**Predicting Precipitation with Machine Learning**

**Yousif Husein**

1. **Introduction**

**1.1 Problem Statement**

Predicting precipitation is a critical task when dealing with weather forecasting since it impacts agriculture, transportation, and public safety. Being able to determine if it will rain accurately and the amount of rain can help people and organizations make decisions beforehand. My project seeks to address this problem using machine learning models to predict precipitation based on weather data from the London Weather dataset.

**1.2 Motivation and Challenges**

The motivation for my project is to help organizations make more accurate and better-informed decisions using machine learning. Machine learning can help better traditional weather forecasting models with a better approach for complex and nonlinear relationships between weather variables. But challenges occur with handling noisy datasets and achieving high prediction accuracy with limited data.

**1.3 Concise Summary of Solution**

In the study I used a dataset with roughly 15,000 datasets for weather in London including things such as cloud cover, sunshine, max temperature, mean temperature, precipitation. The project uses Gaussian Naïve Bayes for classification (rain vs no rain) and a Neural network for regression (predicting precipitation amounts). The dataset was divided into a 60-20-20 split for training, validation, and testing.

1. **Data/Environment**

**2.1 Detailed Introduction**

I used the London Weather Dataset which is a collection of meteorological data that has been recorded over a significant period. The dataset includes multiple features such as:

* Cloud Cover: Measurement of the sky’s cloud coverage
* Sunshine: Total sunshine hours during the day
* Global Radiation: Amount of solar radiation hitting the Earth’s surface
* Maximum Temperature: Highest temperature reached in a day
* Mean Temperature: Average daily temperature
* Precipitation: The amount of rainfall

**2.2 Data Visualization**

A correlation heatmap was made to better understand the data so we can examine the relationship between the features and the target variable which is precipitation. A scatterplot of cloud cover vs precipitation was generated, which visually demonstrates that higher cloud cover values are associated with rainy days.

**2.3 Data Analysis**

Most features like sunshine and cloud cover show varied distributions with cloud cover showing a very strong relationship to precipitation. Rows that contained missing values were removed so there would be consistency through the training process. The target was split into two classes such as 0 and 1 for No Rain and Rain so it would work with Gaussian Naive Bayes.

**2.4 Data Preprocessing**

The data was split into training (60%), validation (20%), and testing (20%) sets. Standard scaler was applied to standardize features which makes sure they have a mean of 0 and a standard deviation of 1.

1. **Method**

**3.1 Gaussian Naive Bayes**

In my project I used Gaussian Naive Bayes to classify precipitation into two categories: Rain and No Rain. The algorithm then calculates the probability of each class with the given features. It uses log priors, feature means and standard deviations, log likelihoods, and prediction. I chose Gaussian Naive Bayes for its simplicity and its ability to handle smaller datasets such as this one.

**3.2 Neural Network**

I used a two-layer feed forward neural network that was designed for regression tasks that is adapted to binary classification. It includes an input layer, hidden layer, and an output layer. The input layer includes five neurons for the five features, the hidden layer uses the Tanh activation function to map input values to the range [-1, 1] which enables it to capture nonlinear relationships. The output layer is 1 neuron with a Sigmoid activation function for the binary classification. The network then was trained using min-batch stochastic gradient descent with a learning rate of .01 and a batch size of 32. Mean squared error was used at first for the loss function but then for classification I employed binary cross-entropy. Finally, the model’s performance was evaluated using accuracy on validation and test sets and learning curves were plotted to monitor the training loss over epochs to test for overfitting.

1. **Results**

**4.1 Explanation**

The experiments were done using the London Weather dataset which was then preprocessed to handle missing values and to transform the target variable for classification and regression. The two models that were tested were:

1. Gaussian Naive Bayes
   1. Predicts whether it will rain with binary classification
   2. Target variable is set to either 0 or 1 for No Rain or Rain
   3. Performance is evaluated using accuracy and a confusion matrix
2. Neural Network
   1. Predicts the amount of precipitation with regression
   2. Target variable is normalized to a scale between 0 and 1
   3. Performance is evaluated using Root Mean Squared Error on the training, validation, and test sets.

Each model was trained with 60% of the data, validated with 20% and tested with the last 20%.

**4.2 Test Results**

Gaussian Naive Bayes

* Testing Accuracy: 65.04%
* Confusion Matrix: This demonstrates the performance with true positives, true negatives, false positives, and false negatives

A graph with blue squares

Description automatically generated

Observations:

* This model had alright accuracy but struggled a lot with false positives
* This result shows the simplicity of this learning model which assumes feature independence.

Neural Network:

* Testing RMSE: 0.0614
* Learning Curve: The training and validation RMSE decreased over the 500 epochs which indicates effective learning

A graph of a graph

Description automatically generated

Observation:

* This learning model performed well in predicting precipitation levels
* The small gap between the training and validation suggests that there is good generalization

**4.3 Analysis of results**

Gaussian Naive Bayes:

* Strengths: Easy and quick to train and easy to read.
* Weaknesses: Struggles with overlapping features and nonlinear relationships

Neural Network:

* Strengths: Captures nonlinear relationships efficiently and effectively
* Weaknesses: Requires more computational resources and more careful/precise tuning for hyperparameters

The trade off between the simplicity of Gaussian Naive Bayes and the complexity of Neural Networks shows the difference between handling different types of data such as linear vs nonlinear.

**4.4 Experiments to Support Analysis**

1. Feature Importance:

* Cloud cover shows a strong correlation with precipitation and makes it a key feature
* Visualizations like scatterplots help highlight this relationship and many others

2. Decision Boundary:

* Gaussian Naive Bayse decision boundary shows how it’s limited when it comes to capturing nonlinear separations

A chart of a decision boundary

Description automatically generated with medium confidence

3. Learning Curve for Generalization:

* The learning curve for Neural Network shows there is successful training without much overfitting

1. **Conclusion**

**5.1 Concluding Remarks**

In my projects, I explored two machine learning models (Gaussian Naive Bayes and Neural Network) to predict rainfall using a London Weather dataset. This process involved data preprocessing, model training, and evaluation with proper visualizations. It also helped highlight the importance of choosing the right models for the right tasks or else you will struggle to get the results you are aiming for.

**5.2 What I Learned**

I learned the application of machine learning on real world problems and when to use what model over other models. The main thing I learned is the effectiveness of Gaussian Naive Bayes and Neural Networks.

* Gaussian Naive Bayes: Shows the simplicity and efficiency of smaller models especially for classification tasks. But it struggles with nonlinear relationships in data
* Neural Networks: Shows the power of complex models and how it captures nonlinear patterns and how important it is to tune hyperparameters properly to ensure proper training and generalization

I also learned useful techniques for feature scaling, target transformation, and handling missing values. I gained experience with tuning hyperparameters to meet my needs such as learning rate and batch size so my models can be optimized without underfitting or overfitting. I also realize the importance of splitting the data into training, validation, and testing since it helped me optimize my model performance and generalization.

1. **References**

“London Weather Data.” www.kaggle.com, [www.kaggle.com/datasets/emmanuelfwerr/london-weather-data](http://www.kaggle.com/datasets/emmanuelfwerr/london-weather-data).

1. **Acknowledgement**
2. ChatGPT by OpenAI:
   1. Throughout the project I independently implemented machine learning models from previous homework assignments to predict rainfall using the London Weather dataset. During this process I used ChatGPT to help refine my knowledge and used it as a tool to assist with code adaptation, more specifically to align the structure of my old homework to the new dataset. I used ChatGPT to clarify concepts and fix any issues I got stuck on and couldn’t figure out.
3. **Source Code**

<https://github.com/YousifHusein/AILearningModel>