1. DATASET

The dataset contains weather information collected through observations from various weather stations in Australia. The dataset is a mix of records containing categorical and numerical variables like wind pressure, wind direction, humidity, minimum and maximum temperature for the day, sunshine, evaporation. The objective of this research is to predict whether it will rain the next day. The dataset contains 142193 entries describing the daily weather conditions over a period of eight years. The information about rainfall on present day is also given. The link for the dataset is given below.

https://www.kaggle.com/jsphyg/weather-dataset-rattle-package/downloads/weather-dataset-rattle-package.zip/2

2. RELATED WORK

The work done by Kaggle users on this dataset includes building machine learning algorithms such as decision tree, support vector machine, random forest, naive bayes classifier, logistic regression. Most of the work focussed on building different models and comparing their performance based on classification accuracy. The most common comparison was found to be between logistic regression, support vector machine and random forest algorithms, with random forest algorithm coming out on top frequently with accuracy of about 84-85%. The methods used to improve performance of these algorithms included hyper parameter tuning using grid search followed by cross validation. This helps to find the best parameters for the model to be trained.

Some of the Kernels also include models trained with k nearest neighbour and neural network algorithms. These algorithms were not a good fit for this classification problem, as they overfitted the training data and hence were not suitable to be applied to test data for classification. In some cases, the algorithm did not generate good results, as clustering using k nearest neighbour algorithm was not well suited for this problem.

As discussed by the Kaggle users working on this dataset, the variable representing risk of rainfall on the next day is to be excluded while training the model. This variable gives amount of rainfall in millimetres for the next

day, which would leak the future information to the model. This will give a false impression that the model is performing classification with very high accuracy.

3. RESEARCH GAPS AND PROPOSED WORK

The previous work done on this dataset includes training classifiers using algorithms such as logistic regression, naive bayes, decision tree, random forest. In order to improve performance of these classifiers, hyper parameter tuning techniques were used. The most popular technique was grid search along with cross validation to determine the best parameters for the model. The class distribution for the target variable 'RainTomorrow' is not balanced in the dataset.

In the proposed research, class imbalance issue is handled using upsampling technique. Then, randomised search with cross validation is used, to determine the best parameters for the model by running multiple iterations of the search. The best parameters obtained in this step are used to train the random forest classifier and then to make predictions using test data for evaluating the model.

4. IMPLEMENTATION

Preprocessing

Handling Missing Values:

After loading the dataset and examining values for each column, it was found that there were missing values (NA) present in the dataset. The records containing these missing values (29268) were not considered for this study, reducing the total number of observations to 112925.

Removing Outliers:

Z-score normalisation technique was used to detect outliers in the distribution of the dataset. The outliers detected (5057) were removed from the dataset, as they might influence the rain forecast outcome. After dealing with outliers, the cleaned dataset had 107868 records.

One-Hot Encoding:

The categorical variables were converted to numerical format by representing them as two dimensional arrays. This process is one-hot

encoding, and it is necessary as the model is to be trained using numerical data. The categories cannot simply be represented using numbers as it gives false impression of records having higher number values to be more significant. ‘RainToday’ and ‘RainTomorrow’ had values ‘yes’ and ‘no’. They were represented using 1 and 0. While variables ‘WindGustDir’, ‘WindDir3pm’, ‘WindDir9am’ were represented using a two dimensional array as they had possible values with more than two categories.

Standardisation:
The numerical variables in the dataset were standardised using min-max technique. This helps to get all the inputs in a range between 0 and 1, which is always ideal for a binary classification task.

Handling Class Imbalance:
The class distribution for the target variable ‘RainTomorrow’ is imbalanced, as nearly 77% of the samples have the target value as 0. To handle this issue, upsampling technique is used, where instances for minority class having target value as 1 are generated randomly to match with the number of instances of majority class.

Feature Selection:
Feature selection is used to identify the most important features that can be used to train the classifier. This helps to reduce the number of inputs to the model and hence the training time. The SelectFromModel API of Scikit-learn is used to perform feature selection. It assigns weights to each input feature based on their importance. Only those features are selected that can explain the variance in the outcome variable effectively. The model is trained using only these features instead of all the features in the dataset.

The most important features identified in this process are listed below.
MinTemp
MaxTemp
Rainfall
WindGustSpeed
WindSpeed9am
WindSpeed3pm
Humidity9am
Humidity3pm

Pressure9am
Pressure3pm
Temp9am
Temp3pm

As discussed earlier, the variable representing risk of rainfall on the next day was excluded from the set of input features. This variable specifies amount of rainfall in millimetres for the next day, which would leak the future information to the model and influence the predictions made by the model.

The columns ‘Evaporation’, ‘Sunshine’, ‘Cloud9am’, ‘Cloud3pm’ had many missing values with less than 60% of records having actual data. Hence, these columns were not considered for training the model. Also, since this research does not concern rain forecast specific to any date or location, location and date columns are dropped from the set of input features.

Model Selection

Random Forest Classifier:
The data to be used as input for the classifier is divided in two parts, training set and test set. For training, a large proportion of the data is to be used, as it helps the model to learn different features and patterns present in the data. We use 70% of the data for training and the rest for testing. A random forest algorithm is used to predict whether it will rain on the next day.

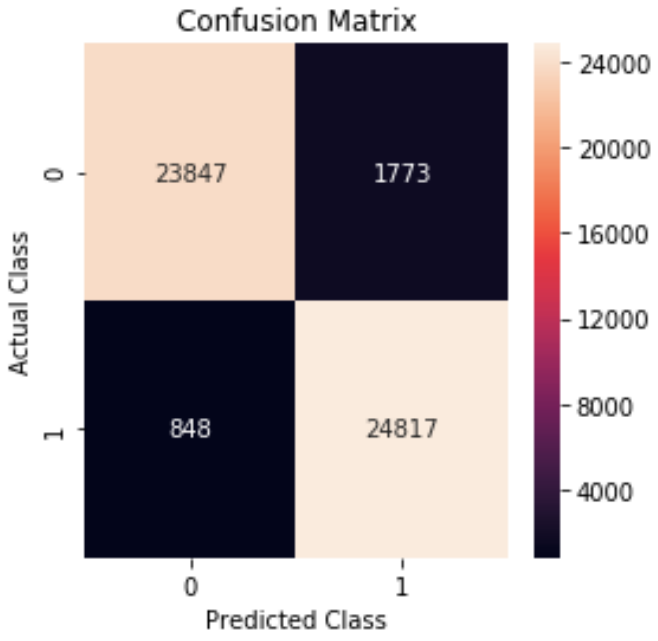
The base classifier was evaluated using test data. The classification accuracy achieved by the model was over 93%. This was the default implementation of the random forest classifier without explicitly specifying the values for hyper-parameters to the model.

In order to improve the performance of the classifier, the best hyper-parameters for the model were determined using randomised search method with cross validation. A three fold cross validation with thirty iterations were used to perform the randomised search to find the best attributes. The best parameters estimated by the randomised search were as follows:

Number of trees in the forest: 1200
Number of features to consider at every split: Set to ‘auto’.
Maximum number of levels in tree: 50
Minimum number of samples required to split a node: 2
Minimum number of samples required at each leaf node: 2
Method of selecting samples for training each tree: Bootstrap set to false

5. MODEL EVALUATION

The model trained using input features was used to make predictions using the test dataset. These predictions were compared against the actual values for classes in the test dataset to determine how effective this classification model is in predicting the probability of rainfall on next day. True positives (TP) is the number of instances classified as positive by the classifier that are actually positive. True Negatives (TN) is the number of instances classified as negative by the classifier that are actually negative. False positives (FP) is the number of instances classified as positive by the classifier that are actually negative. False negatives (FN) is the number of instances classified as negative by the classifier that are actually positive. These values are displayed using a confusion matrix as shown below.



Following metrics are used to assess the performance of the model based on predictions made using test data.

Classification Accuracy:

Classification accuracy is defined as the number of instances classified correctly from the overall predictions made by the classifier. It can be calculated as (TP+TN)/(TP+TN+FP+FN). In this case, the classification accuracy when the model is evaluated using test dataset is 94.88%. This shows that the model classified nearly 95% of the inputs correctly.

Precision and Recall:

Precision is defined as the fraction of positive instances classified correctly by the classifier. It can be calculated as (TP)/(TP+FP). In this case, the precision when the model is evaluated using test dataset is 93.33%. This means, the model predicted outcome as 1 for over 93% of the inputs where the actual outcome was also 1. Recall is defined as the fraction of instances classified as positive by the classifier that are actually positive. It can be calculated as (TP)/(TP+FN). In this case, the recall when the model is evaluated using test dataset is 96.69%. In this case, recall is the most important metric for evaluating the model, as correctly predicting whether it will rain the next day is an important factor to consider. High recall value shows that the model is good at forecasting rainfall for the next day.

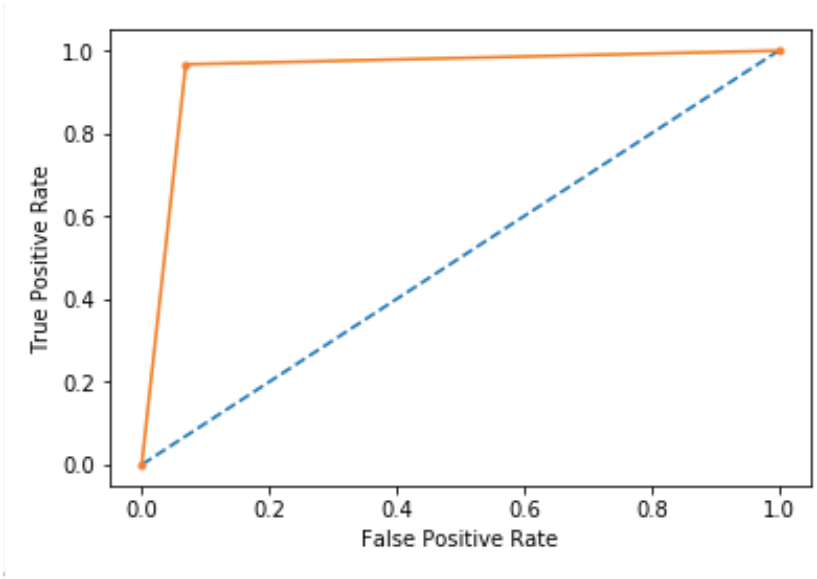
F1 Score:

F1 score is defined as the harmonic mean between precision and recall. As precision increases, recall decreases, and vice versa. F1 score finds an optimal trade-off between precision and recall. It can be calculated as 2*(Precision*Recall)/(Precision+Recall). In this case, the F1 score when the model is evaluated using test dataset is 94.98. This shows a good blend between precision and recall.

Receiver Operating Characteristics:

Receiver Operating Characteristics (ROC) plot shows relationship between precision and recall. In this plot, true positive rate (TP/TP+FN) is plotted on Y-axis while false positive rate (FP/FP+TN) is plotted on X-axis. The effectiveness of ROC is measured by calculating the total area under the

curve (AUC) in ROC plot. Higher the AUC, better is the blend between precision and recall for the model. ROC for this task is shown below.



In this case, the value AUC after evaluating the model using test dataset is 0.949. This value shows that proportion of false positives is very less compared to the proportion of true positives. The curve is very close to the top left corner of the graph, which represents a classifier that often predicts the outcome correctly.

6. DISCUSSION

The results achieved by Kaggle users as demonstrated by the work done in kernels show that random forest algorithm came out on top in most of the experiments with classification accuracy of up to 85%. The methods used to train the classifier were different in each kernel, with some using default implementation for training the classifier while some using hyper-parameter optimisation techniques like grid search to find the best parameters for the model.

Grid search for finding the best hyper-parameters for the model is an effective method of improving performance of the classifier. However, as grid search uses every possible combination of parameters in the grid to

evaluate the model and find the best set of hyper-parameters, it is computationally very expensive. Also, the class distribution of the target variable in the dataset is not balanced. This can lead to a model that gives a good classification accuracy for the class that has majority in the distribution. Handling the class imbalance problem can improve the classification accuracy of the model, and it makes the decisions made by the model more sensible, considering it has been trained on data having equal distribution of all the possible classes.

In this task, the class imbalance problem was handled using upsampling technique. Also, randomised search is used instead of grid search, as it is computationally less expensive with respect to time and resources required for execution. The classification accuracy achieved with this technique was 94.88%, which is better than results of any of the previous work done by Kaggle users. Although grid search optimisation may achieve even better results, random search was used to reduce training time and computation cost for the algorithm.