Rain in Australia

Sudarshan Damodharan

April 2024

1 Exploratory Data Analysis

In my analysis of the dataset on weather conditions, I began by loading the data from a CSV file and immediately addressed missing values in the 'RainTomorrow' column, which is crucial for my target prediction. I removed rows where this variable was missing, ensuring that my model wouldn't have to handle these uncertainties.

I then proceeded to split the dataset into features (X) and the target variable (y). To better understand the structure of the data, I identified which columns had missing values and which were completely filled. This differentiation was important for deciding how to handle missing data and preparing for any necessary encoding for categorical variables.

To address the categorical variables with missing data, I implemented a method to randomly fill these gaps using observed values. This approach introduced some noise but was a practical compromise to retain as much data integrity as possible.

Additionally, I transformed the 'Date' column into more analytically useful cyclical features using sine and cosine transformations. For the quantitative features with missing values, I imputed these with the mean.

Finally, I encoded categorical variables and generated additional derived features to better capture temporal patterns and interactions among variables such as temperature and humidity. After scaling the data, I split it into training and testing sets.

2 Feature Analysis

In this section of my analysis, I focus on identifying the most influential features in predicting whether it will rain tomorrow using a LassoCV model, which is a type of linear model that includes L1 regularization. The regularization helps in feature selection by shrinking coefficients of less important features to zero, thus helping me focus on the most relevant predictors.

First, I initialize the LassoCV model specifying 5-fold cross-validation to ensure that my model's performance is robust and not overly fitted to a particular subset of the data. I then fit this model to the training data. The .ravel() method is used on y-train to convert it from a column vector to a 1D array, which is a format suitable for the model fitting process.

Once the model is trained, I extract the coefficients and identify those features whose absolute coefficients are greater than or equal to 0.01. This threshold helps me to focus only on features with a significant impact on the outcome, ignoring those with negligible effects.

After identifying these important features, I proceed to analyze their interrelationships through a correlation matrix. I append the target variable 'RainTomorrow' to the list of significant features

and then create a new dataframe combining both the features and the target. The correlation among these selected variables is calculated, which provides insight into how these variables interact with each other.

I visualize this correlation matrix using a heatmap, where the matrix is displayed with varying colors representing different levels of correlation. This visual aid helps me quickly identify highly correlated features, which could be indicative of redundant information or potential multicollinearity issues that might affect the model's performance.

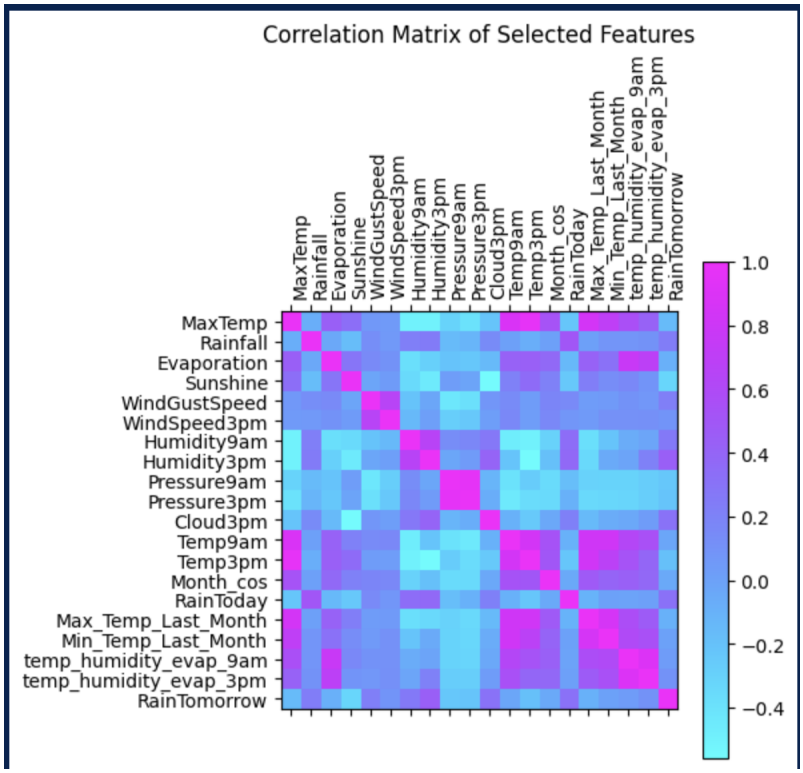

Figure 1: Correlation Matrix

Overall, this part of my analysis helps me refine the feature set for better predictive modeling and understand the dynamics between key variables in the context of predicting rain on the following day.

3 Model Building and Evalatuation

In this section of my project, I employed a multi-layer neural network to predict rainfall on the following day, using a fully connected architecture. The network comprises two hidden layers with tanh activations and a sigmoid output layer for binary classification. I defined the model in a Python class using PyTorch's nn.Module, which makes it easy to build and train neural networks.

To begin, I initialized the network with specific input, hidden, and output sizes. The training process involved preparing the data as tensors, setting up a loss function (BCELoss for binary cross-entropy), and choosing an optimizer (Adam), which is effective for these types of networks due to its adaptive learning rate capabilities.

I trained the model over 100 epochs, printing out the loss at every tenth epoch to monitor the training progress. This allowed me to ensure that the model was learning effectively over time.

After training, I evaluated the model on a test set, converting the sigmoid outputs to binary predictions to calculate the accuracy.

Further refining the model involved hyperparameter tuning through grid search, evaluating combinations of hidden layer sizes and learning rates using cross-validation. This approach helped me find the optimal parameters that yielded the highest validation accuracy, thereby enhancing the model's performance.

Finally, I retrained the model with these optimal parameters and again tested its performance on the test data. The results were quantified not just in terms of accuracy but also precision, recall, and F1 score, providing a comprehensive view of the model's predictive power. The confusion matrix provided further insights into the true positive and false positive rates, critical for understanding the model's behavior in a real-world scenario.

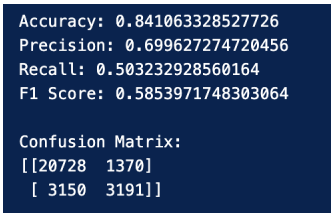

Figure 2: Correlation Matrix