Data 1030 Final Report: Will It Rain Tomorrow? Predicting Next-Day Rainfall in Australia.

Ruoheng Yuan December 14, 2024
Data Science Institute Brown University
Github Link: https://github.com/DaveYuan23/Data1030_rain_au

Contents

1 Introduction

Rain plays an essential role in our lives, providing water for plants, animals, and humans. It replenishes rivers, lakes, and groundwater, sustaining ecosystems and agriculture. Understanding and predicting rainfall is therefore crucial for planning and decision-making, especially in regions like Australia, where weather patterns can be highly variable and impactful.

In this report, we aim to construct a machine learning pipeline to predict whether it will rain in Australia tomorrow. This is formulated as a binary classification problem, predicting next-day rainfall. The data comes from the Kaggle ”Rain in Australia Competition,” available here. The observations were gathered from a multitude of weather stations managed by the Australian government. Additionally, daily observations can be accessed at this official site.

The balance of this dataset is roughly 7:3, with the ”no” class comprising 70% and the ”yes” class 30%. This dataset contains a significant amount of missing values for continuous variables. Chandrima [1] addressed class imbalance using oversampling for the minority class and handled missing values with the MICE package. She trained seven models, with XGBoost, CatBoost, and Random Forest ranking highest. The best accuracy achieved was 0.95 with XGBoost. Fahad [5] applied mean imputation for missing continuous values and trained five models, with XGBoost again performing best at 0.86 accuracy.

In this report, we experimented with five models: Logistic Regression (with L1, L2 penalties), Random Forest, KNN, and XGBoost. We did not oversample the data for class imbalance, and instead of imputing missing values, we applied a reduced-feature method [6]. Our results closely mirror Fahad’s, with XGBoost as the best performer at 0.87 accuracy.

2 EDA

In the EDA part, we will look deeper into our dataset and introduce the dataset from its shape, whether it has missing values, to visualizations of various features and target variables.

2.1 Data Overview

This dataset contains 10 years of daily weather observations from 49 different Australian weather stations. There are 145,460 rows and 23 columns in this dataset. We have two columns describing the raining conditions in Australia. One is **RainToday**, which records the raining condition on the observation day. The other one is **RainTomorrow**, which is our target variable. Figure 1 Our job is to predict whether it will rain tomorrow.

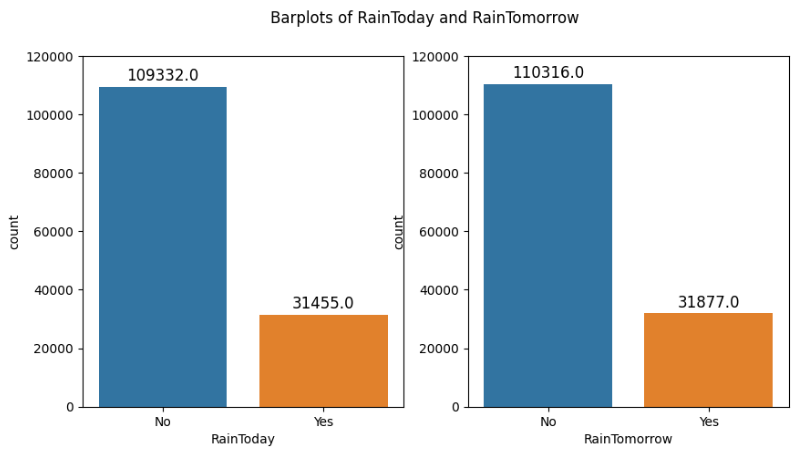


Fig. 1. These bar plots show the class balance of two variables: RainToday and RainTomorrow. The blue bars indicate no rain, and the orange bars indicate rain.

Since our task is to predict the rainfall condition in Australia for the next day (RainTomorrow), and we are using features like RainToday as part of the input, we are leveraging past information to predict future outcomes. In this scenario, the data points are not independent and identically distributed (i.i.d.). To avoid data leakage when training a model with non-i.i.d. data, we will use a *TimeSeriesSplit* approach instead of a basic train-test split.

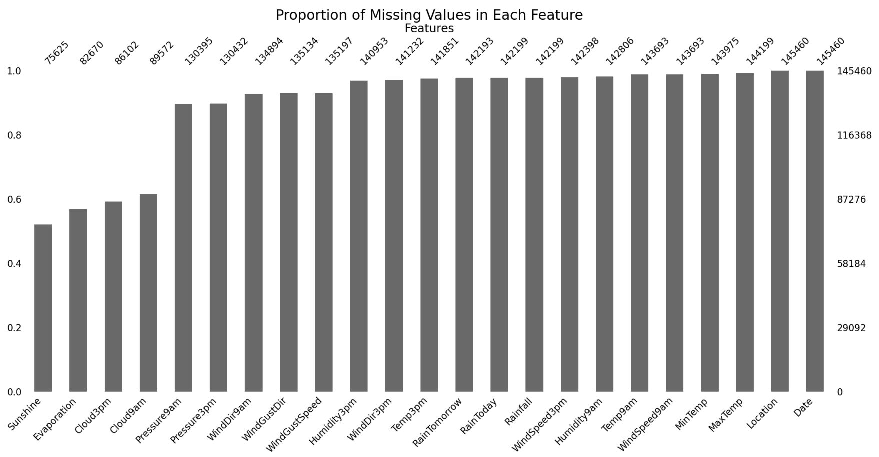


Fig. 2. A bar plot illustrates the proportion of missing values in each feature.

The issue of missing values is highly significant in our dataset, as shown in Figure 2. Among the 23 columns, 21 contain missing values, including the target variable. For categorical variables, 4 out of 6 have missing values, while all continuous variables in the dataset are also affected. We will discuss our approach to handling these missing values in Section 3.2.

2.2 Visualizations

Wind direction is an important factor in determining whether tomorrow will be a rainy day. Exploring which wind direction is associated with the most rainy days is interesting and gives us a sense that if we feel the wind blowing from that direction, we might expect rain soon.

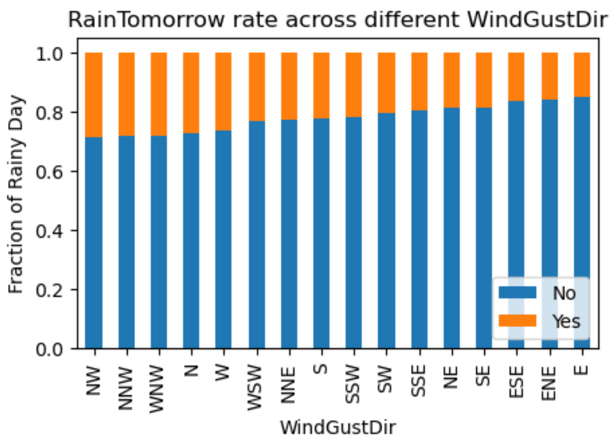


Fig. 3. A stacked barplot illustrating which wind direction is associated with more rainy days.

As we can see from Figure 3, the NW wind direction has the highest proportion of rainy days, followed by NNW and WNW. All of these directions are primarily located in the northwest quadrant. This suggests that if we feel the wind coming from the northwest quadrant, it is likely to rain tomorrow.

Next, we dive into the correlation plot of continuous variables to identify whether there are strong correlations between features.

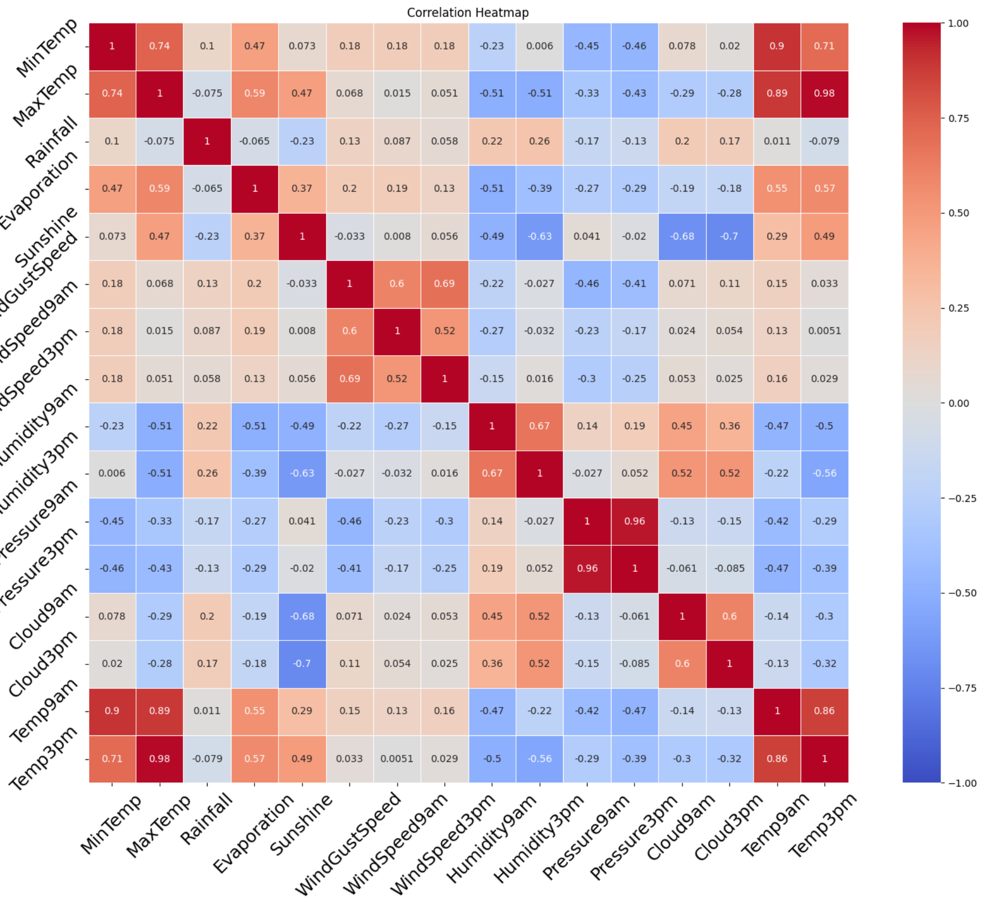


Fig. 4. A correlation plot of continuous variables. Strong correlations are observed among 'Temps' and 'Pressures'.

As shown in Figure 4, the following features exhibit strong correlations:

- **MinTemp**: Highly positively correlated with Temp9am, Temp3pm, and MaxTemp.
- **MaxTemp**: Highly positively correlated with Temp9am and Temp3pm.
- **Pressure9am**: Highly positively correlated with Pressure3pm.

After careful consideration, we decided to drop the columns `Temp9am`, `Temp3pm`, and `Pressure3pm` to reduce redundancy and simplify the model.

3 Methods

In this section, we will introduce our machine learning pipeline, including how we split our dataset, how we handle missing values in continuous variables, and how we tune hyperparameters. Additionally, we will discuss our model selection process and the hyperparameter grid we chose.

3.1 Data Splitting

As we mentioned in Section 2.1, our dataset is not i.i.d., and it can be considered a time series problem, so we cannot simply shuffle our data and split it. Instead, we will use TimeSeriesSplit.

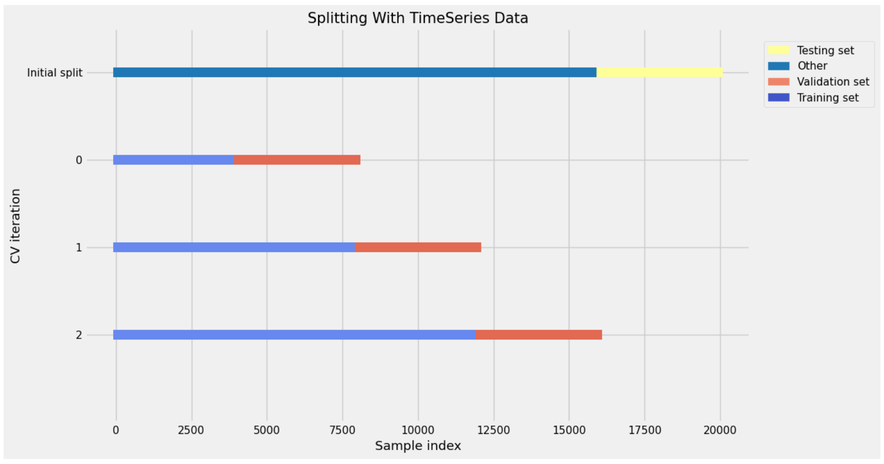


Fig. 5. Splitting techniques used in the ML pipeline.

First, we perform a basic split without shuffling our data (to preserve the time order) to create the test set from the entire dataset. Next, we utilize a time series k-fold method to split the rest of our dataset. As shown in Figure 5, we always use data from the past to train our model and data from the future to test it.

3.2 Preprocessing

Here is the data preprocessing for our problem. We ensure that every categorical variable is properly encoded, and continuous variables are normalized using a standard scaler.

- **Categorical Variables:**
  - We create a new class 'other' to impute all the missing values for the categorical variables.
  - We use a one-hot encoder to encode the categorical variables.
- **Continuous Variables:**
  - We don't impute any missing values in continuous variables.
  - We use a standard scaler to normalize all the continuous variables.
- **After Preprocessing:**
  - We will have 116 features.
  - We will have 142,193 observations.

3.3 ML Pipeline

The Reduced Feature method is a technique for training machine learning models on datasets with continuous missing values. Sometimes, we do not want to impute our continuous variables since it could distort their original distribution. In the Reduced Feature method, a separate model is trained for each missing value pattern using data points that belong to that pattern. When deploying the model, it first identifies which missing value pattern a new data point belongs to, then applies the corresponding model. Here is a pseudocode for our ML pipeline:

```
Input: Set of models $\mathcal{M}$, random states $\mathcal{R}$, unique rows $\mathcal{U}$ (reduced feature method),
       parameter grid $\mathcal{P}$, number of folds $k$
Output: Comparison of test scores for different models
foreach model $m \in \mathcal{M}$ do
    foreach random_state $r \in \mathcal{R}$ do
        foreach unique_rows $u \in \mathcal{U}$ do
            foreach parameter $p \in \mathcal{P}$ do
                foreach fold $f$ in $k$-fold do
                    Compute validation score val_score;
                end
                average_val_score across k-folds;
            end
            Find the best parameter $p^*$ based on average_val_score;
            Obtain test score using the best estimator;
        end
        Repeat the pipeline and compute test scores over all $r$ random states;
    end
    Compare different models using their test scores;
end
```

Algorithm 1: Pipeline with Reduced Feature Method

3.4 Model Selection

Our model selection includes Logistic Regression with L1 and L2 regularization terms, Random Forest, KNN, and XGBoost.ß We did not choose SVC because it is not very efficient, especially for our specific problem, where we need to train a new model for each missing value pattern.

Table 1: ML Algorithms and Parameter Grid

ML Algorithms	**Reduced Feature**	**Regularization Term**	**Tuning Parameters**	**Parameter Grid**
Logistic Regression	Yes	L_1	C	$[0.01, 0.1, 1, 10, 100]$
Logistic Regression	Yes	L_2	C	$[0.01, 0.1, 1, 10, 100]$
Random Forest	Yes	–	$max\_features$ max_depth	$max\_features$: $['log2', 'sqrt']$ $max\_depth : [2, 4, 8, 16, 32]$
KNN	Yes	–	n-neighbors	$[2, 4, 6, 8, 16, 32, 64]$
XGBoost	No	–	α, λ	$\alpha : [0, 0.01, \mathbf{0.1}, 1, 10, 100]$ $\lambda : [\mathbf{0}, 0.01, 0.1, 1, 10, 100]$

4 Results

In this section, we will examine the results of our ML pipeline. We will discuss the test scores of each ML model and how they compare to the baseline. Additionally, we will analyze the global and local feature importance.

4.1 Test Scores

After running the ML pipeline from Section 3.3, we calculated the test scores for all five models. Since overall accuracy is crucial for our problem, we used accuracy as the evaluation metric. As shown in Table 2, the baseline accuracy is 0.761 (calculated by predicting all observations as the negative class, i.e., no rain). XGBoost achieves the highest test accuracy at 0.869, followed by Random Forest and L1. Figure 6 compares the models' performance to the baseline accuracy. An interesting observation is the near-zero standard deviation for each model. This is due to TimeSeriesSplit, which ensures consistent training and test sets across splits. While some models, like Random Forest and XGBoost, have inherent randomness, only Random Forest shows noticeable variability in results. Consequently, the overall test scores exhibit minimal standard deviation.

Table 2: Model Performance Comparison

	Baseline	L1	L2	Random Forest	KNN	XGBoost
Accuracy	0.761	0.862	0.858	0.864	0.857	0.869

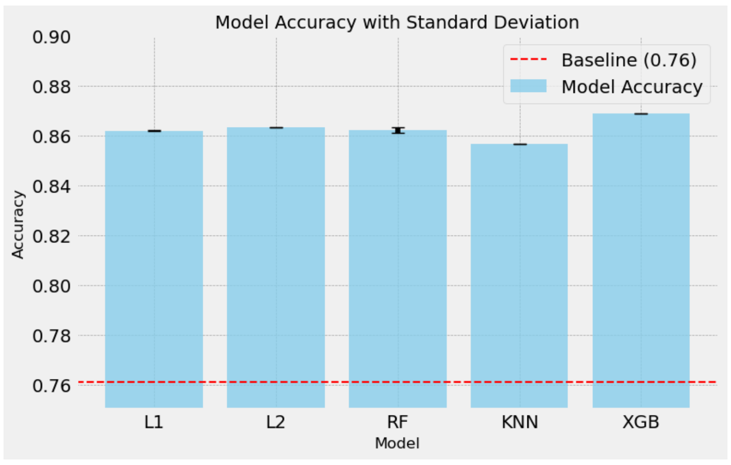


Fig. 6. A bar plot showing the test scores of different ML models compared to the baseline accuracy.

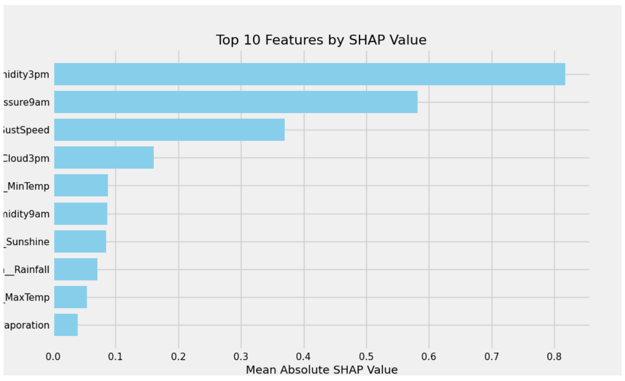
The optimal parameter set for our XGBoost model is: $reg\_alpha : 0.1, reg\_lambda : 0.0$

4.2 Global Feature Importance

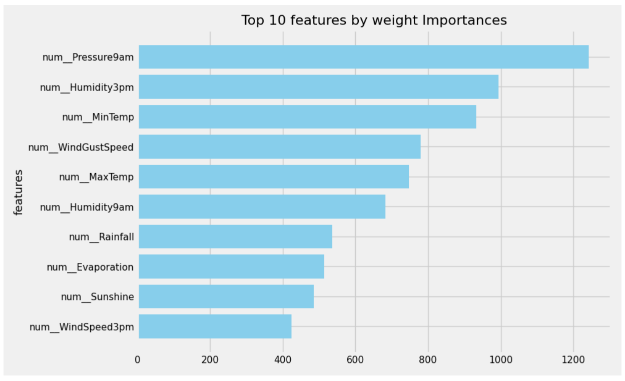
In the global feature importance section, we measured three different global importance metrics. They are SHAP feature importance, weight importance in XGBoost, and permutation importance. As we can see from Figure 7a, the top three features by SHAP importance are Humidity3pm, Pressure9am, and WindGustSpeed. For weight importance, as shown in Figure 7b, the top three features are Pressure9am, Humidity3pm, and MinTemp. For permutation importance, illustrated in Figure 7c, the top three features are Humidity3pm, WindGustSpeed, and Pressure9am.

Table 3: Top 3 Feature Overlap Across Different Importance Metrics

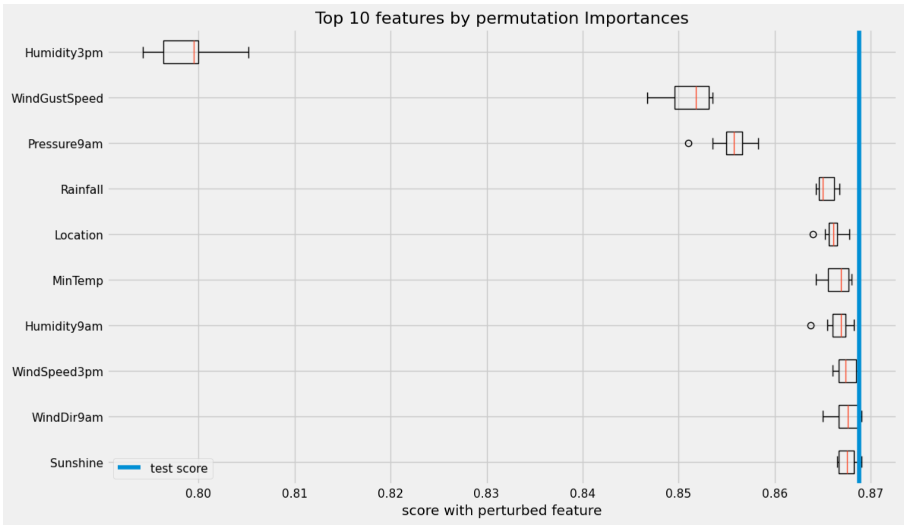
Rank	SHAP Importance	Weight Importance	Permutation Importance
1	Humidity3pm	Pressure9am	Humidity3pm
2	Pressure9am	Humidity3pm	WindGustSpeed
3	WindGustSpeed	MinTemp	Pressure9am



(a) Top 10 features by SHAP importance



(b) Top 10 features by weight importance in XGBoost



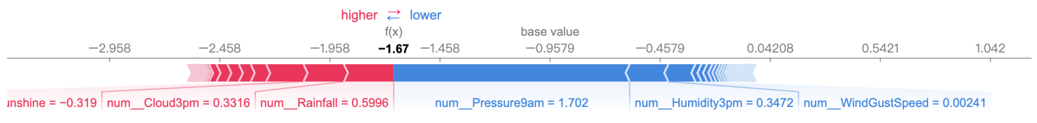
(c) Top 10 features by permutation importance

Fig. 7. Global feature importance using three different metrics.

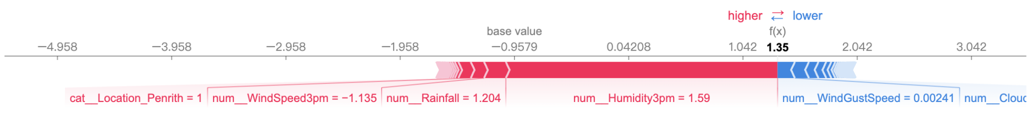
Humidity and pressure are two very important features for predicting rain tomorrow, as seen from the above analysis. The other important features are related to wind, pressure, and temperature.

4.3 Local Feature Importance

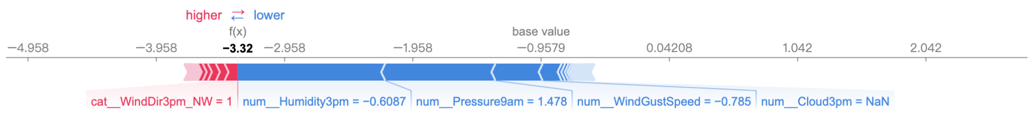
In the local feature importance section, we calculated SHAP[4] values and created three different SHAP force plots to interpret our model's predictions. For the first data point (Figure 8a), Pressure9am, Humidity3pm, and WindGustSpeed push our prediction from baseline -0.96 to -1.67. For the second data point (Figure 8b), we have a prediction above the baseline, with Rainfall, Humidity3pm, and WindSpeed3pm pushing the value to 1.35. For the third data point (Figure 8c), Humidity3pm, Pressure9am, and WindGustSpeed push our prediction below the baseline to -3.32. Our local feature importance results are quite consistent with our global feature importance findings, since features related to pressure, humidity, and wind again play very important roles in the model's predictions.
Note that we calculated SHAP values for XGBoost. We observed that the SHAP values for XGBoost focus on the logits instead of the probability of the model's predictions. This is the reason why we observe prediction values outside the $0 - 1$ range.



(a) SHAP force plot for index 2 point in the test set.



(b) SHAP force plot for index 4 point in the test set.



(c) SHAP force plot for index 200 point in the test set.

Fig. 8. Local feature importance visualized using SHAP force plots for selected test points.

5 Outlook

Our ML pipeline uses feature reduction to address significant missing values, making it impractical to train on the entire dataset. Thus, we utilized only a fraction of the data. Incorporating more data points is a potential way to enhance the model's predictive power.

Since we selected accuracy as our evaluation metric, we observe from Figure 9 that the False Negative rate is relatively high. To mitigate this issue, we will consider adopting alternative evaluation metrics, such as the f_β score, which balances precision and recall according to specific needs. Furthermore, we may explore models with stronger predictive capabilities, such as Multi-Layer Perceptrons (MLP) and Long Short-Term Memory (LSTM) networks.

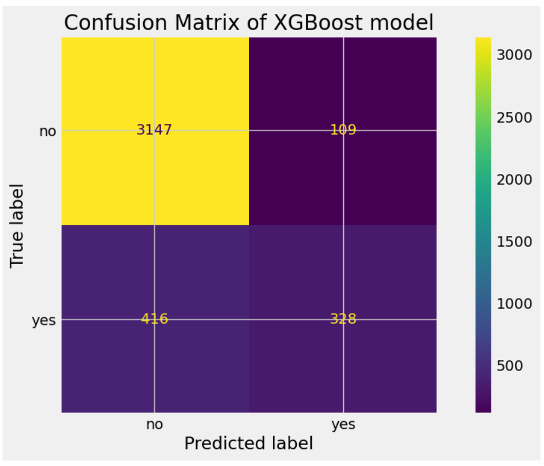


Fig. 9. Confusion matrix of our XGBoost model.

For improving model interpretability, we plan to apply techniques like LIME, which generates local surrogate models to explain individual predictions. Additionally, Individual Conditional Expectation (ICE)[3] plots will be used to illustrate how the predictions for individual instances change as a specific feature varies. To capture the global marginal effects of one or two features on the predicted outcomes, we will employ Partial Dependence Plots (PDPs)[2].

References

[1] Chandrima D. *Rainfall Prediction: 7 Popular Models*. Tech. rep. Available at https://www.kaggle.com/code/chandrimad31/rainfall-prediction-7-popular-models/notebook. Kaggle, 2020. (Visited on 12/07/2024).

[2] Jerome H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.

[3] Alex Goldstein et al. "Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation". In: *Journal of Computational and Graphical Statistics* 24.1 (2015), pp. 44–65. DOI: 10.1080/10618600.2014.907095.

[4] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017), pp. 4765–4774. URL: https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c4
Paper.pdf.

[5] Fahad Mehfooz. *Rain Prediction with 90.65% Accuracy*. Tech. rep. Available at https://www.kaggle.com/code/fahadmehfoooz/rain-prediction-with-90-65-accuracy. Kaggle, 2021. (Visited on 12/07/2024).

[6] Maytal Saar-Tsechansky and Foster Provost. "Handling Missing Values when Applying Classification Models". In: *Journal of Machine Learning Research* 8.57 (2007), pp. 1623–1657. URL: http://jmlr.org/papers/v8/saar-tsechansky07a.html.