THE GEORGE
WASHINGTON
UNIVERSITY
WASHINGTON, DC

Machine Learning I DATS 6202

(MS In Data Science)

Group Report

Group 3

Rainfall Prediction

Instructor: Amir Jafari

Authors:

Sudhanshu Deshpande

Date: 05/01/2023

TABLE OF CONTENTS

1: Introduction

Rainfall prediction is a critical application of machine learning that has a wide range of practical applications, from agriculture to transportation to disaster management. In this project, our objective is to use the dataset to predict whether it will rain on the next day based on various weather observations made on the current day.

We will first investigate the dataset using exploratory data analysis (EDA). We will pre-process the data after examining it to manage missing values, encode category variables, and scale numerical characteristics. Then, using different algorithms such as logistic regression, MLP Classifier and random forest, we will train and evaluate the machine learning models. Finally, we will select the best performing model and fine-tune its hyperparameters using cross-validation. The result will be a machine learning model that can accurately predict whether it will rain on a given day based on the weather observations. This project has practical applications in weather forecasting and risk assessment and can help inform decision-making in various industries.

2: Dataset Description

The weather dataset contains daily weather observations from various weather stations across Australia, spanning from 2007 to 2017. The dataset includes 142,193 instances and 24 features, including temperature, humidity, rainfall, wind speed, and direction, among others. The target variable is *'RainTomorrow'*, which indicates whether it rained on the following day. The data is in a structured format, with mostly numerical and categorical features. The dataset has missing values, which will require data pre-processing before modelling. The weather dataset is a suitable candidate for binary classification tasks related to rain prediction and risk assessment.

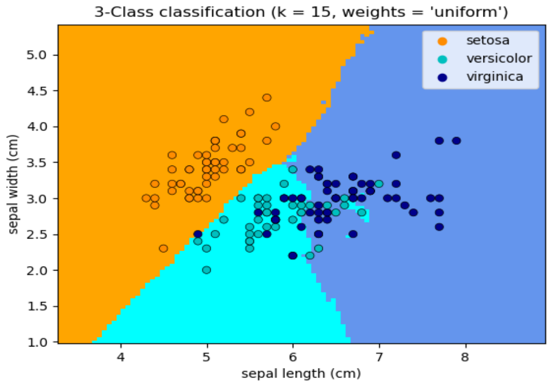
3: Machine Learning Algorithms

Machine learning algorithms use statistical models and algorithms to identify patterns in data and make predictions or decisions based on that data. As we have a binary classification problem, we have used the following methods.

3.1. KNN (K-Nearest Neighbors) Classifier

KNN Classifier is a non-parametric machine learning algorithm used for classification tasks. It is based on the idea that similar data points tend to belong to the same class. The algorithm determines the class of a new data point by finding the k nearest neighbors in the training data and assigning the most common class among them as the predicted class for the new data point. The algorithm is simple and easy to implement but can be computationally expensive for large datasets. It also requires careful selection of the hyperparameter k, which determines the number of neighbors to consider.

In summary, K Neighbors Classifier is a simple and effective non-parametric algorithm for classification tasks. It works by finding the k nearest neighbors in the training data and assigning the most common class among them as the predicted class for a new data point.



4. Experimental Setup

Our goal is to develop a machine learning model that can accurately predict whether it will rain on a given day based on the weather observations. To achieve this, we will first explore the dataset using various exploratory data analysis (EDA) techniques, such as data cleaning, univariate analysis, and bivariate analysis.

After exploring the dataset, we will pre-process the data to handle missing values, encode categorical variables, and scale numerical features. Next, we will train and evaluate machine learning models using algorithms K Neighbours classifier.

- Data Preparation:
  - Load the dataset and split it into training and testing sets.
  - Identify the missing values in the dataset.

- Imputation of Numerical Variables:
  - For numerical variables, impute the missing values using the mean of the non-missing values in that column.

- Imputation of Categorical Variables:
  - For categorical variables, impute the missing values using the mode of the non-missing values in that column.
  - Convert categorical variables into binary variables using one-hot encoding.

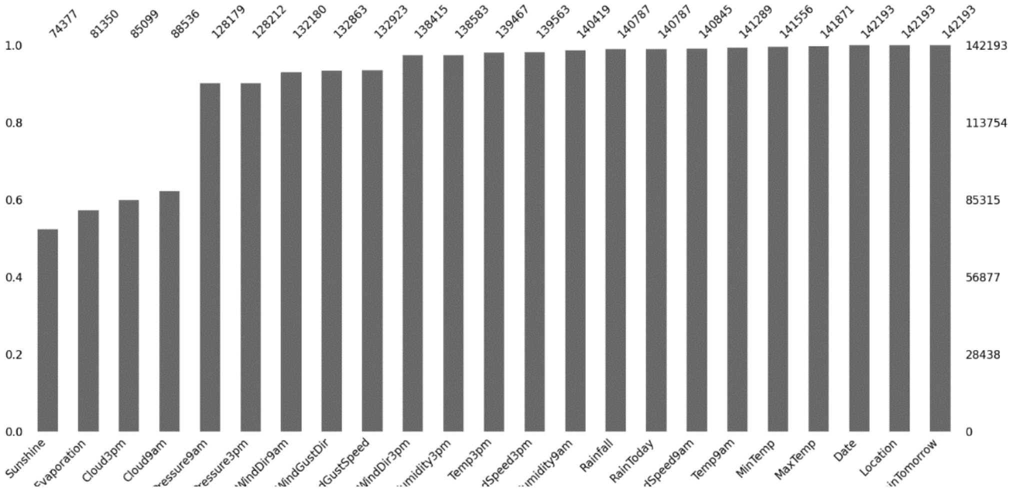
- Feature Scaling:
  - Standardize the numerical features by scaling them to have zero mean and unit variance using standard scaling.

- Feature Selection:
    - Use the SelectKBest algorithm to select the top K features based on their importance scores.

- KNN Model Training and Evaluation:
    - Train a KNN classifier on the selected features using the training set.
    - Evaluate the performance of the model using the testing set.
    - Repeat the experiment with different values of K and select the optimal value based on the model's performance metrics.

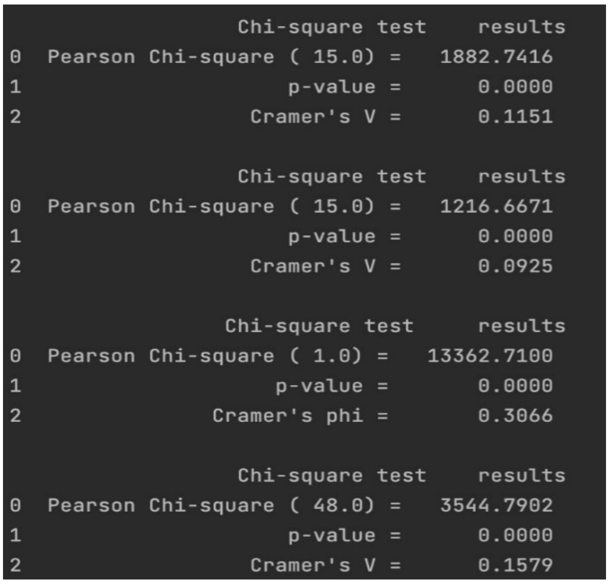
5. Results

Data Imputation



For imputing data I have replaced missing values in numerical variable with the mean and missing value of categorical variable with mode.

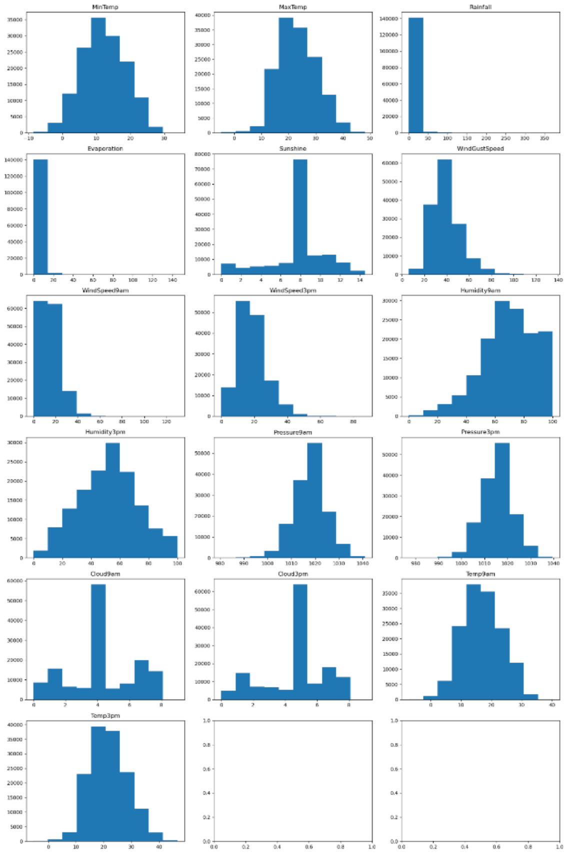
Feature Importance

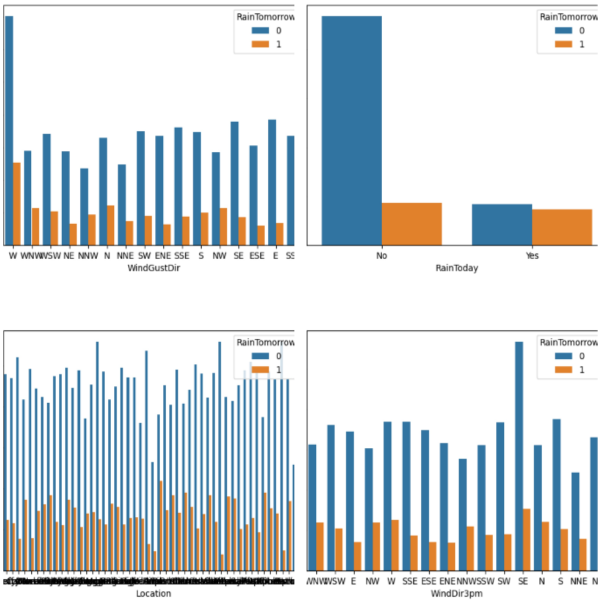


```
                    Chi-square test     results
0  Pearson Chi-square ( 15.0) =    1882.7416
1                   p-value =       0.0000
2                Cramer's V =       0.1151

                    Chi-square test     results
0  Pearson Chi-square ( 15.0) =    1216.6671
1                   p-value =       0.0000
2                Cramer's V =       0.0925

                   Chi-square test      results
0  Pearson Chi-square ( 1.0) =    13362.7100
1                  p-value =        0.0000
2             Cramer's phi =        0.3066

                    Chi-square test     results
0  Pearson Chi-square ( 48.0) =    3544.7902
1                   p-value =       0.0000
2                Cramer's V =       0.1579
```

For feature importance we have used Chi-square test for categorical variables and SelectKBest for the numerical data.

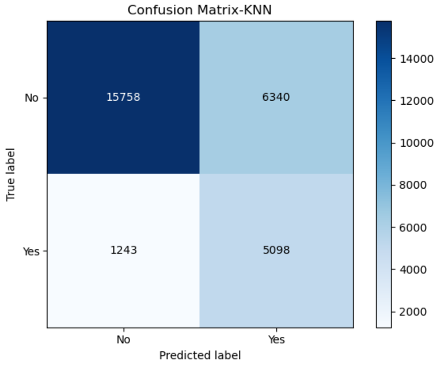
- ## Data Distribution (EDA)





Here we can see that the data is fairly distributed.

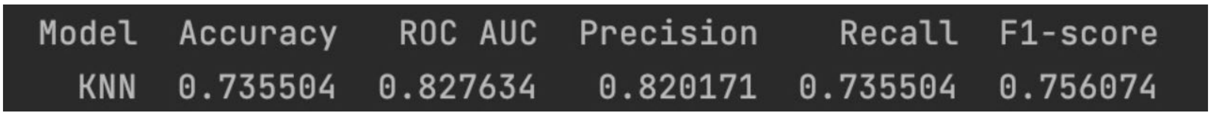
- ## KNN



- The total number of instances in the test set is 26,439 (15758 + 6340 + 1243 + 5098).
- Out of these instances, 15,758 were actually negative and were correctly classified as negative (true negatives).
- 6,340 instances were actually positive and were incorrectly classified as negative (false negatives).
- 1,243 instances were actually negative and were incorrectly classified as positive (false positives).

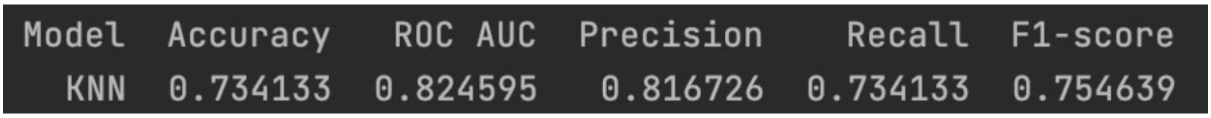
- 5,098 instances were actually positive and were correctly classified as positive (true positives).

- Before Performing PCA



```
Model  Accuracy   ROC AUC  Precision    Recall  F1-score
  KNN  0.735504  0.827634   0.820171  0.735504  0.756074
```

- After Performing PCA



```
Model  Accuracy   ROC AUC  Precision    Recall  F1-score
  KNN  0.734133  0.824595   0.816726  0.734133  0.754639
```

6. Summary and Conclusions

As we can see from the model which we build, we got the accuracy for the model before performing PCA and after performing the PCA is almost same, there is not much change in the accuracy and the F1-score of the model.

But when we utilized the K-Nearest Neighbors Classifier algorithm to build a classification model. The model was trained on the training data using 10-fold cross-validation, where k was set to 3. The accuracy of the model was evaluated using the cross_val_score() function in the scikit-learn library.

The results of the cross-validation indicate that the model had an average accuracy of 0.865 with a standard deviation of 0.007, which suggests that the model's performance is consistent across the different folds of the cross-validation process. These results demonstrate that the K-Nearest Neighbors Classifier algorithm is a suitable choice for this classification problem.

- Percentage of work from internet
  I have taken around 80 lines of code from internet modified around 35 lines of code and written 30 lines of code.
  Percentage of work: 40.90%

7. References

Machine Learning Mastery article on K Nearest Neighbors: https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

Towards Data Science article on K Nearest Neighbors: https://towardsdatascience.com/knn-for-classification-with-scikit-learn-python-7cbef5d914eb

Kaggle tutorial on K Nearest Neighbors: https://www.kaggle.com/learn/intro-to-machine-learning

Appendix

- Computer used:
  - MSI Gaming
- Software used:
  - Pycharm Professional
  - GitHub
  - Microsoft Word
  - Microsoft Powerpoint