Rain in Australia:

The objective is to predict whether rain will come tomorrow or not based on the location's weather conditions. The dataset provides us with various variables such as Wind speed, humidity, temperature, pressure etc. Occurrence of rain is indicated by 1 and non-occurrence is indicated by 0. Independent variables are selected based on their correlations and through few trial and error methods.

Reason for selecting the Dataset:

Rain in Australia is an interesting dataset because we study the weather patterns and the features are diverse and challenging to understand and build models. It has the right number of training samples and features for building various learning algorithms and experiment on them. It provides the right learning platform.

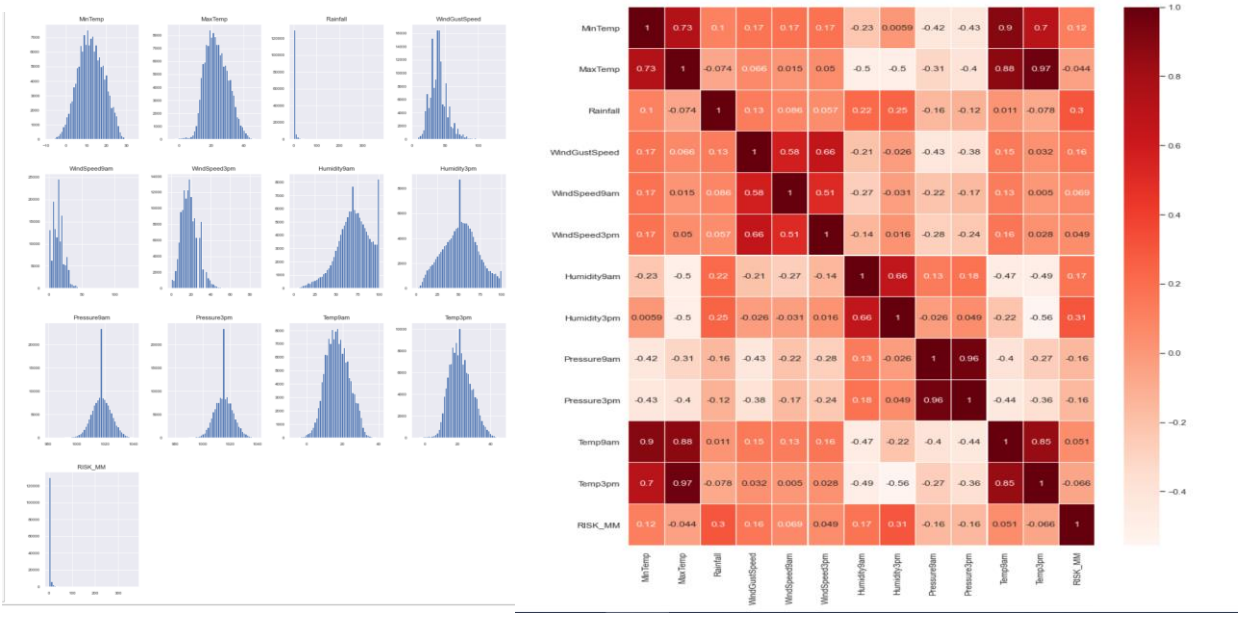
Link to Dataset: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package

Independent Variables: Mintemp, Maxtemp, Rainfall, WindGustSpeed, WindSpeed9am, WindSpeed3pm, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, Cloud9am, Cloud3pm, Temp9am, Temp3pm, Raintod

Dependent Variables: Raintmr (Binary variable 0s and 1s)

- There are null values in the dataset and they are removed using dropna () function. Min-max
- normalization is done on all the selected features.
- The dataset is divided into 70/30 split using train_test_split from sklearn. The 70 % of the data is used for the training the model and remaining 30% is used for testing.
- We have removed fields like Evaporation, Sunshine, Cloud3pm, Cloud9am since it has more than 50% of null value.

Histogram and Correlation Matrix:



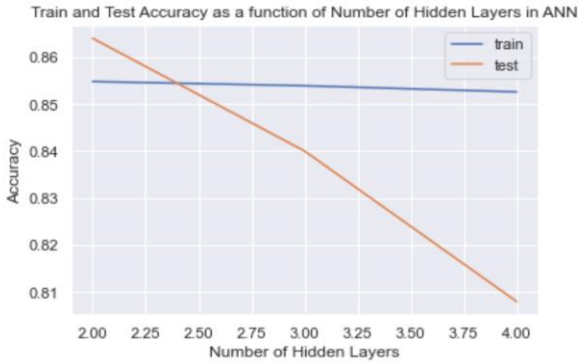
In selecting a suitable model, binary classification techniques based on Logistic Regression, Decision Tree, Random Forest are used. Also, Neural Network with different Kernel function and activation function and SVM with linear function is used.

Artificial Neural Networks:

Tensorflow and keras packages in python is used to implement Artificial Neural Networks with various hyperparameters and activation functions.

Experimentation with various number of Hidden Layers:

We tried with increasing the number of hidden layers in ANN to find the optimum number of layers at which model generalizes well. The Below graph shows the train and test accuracy for various number of hidden layers in the left and right shows the confusion matrix for ANN model with 3 hidden layers.



[12 92]]

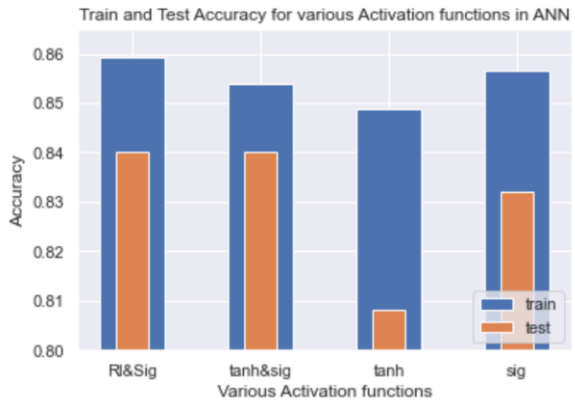
	precision	recall	f1-score	support
0.0	0.82	0.92	0.87	61
1.0	0.91	0.81	0.86	64
accuracy			0.86	125
macro avg	0.87	0.87	0.86	125
weighted avg	0.87	0.86	0.86	125

0.864

From the graph, we can clearly see that, if we increase the hidden layers more than 3 then there is no increase in performance. The accuracy for model with 3 hidden layers is 86.4%.

Experimentation with various activation functions:

We have experimented mainly with various combinations of activation functions such as sigmoid and tanh because they perform well for binary classifications with 0s and 1s which is similar to our problem. To compare them with other functions, we have used Relu activation function as well. The below bar chart on left show the training and test accuracy for the various combinations of activation functions and the right shows the confusion matrix with Relu function for hidden layers and sigmoid function for output layer.



	precision	recall	f1-score	support
0.0	0.81	0.92	0.86	61
1.0	0.91	0.80	0.85	64
accuracy			0.86	125
macro avg	0.86	0.86	0.86	125
weighted avg	0.86	0.86	0.86	125

0.856

From the chart, it is clear that the combination of relu at hidden layer and sigmoid at output layer performed the best. The accuracy increased to 85.6%.

Decision Tree:

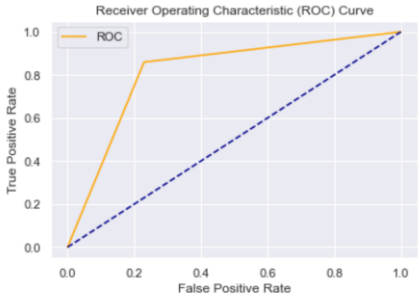
To implement Decision Tree as learning algorithm on the train data set, sklearn package in python is used. We can also work with Gini index for root node as Gini index method works best when the target variable is binary, but to reduce complexity, we have not done Gini.

Below is the Classification report along with the confusion matrix and test accuracy score:

```
[[47 14]
 [ 9 55]]
              precision    recall  f1-score   support

           0       0.84      0.77      0.80        61
           1       0.80      0.86      0.83        64

    accuracy                           0.82       125
   macro avg       0.82      0.81      0.82       125
weighted avg       0.82      0.82      0.82       125
```



We can see that accuracy is approx. **82%** for this data set and hence there are less or no chances of overfitting.

```
{'max_depth': 6, 'max_leaf_nodes': 8, 'min_samples_split': 2}
0.904
```

With the help of Grid Search, we can determine that for the given data set, max depth will be 6 and max number of leaf nodes will be 8.

Support Vector Machines:

To implement SVMs on the data set Sklearn package in python was used with kernels function Linear. Another parameter that is C(Regularization parameter) with different values was implemented for Linear kernel in order to find the best C at which model best fits the data and performs classification at its best. The significant value of regularization parameter was chosen from different values such as 0.001, 0.01, 0.1, 1, 10, 100 which gives the lowest misclassification rate on test data**.**
To perform SVM as learning algorithm, initially Linear Kernel is used on training data set that is 70% of the data set with various values of C (Regularization parameter) such as 0.001, 0.01, 0.1, 1, 10,100. After implementing Linear kernel, cross-validation with the value of k-fold as 6 is applied in the training data set in order to obtain train score and test score. All the test and train score was then averaged and model was evaluated on the remaining 30% test data set. The algorithm performs well with C=10 as we are getting maximum test accuracy 0f approx. **86%** at this value.
The model does not overfit

```
train: [0.504, 0.8133333333333334, 0.84, 0.8453333333333334, 0.848, 0.848]
test: [0.488, 0.832, 0.848, 0.856, 0.856, 0.856]
Index: [0.001, 0.01, 0.1, 1, 10, 100]
```

```
Best parameters: {'C': 10}
Best cross-validation score: 0.85
```

```
Cross-validation scores: {} [0.9047619  0.73015873 0.88888889 0.87096774 0.82258065 0.82258065]
```

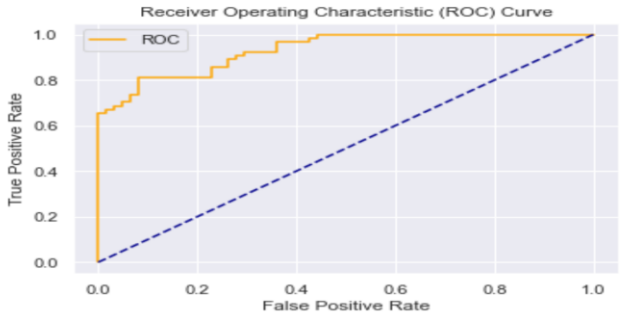
The above screen shows cross validation score for the same dataset for different values of K.

Logistic Regression:

A logistic model is motivated from an assumption that the log odds of an (label) event, that is Ln (p/1−p), can be modeled as a function of a vector of features and a vector of parameters associated with these features.

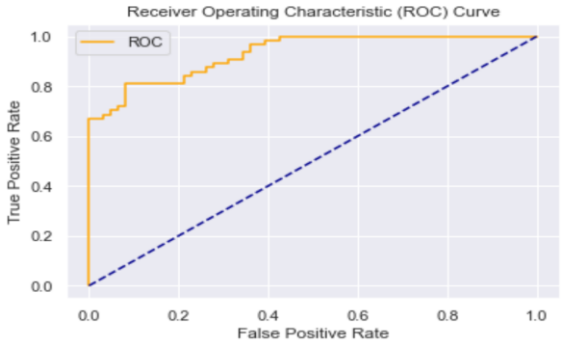
The train score for Logistic Regression is approx **85%** and test score is almost **86%**. This says over model fits nicely to the data set and also there is not much difference between these scores.

```
Train score: 0.8427
Test score: 0.8640
```



Further “saga” solver is used with the same model but there was not much improvement found in both train and test data.

```
Train score: 0.8453
Test score: 0.8640
```



Furthermore, gridsearch and cross validation are used. Below pictures shows result for grid search and cross validation scores:

```
GridSearchCV(cv=6, estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100],
                         'penalty': ['l1', 'l2']},
             return_train_score=True)
```

```
Best parameters: {'C': 1, 'penalty': 'l2'}
Best cross-validation score: 0.84
```

```
Cross-validation scores:[0.9047619  0.73015873 0.88888889 0.85483871 0.82258065 0.82258065]
```

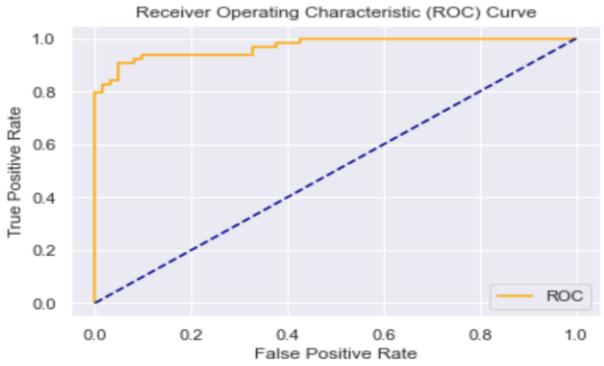
Best cross-validation score is approximately 84%.

Random Forest:

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

For random forest, we have worked with confusion matrix, F1 score and Roc curve with n_samples = 200. The F1 score is as high as 91.20%.

```
Confussion matrix: [[58  8]
 [ 3 56]]
F1 score: 0.9105691056910569
0.912
```



Conclusion:

We tried different algorithms like Artificial Neural Networks, Decision Tree, Support Vector Machines, Logistic Regression and Random Forest to fit the data set and noted its accuracy for Train and Test dataset. We also tried doing ROC for the algorithms.

Algorithm	Score
ANN (relu and sigmoid)	85.6%
ANN (tanh and sigmoid)	86.4%
ANN (tanh)	80.8%
ANN (sigmoid)	83.2%
Decision Tree	81.6%
SVM	91%
Random Forest	91%

We can conclude with this that there is not much difference in all the algorithms selected, and all the algorithm has score above 80% which states our model fits well. Also, there is no proble of over fitting or under fitting with any of these algorithms.

Random Forest, Support Vector Mechans and Neuran Networks with tanh and sigmoid functions gives best score where as Decision Tree and Neural Networks with only tanh function gives us worst score compared to all this.