Finding the Best Binary Classification Model to Predict Rain

Rio Pramana
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
rio.pramana@binus.ac.id

Debora
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
debora002@binus.ac.id

Enrico Fernandez
Computer Science Department
School of Computer Science
Bina Nusantara University
Jakarta, Indonesia
enrico.fernandez@binus.ac.id

***Abstract*—Being able to predict whether it will rain tomorrow or not is important, especially for the agricultural sector. With this ability, people can prevent damage such as crop loss and property damage caused by the rain. But predicting rain is not trivial, especially if done manually. This calls for the help of machines to perform this task. We have tried to achieve this by building different models that can perform binary classification to predict whether it will rain tomorrow or not. From these different models, we found that the Random Forest model is the best model to use, even by only using 18 features to predict rain, reaching an accuracy score of 85,81%.**

Keywords—Rain Prediction, Random Forest Classifier, Machine Learning, Classification, Binary Classification

I. Introduction

The rainfall prediction plays an important role especially for the field in the agricultural sector. An excessive and irregular rainfall can have a variety of repercussions, such as crop loss and property damage. Therefore, a model that can forecast the rain is required as an early warning that can limit the risk of agricultural property damage. The prediction result can help farmers and water resources to be utilized efficiently. Predicting rainfall is challenging and accurate results must be taken into account. Traditional methods to predict rainfall cannot work in an efficient way, as machine learning techniques can produce more accurate results. Different techniques produce different accuracies, so it is important to choose the right algorithm and model it according to the requirements.

This paper is organized as follows: the next section focuses on the literature review. The methodology utilized is given in section III. Section IV constitutes the results and discussions, and finally, the study's conclusions are given in section V.

II. Literature Review

A. Binary Classification

Binary classification is a task of classifying (the process of predicting categorical variables) elements of a set into only two classes. Binary classification has several applications such as seen in Fig. 1 below [1].

Application	0	1
Medical Diagnosis	Healthy	Diseased
Email Analysis	Not Spam	Spam
Financial Data Analysis	Not Fraud	Fraud
Marketing	Won't Buy	Will Buy

Fig. 1. Binary classification application examples

There are a few commonly used methods for binary classification, they are:

1. Decision trees
2. Random forests
3. Bayesian networks
4. Support vector machines
5. Neural networks
6. Logistic regression
7. Probit model

B. Random Forest

Random forests is one of the ensemble learning methods that is often used for classification tasks. For classification tasks, the output will be the class selected by most trees [2]. It is believed that random forests usually outperform decision trees.

Random forest is believed to be an easy-to-use machine learning algorithm but still produces great results even without hyperparameter tuning [3]. Random forests are available to use in Python by using the sklearn.ensemble package [4].

C. Feature Importance

Feature importance generally refers to techniques used to calculate the importance score for every input feature for a given model. The higher the importance score, the larger the effect of that particular input feature on the performance of the model. Feature importance is useful for feature selection [5].

D. Hyperparameter Tuning

Hyperparameters are parameters whose value is used to control the learning process of an algorithm. Hyperparameter tuning (sometimes called hyperparameter optimization) in machine learning means the process of selecting a set of optimal hyperparameters for a learning algorithm to produce the optimal result [6].

Generally we can not immediately identify the optimal hyperparameters for a given model. Therefore, we would

ideally like to explore a range, if not all of the possible hyperparameter values and combinations to find the optimal set of hyperparameters. But in machine learning we will ideally ask the machine to perform this task and that is why we do hyperparameter tuning.

III. Methodology

A. Data Collection

The data set used in this paper is obtained from the kaggle.com platform at the following address (**https://www.kaggle.com/jsphyg/weather-dataset-rattle-package#weatherAUS.csv** accessed on _ June 2022). The dataset consists of a sample of 145460 data with information on 23 study variables as shown in Table I. The data describe meteorological information from different cities in Australia collected daily for 10 years. In particular, there is a Boolean variable that indicates whether it rains on the next day, RainTomorrow, which will be the target variable that will be tested to predict using several machine learning algorithms.

TABLE I. Dataset Descriptions

Rainfall Parameters	Units	Description
Date	yyyy-mm-dd	Day on which the measurement is carried out
Location		
Minimum Temperature	Degree Celsius	The minimum temperature in Degree Celsius
Maximum Temperature	Degree Celsius	The maximum temperature in Degree Celsius
Rainfall	Millimeters (mm)	Rainfall amount recorded for the day
Evaporation	Millimeters (mm)	Rainfall amount recorded for the day
Sunshine	Hours	Number of hours for the sun radiant during the day
WindGustDir		The direction of the wind gust in 24h
WindGustSpeed	km/h	Speed of the wind gust in 24h
WindDir9am		The direction of the wind at 9 a.m.
WindDir3pm		The direction of the wind at 3 p.m.
WindSpeed9am	km/h	The wind speed in the 10 min before 9 a.m.
WindSpeed3pm	km/h	The wind speed in the 10 min before 3 p.m.
Humidity9am	Percentage (%)	Humidity at 9am
Humidity3pm	Percentage (%)	Humidity at 3pm
Pressure9am	hpa	Atmospheric pressure at 9 a.m.
Pressure3pm	hpa	Atmospheric pressure at 3 p.m.

B. Preprocessing Data and Exploratory Data Analysis

To carry out the model, a certain set of operations was carried out to prepare the data set.

1. Handling missing data (target variable)

   After we checked the initial dataset, we found that the target variable (RainTomorrow) has some missing data. From 145460 rows of data, there are 3267 rows of data where the target variable has a missing value. Because 3267 is only a small percentage of the whole dataset (less than 0.03%), we decided that it is best to drop those rows of data with little to no impact to the whole dataset rather than having to impute them and potentially introducing bias to the data.

   We have also found small to large amounts of missing data in almost every independent variable of the dataset. But we decided to handle them after we had done some other preprocessing steps.

2. Feature Engineering

   When checking the cardinality of each categorical variable, we found 1 categorical variable that has a very high level of cardinality, which is the 'Date' variable. 'Date' variable contains 3436 labels compared to other categorical variables that only contain less than 50 labels each (Fig. 2).

```
Date contains 3436 labels
Location contains 49 labels
WindGustDir contains 16 labels
WindDir9am contains 16 labels
WindDir3pm contains 16 labels
RainToday contains 2 labels
RainTomorrow contains 2 labels
```

Fig. 2. Cardinality of categorical variables

This could result in having 3436 additional columns/features when we encode the 'Date' variable which is a huge problem. To solve this, we decided to separate the 'Date' variable to 3 variables/features. They are 'Year', 'Month', and 'Day'.

3. Drop unnecessary features

The next step that we decided to take is removing any duplicate values and unnecessary features, if they exist in the dataset. After checking for duplicate values, we found none. Therefore, we continued to identify unnecessary features. We did a multivariate analysis by plotting a correlation heatmap of every feature to identify any unnecessary features.

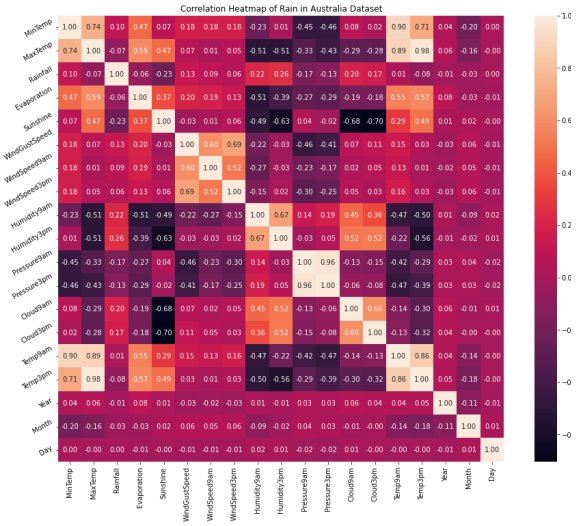

Fig. 3. Correlation heatmap visualization

From the correlation heatmap seen in Fig. 3, we noticed a few features that have a very strong and positive correlation (Greater than or equal to 0.9). They are:

a) MinTemp & Temp9am (0.9)
b) MaxTemp & Temp3pm (0.98)
c) Pressure9am & Pressure3pm (0.96)

These three correlations mean that they give very similar, if not identical, effects/contributions to the prediction of the model. Therefore, one of the variables in each strongly positive correlation ($\geq$ 0.9) is deemed unnecessary. We decided to remove variable 'Temp9am', 'Temp3pm', 'Pressure9am', and keep the rest.

4. Splitting training and test dataset

Before we do anything else such as encoding and feature scaling, we have to split the dataset first to avoid mistakes such as overfitting. We decided to split the data into 80% training dataset and 20% test dataset.

5. Handling missing data (features)

Assuming the missing data are categorized as Missing Completely At Random (MCAR), we decided to handle the missing data by imputing the missing values. There are a few features missing around 50000 data (around 34% of the dataset) so we can not drop them as it will affect the dataset tremendously.

For numerical data, we chose to do median imputation because it is robust to outliers (we have not dealt with outliers yet at this stage). To avoid overfitting, we did imputation over the training set, and then propagated them to the test set.

For categorical data, we chose to do mode imputation by replacing missing data with the most frequent values of the respective column. We did the same thing as in the handling of numerical data to avoid overfitting.

6. Engineering outliers data

When we looked at the data, we suspected 4 variables have outliers, they are 'Rainfall', 'Evaporation', 'WindSpeed9am', 'WindSpeed3pm'. To confirm our suspicion, we drew the boxplots to visualize the outliers as seen in Fig. 4.

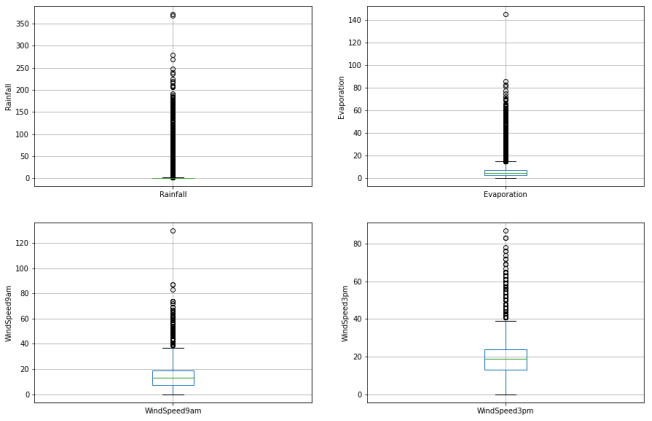

Fig. 4. Boxplots for visualizing outliers

Based on the boxplots, our suspicion was confirmed. We decided to handle these outliers by using the top-coding approach to cap the maximum values and remove outliers above this cap. The cap for 'Rainfall' is 2.0, 'Evaporation' is 14.6, 'WindSpeed9am' is 37.0, and 'WindSpeed3pm' is 40.5.

7. Encoding categorical data

To encode categorical data, we used Binary Encoder and pd.get_dummies(). After the encoding was done, we ended up with 115 features for the model to use.

8. Feature Scaling

We used MinMaxScaler to scale the features on our training and test dataset.

C. *Modelling*

The objective of the project is to analyze the possibilities of predictions using machine learning algorithms as a tool for predicting rain as another alternative from traditional almanac rain predictions. In this project, the following algorithms were applied: logistic regression, decision tree and random forest. The algorithms are described below.

1. Logistic Regression

Logistic Regression is a statistic-based algorithm that is used in classification problems. It allows predicting the probability of an input that belongs to a certain category. Sklearn library has a module that can call logistic regression, which is linear_model. SWe used {solver = 'liblinear', random_state = 0} as hyperparameters for this model.

```
Confusion matrix

 [[20882  1185]
 [ 3084  3288]]

True Positives(TP) =  20882

True Negatives(TN) =  3288

False Positives(FP) =  1185

False Negatives(FN) =  3084
ROC AUC RF: 0.8708
              precision    recall  f1-score   support

          No       0.87      0.95      0.91     22067
         Yes       0.74      0.52      0.61      6372

    accuracy                           0.85     28439
   macro avg       0.80      0.73      0.76     28439
weighted avg       0.84      0.85      0.84     28439
```

Fig. 5. Logistic Regression Confusion Matrix

2. Decision Tree

Decision tree helps in predicting the target parameter based on several input parameters. This tree can be made by splitting the data set into a subset based on the attribute value. This algorithm was implemented using the DeicisionTreeClassifier function.

```
Confusion matrix

 [[20150  1917]
 [ 3411  2961]]

True Positives(TP) =  20150

True Negatives(TN) =  2961

False Positives(FP) =  1917

False Negatives(FN) =  3411
ROC AUC RF: 0.7311
              precision    recall  f1-score   support

          No       0.86      0.91      0.88     22067
         Yes       0.61      0.46      0.53      6372

    accuracy                           0.81     28439
   macro avg       0.73      0.69      0.70     28439
weighted avg       0.80      0.81      0.80     28439
```

Fig. 6. Decision Tree Confusion Matrix

3. Random Forest

Random Forest is a mix of tree predictors, each tree dependent on the values of the random vector. This algorithm was implemented using the RandomForestClassifier function.

```
Confusion matrix

 [[21235   832]
 [ 3180  3192]]

True Positives(TP) =  21235

True Negatives(TN) =  3192

False Positives(FP) =  832

False Negatives(FN) =  3180
ROC AUC RF: 0.8888
              precision    recall  f1-score   support

          No       0.87      0.96      0.91     22067
         Yes       0.79      0.50      0.61      6372

    accuracy                           0.86     28439
   macro avg       0.83      0.73      0.76     28439
weighted avg       0.85      0.86      0.85     28439
```

Fig. 7. Random Forest Confusion Matrix

D. Evaluation metrics

To evaluate the efficiency of various algorithms being used, it can be done by using evaluation metrics. In this paper, we focus on the accuracy, precision, recall, f-measure and confusion matrix. These can be defined as follows:

Confusion Matrix

The confusion matrix is a performance measurement for machine learning classification and useful for measuring Recall, Precision, Specificity and Accuracy [7] as shown in Table II.

TABLE II. CONFUSION MATRIX

		Actual	
		Positive(P)	Negative(N)
Prediction	Positive(P)	True Positive (TP)	False Positive (FP)
	Negative(N)	False Negative (FN)	True Negative (TN)

It provides different combinations of predicted and actual values, Where:

1. TP is the total number of positive values that are equal to the predicted positive.
2. FP is the total number of negatively classified data that is positive falsely.
3. FN is the total number of positive values that have been classified as negative falsely.
4. TN is the total number of negatively values that is correctly classified.

Other evaluation metrics such as recall, precision, accuracy and F1 score can be deduced in a mathematical expression respectively in equations 1, 2, 3 and 4.

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (1)$$

$$Precision = \frac{true\ positive}{true\ positive + false\ negative} \quad (2)$$

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ input\ samples} \quad (3)$$

$$F1\ Score = \frac{2\ x\ recall\ x\ precision}{recall + precision} \quad (4)$$

We then compared the results of each model with each other. We also calculated the null accuracy, which is the accuracy that could be achieved by always predicting the most frequent class. We can not say that our models performed well based on the accuracy if their accuracy is almost identical as the null accuracy. The null accuracy is calculated by dividing the number of data of the most frequent class and the total number of data in the dataset.

We also use ROC AUC score as another evaluation metric. ROC AUC stands for Receiver Operating Characteristic - Area Under Curve. It is a technique to compare classifier performance. In this technique, we measure the area under the curve (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5. Therefore, ROC AUC is the percentage of the ROC plot that is underneath the curve.

E. Hyperparameters Tuning

After we compared the results of each model, we then choose the best performing model as our main model for the application. But before we deployed the model, we did a hyperparameters tuning in an attempt to improve the model using Halved GridSearch Cross-Validation.

IV. Results & Discussion

The three (3) models namely: Logistic Regression (LR), Decision Tree (DT) and Random Forest (RF) were tested for accuracy. The performance of the models are presented comprehensively in Table III.

TABLE III. Model Accuracy Comparison

Model	Model Accuracy	Null accuracy
Logistic Regression	0.8499	0.7759
Decision tree	0.8127	0.7759
Random Forest	0.8589	0.7759

From Table III, comparing the model accuracy, random forest shows the highest accuracy compared to the other two. In this project, the Random Forest algorithm is applied as the model by training the test data with training data.

But before we move on to the next step, we are still bothered by the fact that the Random Forest algorithm is trained for 115 features which we think can be reduced by a huge amount. Therefore, we tried to calculate the feature importance of these 115 features to identify which features we can remove without reducing the performance of the model in any significant way.

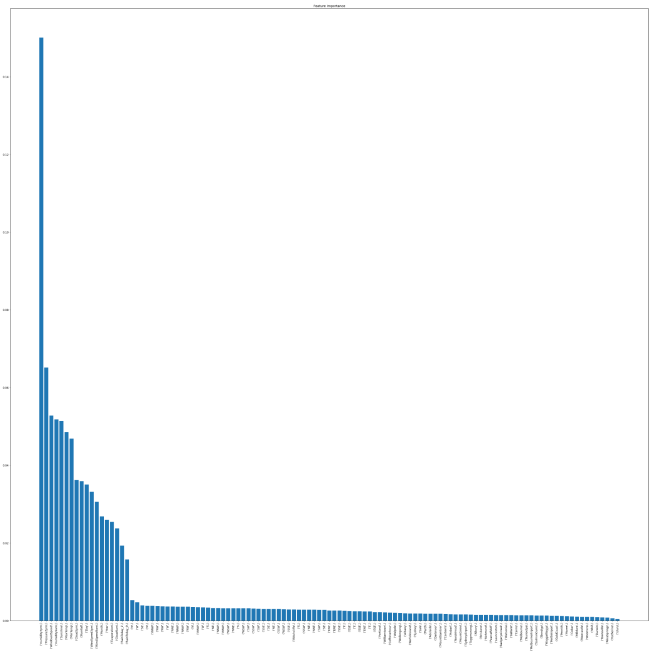

Fig. 8. Feature importance graph

```
('Humidity3pm',)
('Pressure3pm',)
('WindGustSpeed',)
('Humidity9am',)
('Sunshine',)
('MaxTemp',)
('MinTemp',)
('Cloud3pm',)
('Rainfall',)
('Day',)
('WindSpeed3pm',)
('WindSpeed9am',)
('Month',)
('Year',)
('Evaporation',)
('Cloud9am',)
('RainToday_1',)
('RainToday_0',)
('N',)
('W',)
```

Fig. 9. The 20 most important features sorted from the most important one

From Fig. 8 and Fig. 9, we can see that starting from the feature 'RainToday_0', the rest of the features do not contribute enough to the prediction of the model. Only the top 18 features had enough importance to affect the prediction of the model in a significant enough manner.

Therefore, we decided to remove all features except the top 18 ones. Then, we evaluated the model that is trained for these 18 features and compared it to the model that is trained for the initial 115 features. The result is seen in Table IV.

TABLE IV. RANDOM FOREST COMPARISON WITH DIFFERENT FEATURES

Model	Model Accuracy	ROC_AUC
RF_18_feat	0.8581	0.8847
RF_115_feat	0.8589	0.8888

Based on the table above, we can see there is almost no effect by only using the top 18 features compared to using all 115 features. Therefore, we decided to use the model trained for 18 features because it has an almost identical accuracy and ROC AUC score, but it will be easier to take input from the user and the model will run much faster.

The last step is to do hyperparameters tuning. But we can only afford using Halved GridSearchCV instead of the full GridSearchCV because the latter takes too much time to train. Consequently, this led to the Halved GridSearchCV not being able to find the optimal hyperparameters as their hyperparameters suggestion failed to improve our model's accuracy.

```
[ ]  halvedCV_rf.best_params_

     {'criterion': 'entropy',
      'max_depth': 8,
      'max_features': 'auto',
      'n_estimators': 500}

[ ]  halvedCV_rf.best_score_

     0.8535972850146702
```

Fig. 10. Halved GridSearchCV hyperparameters tuning result

We have managed to build a good machine learning model that can predict rain accurately without problems such as overfitting or underfitting so it is a reliable model. Even though this model uses an Australian dataset, we believe that this model can be used to predict rain in any other country just as well given a proper dataset is available.

V. CONCLUSION

In this project, we have made an attempt to build a machine learning model that can predict the rain accurately to reduce the risk of damage caused by rain, especially if people did not expect it.

We used three different models (Logistic regression, Decision trees, and Random forest) to perform binary classification and predict rain. After carefully preprocessing the data and training each model, we found that the random forest model works best as it reaches the accuracy of 0.8589 and ROC AUC score of 0.8888.

But after calculating the feature importance used for this model, we managed to reduce the features from 115 to 18 without affecting the model's performance in a noticeable manner. This new model achieved an accuracy score of 0.8581 and ROC AUC score of 0.8847.

Based on the result of this project, we can conclude that we have successfully built a machine learning model that can predict rain in an accurate way. But the dataset that we used is still limited to Australia. Therefore, we suggest future research and data collection for other countries so we can also utilize this model to predict rain in other countries.

REFERENCES

[1] https://www.learndatasci.com/glossary/binary-classification/

[2] Ho, T. K. (1995, August). Random decision forests. In Proceedings of 3rd international conference on document analysis and recognition (Vol. 1, pp. 278-282). IEEE.

[3] https://builtin.com/data-science/random-forest-algorithm

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285

[6] Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In Automated machine learning (pp. 3-33). Springer, Cham.

[7] https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62]