification_report` from `sklearn.metrics`.
- **Saving the Bayesian Network graph as a .png file:** This part of the code has been commented out

2. RNN (LSTM) Code:

There are two files - **RNN_subset.py** and **RNN_alldata.py**. Both files follow the same steps. **RNN_subset.py** is used for dataset 1 (500 rows) and the other is for dataset 2 (all data- 1461 records).

- **Data Preprocessing:**
 - I. Loading the dataset from the specified CSV file.

- II. Converting the 'date' column to datetime format.
- III. Handling missing values by dropping rows with missing data.
- IV. Encoding the categorical variable 'weather' using label encoding.
- V. Normalizing the numerical variables 'precipitation', 'temp_max', 'temp_min', and 'wind' using StandardScaler.
- **RNN Specific:**
 - I. Defining a function split_data() to split the dataset into input features (X) and target variable (y).
 - II. Calling split_data() to obtain the input features (X) and target variable (y).
 - III. Reshaping the input features (X) for the RNN model to have the shape (samples, time steps, features).
 - IV. Splitting the dataset into training and testing sets using train_test_split().
 - V. Determining the number of output classes based on the unique labels in the target variable.
- **RNN Model Construction:**
 - I. Defining the RNN architecture using TensorFlow's Keras Sequential API.
 - II. Adding an LSTM layer with 64 units and 'relu' activation function as the input layer.
 - III. Adding a Dense output layer with a softmax activation function to predict the output classes.
- **Model Training:**
 - I. Compiling the RNN model with 'adam' optimizer and 'sparse_categorical_crossentropy' loss function.
 - II. Training the model on the training data with 10 epochs and a batch size of 32.
 - III. Validating the model's performance using 20% of the training data.
 - IV. Measuring the total execution time of the RNN model training.
- **Model Evaluation:**
 - a) Predicting the probabilities for the test data using the trained RNN model.
 - b) Obtaining the predicted classes by selecting the class with the highest probability.
 - c) Evaluating the model's performance using accuracy_score, log_loss, and classification_report.
- **Saving the RNN Architecture as a .png file:** This part of the code has been commented out.

IV. Programming Challenges Faced:

- a) **Execution Time in Bayesian Network:** As you can see from Figure 7 and Figure 9, The time complexity for training increases with the increase in sample size. This resulted in a long execution time.
- b) **Lack of certain classes/ labels (weather) in predicted samples.**

```
71: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
```

- c) **Poor performance of Bayesian network model:** As evident from the Bayesian Test Results, the model performance was very poor.
- d) **Initially, faced issues while implementing Variable Elimination.**

The screenshot shows the PyCharm IDE's debugger window. The top bar indicates the current file is `BayesianCopy`. The main pane displays the execution stack during a debugging session.

```
import sys; print('Python %s on %s' % (sys.version, sys.platform))
C:\Users\shash\AppData\Local\Programs\Python\Python311\python.exe "C:/Program Files/JetBrains/PyCharm 2023.2.1/plugins/python/helpers/pydev/pydevd.py"
Connected to pydev debugger (build 232.9559.58)
Evidence: {'precipitation': 0.0, 'temp_max': 12.8, 'temp_min': 5.0, 'wind': 4.7}
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
Evidence: {'precipitation': 2.5, 'temp_max': 4.4, 'temp_min': 2.2, 'wind': 2.2}
WARNING:pgmpy:Found unknown state name. Trying to switch to using all state names as state numbers
>>> Traceback (most recent call last):
  File "C:\Program Files\JetBrains\PyCharm 2023.2.1\plugins\python\helpers\pydev\pydevd.py", line 1500, in _exec
    pydev_imports.execfile(file, globals, locals) # execute the script
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Program Files\JetBrains\PyCharm 2023.2.1\plugins\python\helpers\pydev\pydevimps_pydev_execfile.py", line 18, in execfile
    exec(compile(contents+"\n", file, 'exec'), glob, loc)
  File "C:\AI 550\Scripts\BayesianCopy.py", line 45, in <module>
    predicted_weather = inference.map_query(variables=['weather'], evidence=evidence.to_dict())
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\shash\AppData\Local\Programs\Python\Python311\Lib\site-packages\pgmpy\inference\ExactInference.py", line 573, in map_query
    final_distribution = reduced_ve._variable_elimination(
    ^^^^^^^^^^^^^^^^^^
>>>
```

Figure: Figure showing initial errors encountered while implementing Variable Elimination.

V. Test Results

1. Recurrent Neural Networks (LSTM) Model:

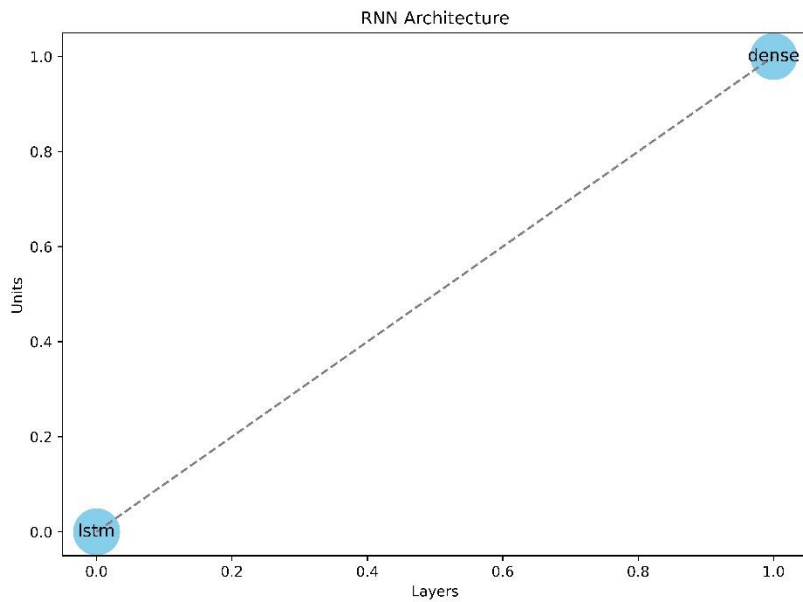


Figure1: RNN architecture in the weather prediction model.

Dataset 1 (500 records)

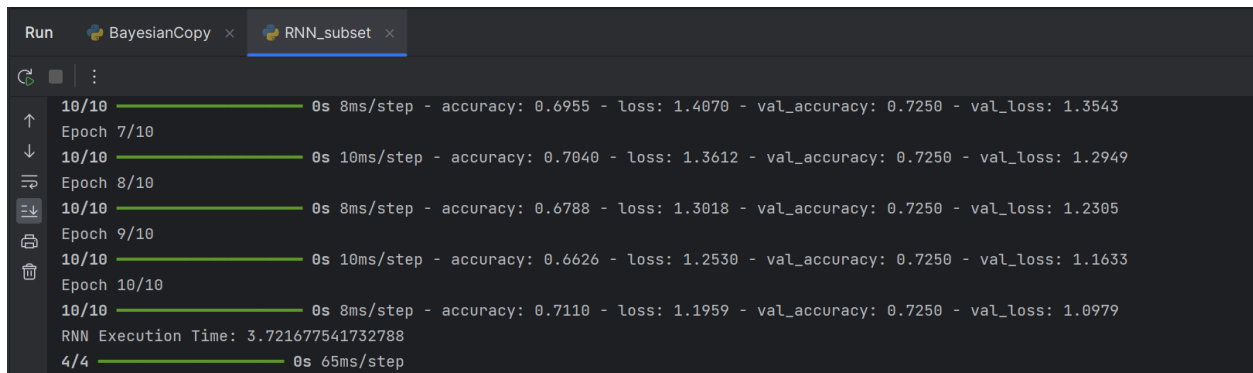


Figure 2: Figure showing the RNN Execution time to be 3.72 seconds

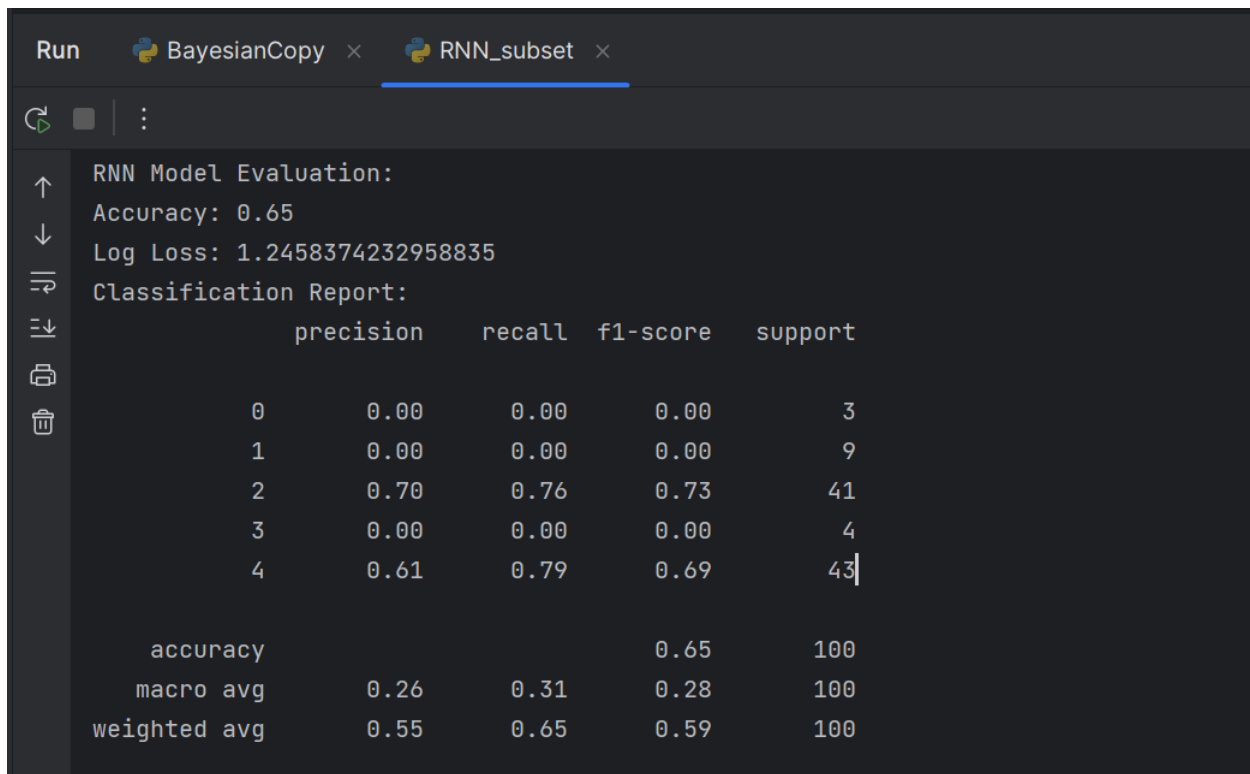


Figure 3: Figure showing the Accuracy, Log Loss and Classification Report.

Dataset 2 (all data)

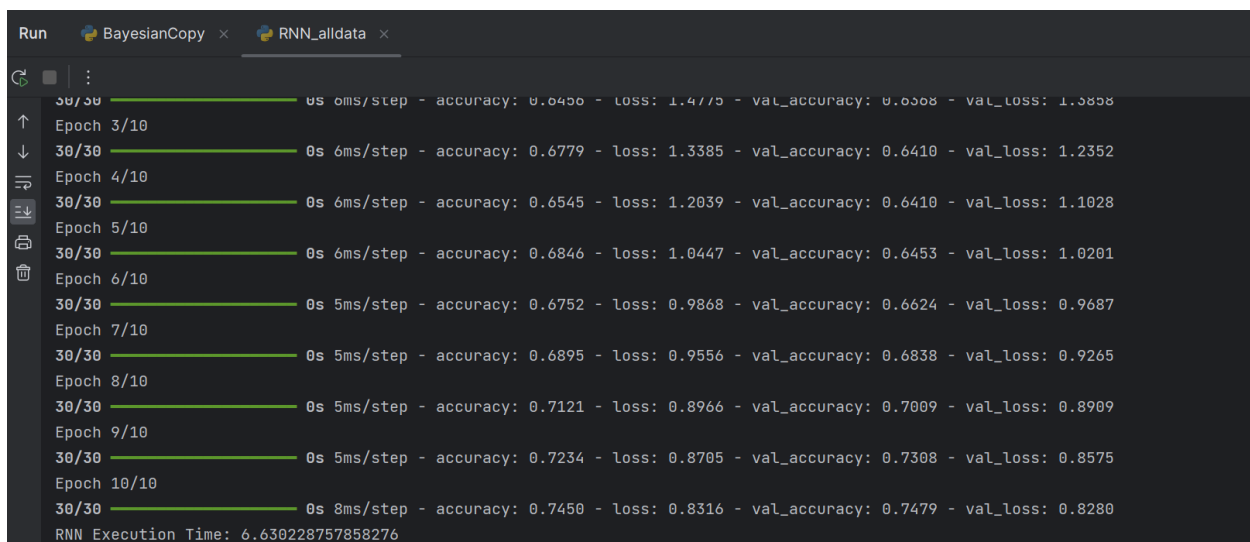
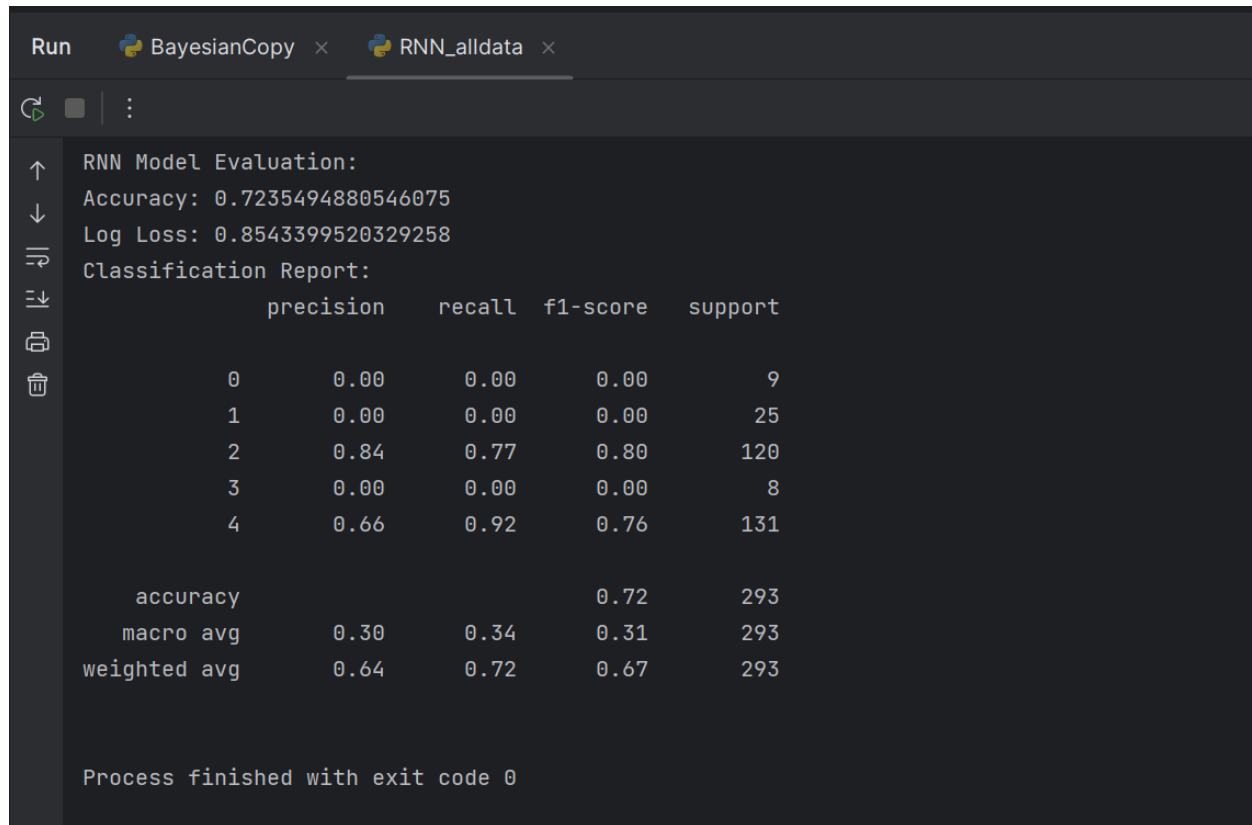


Figure 4: Figure showing the RNN Execution time to be 6.63 seconds



The screenshot shows a terminal window with two tabs: 'BayesianCopy' and 'RNN_alldata'. The 'RNN_alldata' tab is active, displaying the following output:

```
RNN Model Evaluation:
Accuracy: 0.7235494880546075
Log Loss: 0.8543399520329258
Classification Report:
              precision    recall  f1-score   support

     0           0.00       0.00       0.00         9
     1           0.00       0.00       0.00        25
     2           0.84       0.77       0.80       120
     3           0.00       0.00       0.00         8
     4           0.66       0.92       0.76       131

   accuracy          0.72         293
  macro avg          0.30         293
 weighted avg          0.64         293

Process finished with exit code 0
```

Figure 5: Figure showing the Accuracy, Log Loss and Classification Report.

2. Bayesian Network Model

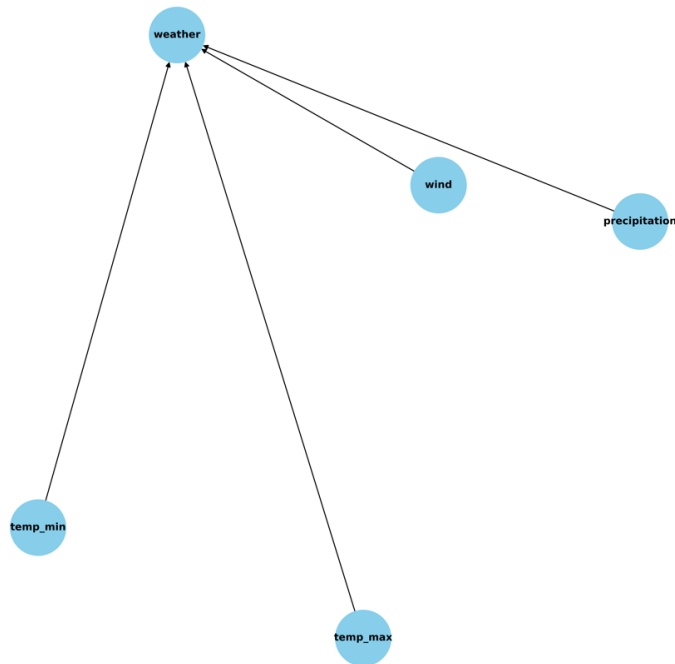


Figure 6: Graph generated by the Bayesian Network Model.

Dataset 1 (500 records)

```

Run  workingbayesian  Bayesian_subset
C:\Users\shash\AppData\Local\Programs\Python\Python311\python.exe "C:\AI 550\Scripts\Bayesian_subset.py"
Time Complexity: 0.0009968280792236328
Time Complexity for Training: 168.9394087791443
+-----+
| weather      | phi(weather) |
+-----+
| weather(drizzle) | 0.2000 |
+-----+
| weather(fog)    | 0.2000 |
+-----+
| weather(rain)   | 0.2000 |
+-----+
| weather(snow)   | 0.2000 |
+-----+
| weather(sun)    | 0.2000 |
+-----+
Time Complexity for Prediction: 0.0
  
```

Figure 7: Figure showing the different Time complexity calculated for Bayesian Network Model

```
Run workingbayesian x Bayesian_subset x
Bayesian Network Model Evaluation:
Accuracy: 0.03
Log Loss: 1.6094379124340994
Classification Report:
      precision    recall  f1-score   support

 drizzle      0.03      1.00      0.06         3
    fog       0.00      0.00      0.00         9
    rain       0.00      0.00      0.00        41
    snow       0.00      0.00      0.00         4
    sun        0.00      0.00      0.00        43

 accuracy      0.03      0.03      0.03       100
 macro avg      0.01      0.20      0.01       100
weighted avg      0.00      0.03      0.00       100

Process finished with exit code 0
```

Figure 8: Figure showing the Accuracy, Log Loss and Classification Report for the Bayesian Network Model.

Dataset 2 (all data)

```
Run workingbayesian x Bayesian_subset x
C:\Users\shash\AppData\Local\Programs\Python\Python311\python.exe "C:\AI 550\Scripts\workingbayesian.py"
Time Complexity: 0.0019948482513427734
Time Complexity for Training: 1165.0171601772308
+-----+-----+
| weather | phi(weather) |
+-----+-----+
| weather(drizzle) | 0.2000 |
+-----+-----+
| weather(fog) | 0.2000 |
+-----+-----+
| weather(rain) | 0.2000 |
+-----+-----+
| weather(snow) | 0.2000 |
+-----+-----+
| weather(sun) | 0.2000 |
+-----+-----+
Time Complexity for Prediction: 0.007989645004272461
```

Figure 9: Figure showing the different Time complexity calculated for Bayesian Network Model

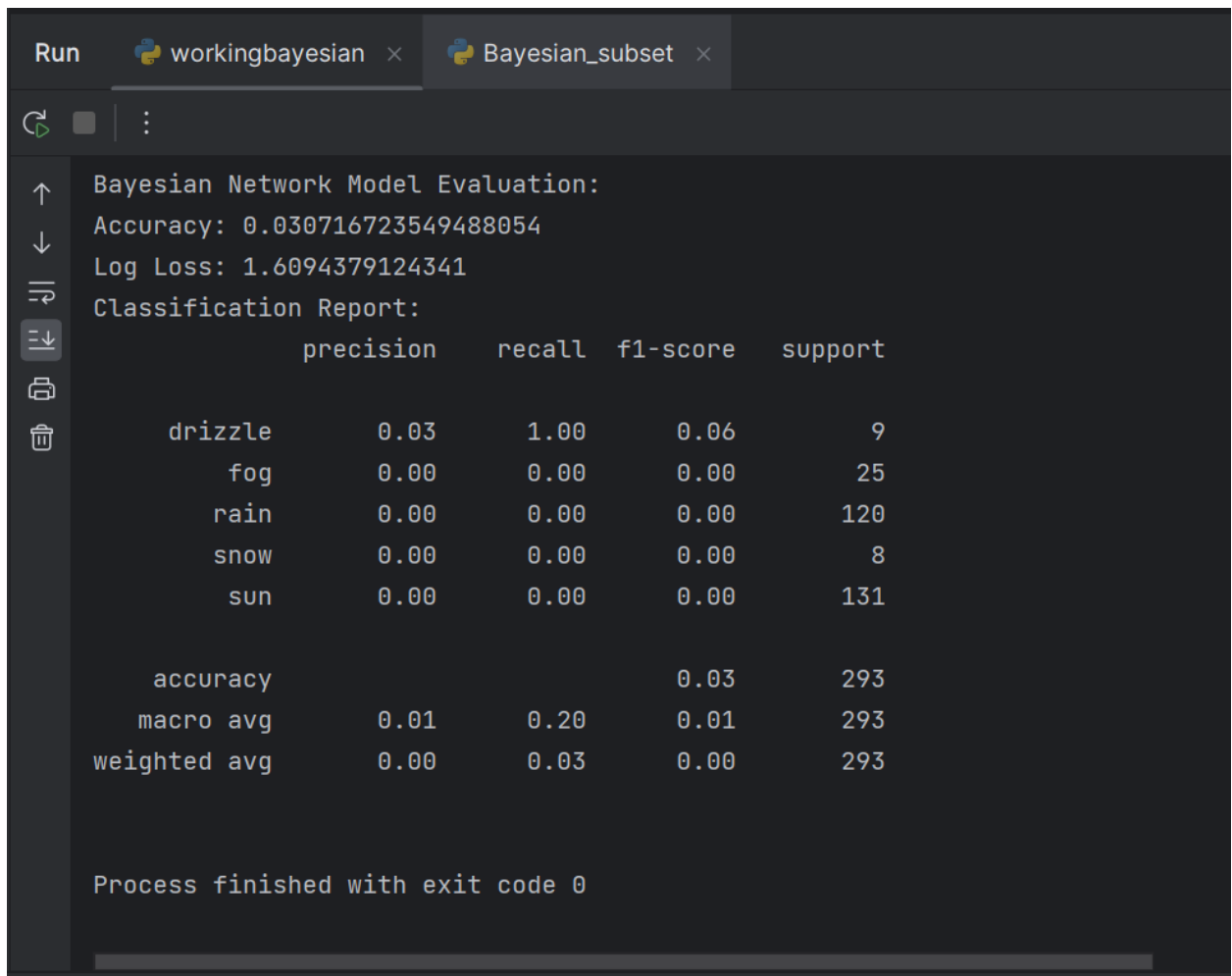


Figure 10: Figure showing the Accuracy, Log Loss and Classification Report for the Bayesian Network Model.

VI. Analysis / Discussion:

1. Comparison of RNN Model 1 (dataset 1) and RNN Model 2 (dataset 2)

- **Execution Time:** Dataset 2 has more execution time (**6.63seconds**) as compare to dataset 1(**3.72 seconds**) for the RNN model, with Dataset 2 having slightly higher execution time due to the larger size of the dataset.
- **Accuracy:** The accuracy of the model is higher for Dataset 2 (**72.35%**) compared to Dataset 1 (**65%**), indicating that the model performs better on Dataset 2.
- **Log Loss:** The log loss is lower for Dataset 2 (**0.85**) compared to Dataset 1 (**1.24**), suggesting that the model's prediction confidence is higher for Dataset 2.
- **Classification Report:** The precision, recall, and F1-score for each class are provided in the classification report for both datasets. The values for these metrics are generally higher for

Dataset 2 compared to Dataset 1, indicating better performance in correctly identifying each class for Dataset 2.

- Overall, the analysis reveals that the RNN model performs better on Dataset 2 compared to Dataset 1 in terms of accuracy, log loss, and classification metrics. Despite the difference in dataset size (500 records for Dataset 1 and larger for Dataset 2), the model's performance is notably improved on the larger dataset.
- These results suggest that the RNN model benefits from a larger dataset, achieving higher accuracy and better predictive performance when trained on more data. This highlights the importance of dataset size in training machine learning models and its impact on model performance.

2. Comparison of Bayesian Model 1 (dataset 1) and Bayesian Model 2 (dataset 2)

- **Time Complexity:** The time complexity for training the Bayesian network model is much higher for Dataset 2 (**1165.02 seconds**) compared to Dataset 1 (**168.94 seconds**), due to the larger size of Dataset 2.
- **Accuracy:** The accuracy of the model is slightly higher for Dataset 2 (0.0307) compared to Dataset 1 (0.03), although the difference is marginal. Both datasets show very low accuracy, suggesting that the model's predictions are not accurate.
- **Log Loss:** The log loss is consistent across both datasets, with a value of approximately 1.609 for both. This indicates that the model's prediction confidence is similar across different dataset sizes.
- **Classification Report:** The precision, recall, and F1-score for each class are provided in the classification report for both datasets. The values for these metrics are consistent between the two datasets, indicating similar performance in terms of correctly identifying each class. However, Dataset 2 has slightly higher precision and recall values for all classes compared to Dataset 1.

3. Comparison of RNN Approach vs Bayesian Network Approach

- **Accuracy:** The RNN model significantly outperforms the Bayesian model in terms of accuracy on both datasets. The accuracy of the RNN model is notably higher, ranging from **65% to 72.35%**, while the accuracy of the Bayesian model is consistently low, around **3%**.
- **Log Loss:** The RNN model achieves lower log loss values compared to the Bayesian model on both datasets. Lower log loss values indicate better predictive performance and higher confidence in the model's predictions. The RNN model achieves log loss values ranging from **0.85 to 1.24**, while the Bayesian model consistently has a log loss of 1.609.
- **Performance across Datasets:** Both models show improvement in performance when applied to Dataset 2 compared to Dataset 1. However, the improvement is more pronounced for the RNN model, which shows a significant increase in accuracy and decrease in log loss on Dataset 2. In contrast, the Bayesian model's performance remains largely unchanged between the two datasets.

- **Overall Assessment:** The RNN model demonstrates superior performance compared to the Bayesian model in terms of accuracy and log loss on both datasets. The RNN model's ability to capture complex patterns and relationships in the data likely contributes to its better performance. Additionally, the RNN model shows more significant improvements when applied to a larger dataset, highlighting its scalability and potential for handling larger and more complex data.

4. **Probable Causes of Poor Performance of Bayesian Network Model:**

The performance of Bayesian Network Model could be a result of the following factors:

- Model Complexity
- Insufficient Data
- Incorrect Model Assumptions
- Inadequate Training
- Incorrect Model Structure
- Overfitting or Underfitting
- Algorithmic Limitations

VII. References

- **Data Source: Weather Prediction Dataset**
<https://www.kaggle.com/datasets/ananthr1/weather-prediction/data>
- **Introduction to RNN and LSTM**
<https://www.theaidream.com/post/introduction-to-rnn-and-lstm>
- **pgmpy- Bayesian Network**
<https://pgmpy.org/models/bayesiannetwork.html>
- **pgmpy – Maximum Likelihood Estimator**
https://pgmpy.org/param_estimator/mle.html
- **pgmpy- Variable Elimination**
https://pgmpy.org/exact_infer/ve.html
- **Keras: The high-level API for TensorFlow**
<https://www.tensorflow.org/guide/keras>
- **The Sequential Model**
https://www.tensorflow.org/guide/keras/sequential_model