MACHINE LEARNING CHALLENGE

Unit Chair: Dr Musa Mammadov
Submission Date: 5:00PM Friday of Week 10

Table of Contents

Dataset Name: Rain in Australia

Group Name: Mon-13 (FANH) **On Campus/Cloud**: On Campus

STUDENT ID	STUDENT FULL NAME	Individual contribution*
218401269	ALEXANDER PAK YU LAI	5
218241616	HARRY WILLIAM LODGE	5
218459058	VIET NAM NGUYEN	5
218271795	JARROD KENG YEN YONG	5

* 5 - Contributed significantly, attended all meetings
4 - Partial contribution, attended all meetings
3 - Partial contribution, attended few meetings
2 - No contribution, attended few meetings
1 - No contribution, did not attended any meetings

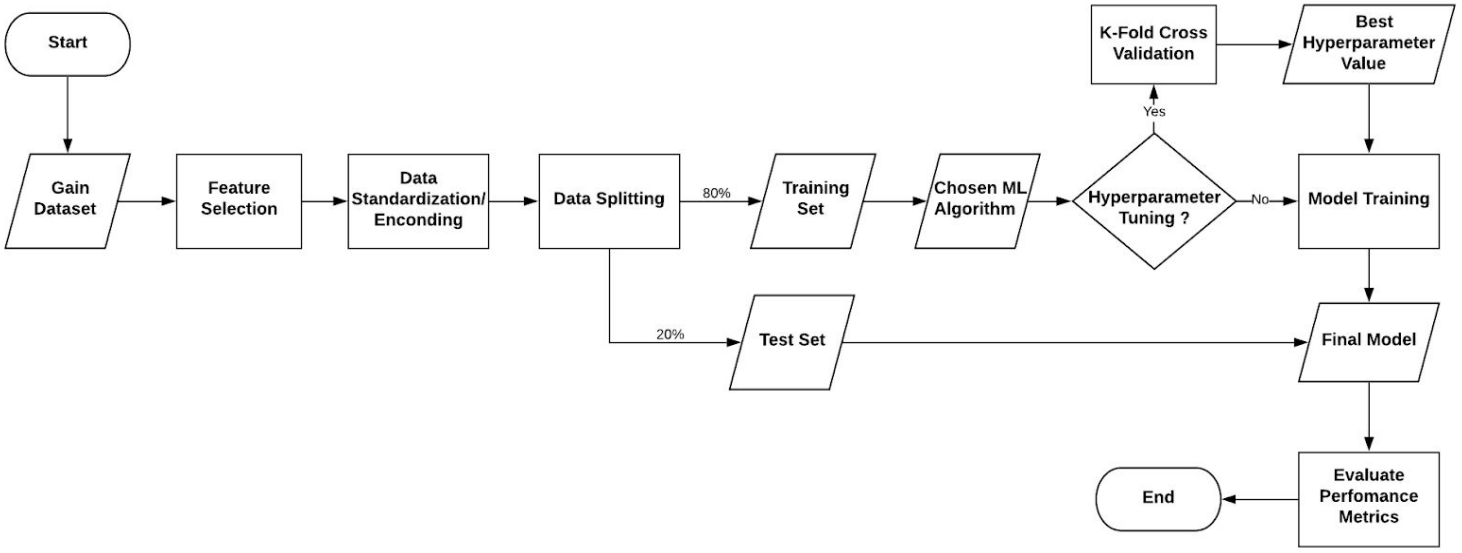
Section 1: Brief Summary & ML Problem Formulation

In the previous assignment, the group was tasked to use the dataset “Rain in Australia” and create a report on the analysis and testing of the dataset. The observations found in the previous report is that many pairs of variables have a strong or high correlation in the positive trend in relation to the field of rain predictions, the data patterns show that humidity and pressure can help predict rain. Through the testing training done by importing linear regression as min and max temp a score of 79% accuracy was given to the model.

The aim of applying machine learning methods to the dataset is to show how accurate the dataset is in predicting rainfall tomorrow and to determine what machine learning method best fits the dataset.

In this report, there are 9 models that were used to train the data. The classification models used are KNN, PCA & XGB, Naïve Bayes, Decision Tree classification and Random forest. The Regression algorithms used are Linear, Logistic and last a Neural network approach was also used to test and train the data. The reason why so many models were used is to see which one fits the best and outputs the most accurate and reliable information for real life scenarios.

The machine learning flowchart below illustrates the step-by-step approach of training and testing models.



Section 2: Results and Discussion

1. Classification Models

A. Feature Selection

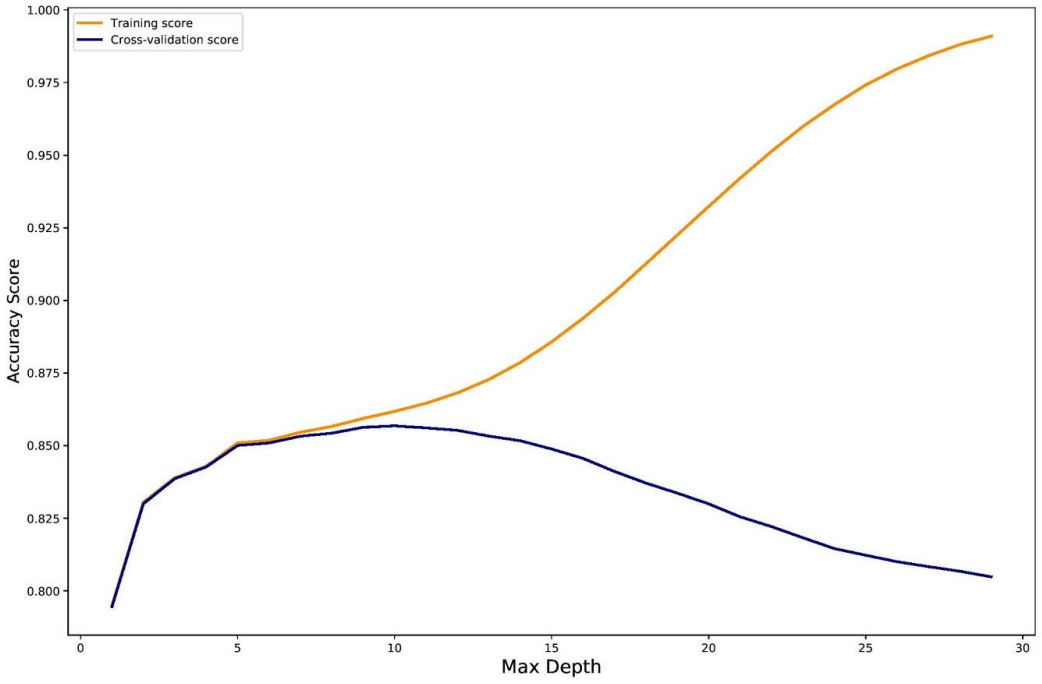
Mutual information between each feature and target (RainTomorrow) can be metrics to sort out the top features for improving model's performance and remove redundant variables. Thanks to the "sklearn" library, the mutual information of each pair between feature and "RainTomorrow" was calculated easily. As a result, the top five features with the highest mutual information scores were chosen to train and test models. They were "Humidity3pm", "Pressure9am", "Pressure3pm", "Rainfall" and "WinGustSpeed".

B. Naive Bayes

Before the dataset fit to the Naive Bayes model, "Yes" and "No" in the target variable were assumed as 1 and 0 respectively. To avoid overfitting, the dataset with top five features above needed to be split where test and training set were in a ratio of one to four. After the model experienced the training process, the train-set and test-set accuracy scores were similar at 0.81. In addition, the precision and recall for class 0 were higher than class 1, which interpreted that the model predicted class 0 more correctly. Overall, the accuracy score was high, so it could be one of the suitable models to determine whether it will rain tomorrow or not.

C. Decision Tree

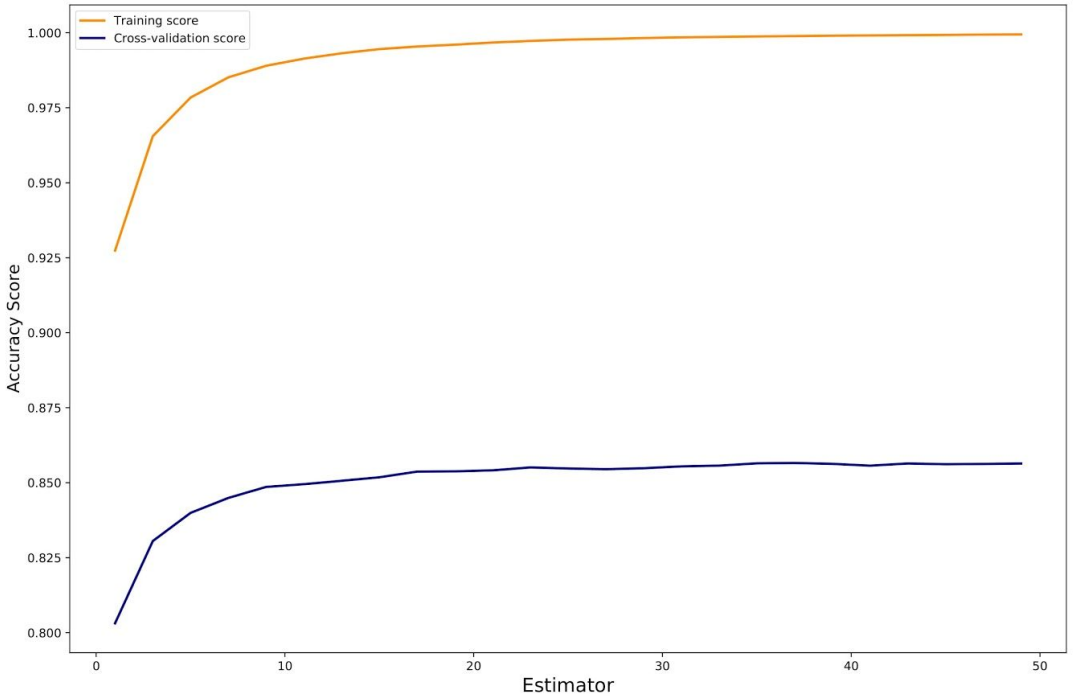
The Decision Tree model is able to process the categorical target, so encoding the target is unnecessary. However, since the numbers in most of the top five features were quite big, the dataset set with these features were applied "MinMaxScaler" technique. According to Hale's article (2019), "MinMaxScaler" reduces the range of each feature down to between 0 and 1 by taking the smallest value out from each value in the feature, and dividing the difference between the biggest and smallest in the feature. The dataset was split in the same strategy used in the Naive Bayes model. Since the Decision Tree model accepts the value of maximum depth as a hyperparameter, the most proper maximum depth can help the model not be overfit or underfit. Therefore, applying 10-fold cross validation on the training set with a range of maximum depth from 1 to 30 resulted in the best fit hyperparameter of 5 in depth.



The result of accuracy scores after the model was trained with value of 5 in depth was 0.85. In addition, the precisions of class “No” and “Yes” were 0.88 and 0.64, and the recalls of them were 0.91 and 0.57 in that order, so these scores illustrated the better prediction on class “No” than “Yes”. Because of high accuracy scores and no overfitting as well as underfitting, the Decision Tree can predict accurately the status of Australian weather tomorrow.

D. Random Forest

The Random Forest is quite similar to the Decision Tree, but the big difference is that it entails many individual decision trees, one of which is called as ensemble, and the best decision tree is selected as the final model (Yiu 2019). The beginning steps the model underwent are the same as the Decision Tree, such as “MinMaxScaler” standardization technique and data splitting with a ratio of 1 to 4. The most time-consuming part is to find the most suitable number of ensembles. This search was completed by 10-fold cross validation on the training set with the number of ensembles varying from 1 to 50.



The above graph shows no intersection between two lines of training and cross-validation. Similarly, they intended to run parallel to each other from 1 to possibly infinity. Therefore, the number of ensembles of 3 could be the best one to reduce overfitting as much as possible. Accordingly, the accuracy scores of the model on the training set and test were 0.96 and 0.83 respectively. Also, the precision and recalls stated that the model predicted better on class “No” than “Yes" since 0.88 and 0.64 were the precisions of class "No” and "Yes", and 0.91 and 0.57 were the recalls of these classes respectively. Overall, because of the big difference between train-set and test-set accuracy scores, the Random Forest model can return overfitting and less accurate prediction and the usage of this model should be considered carefully.

E. K-Nearest Neighbors

KNN chooses the k closest neighbors and then based on these neighbors, assigns a class (for classification problems) or predicts a value (for regression problems) for a new observation.

To start KNN all the needed procedures were imported, specifically the KNeighboursClassifier and train test split. Once that was done, import the cleaned data and change all values of no and yes to 0 and 1 respectively.

To make the training and testing better for this we now drop RiskMM, Date and Location as they didn't play a huge role in the data. As RainTomorrow and RainToday were the only columns with yes and no values we can now change that to a category type for later use.

```
# here we change the data type for these two column
df['RainTomorrow']=df['RainTomorrow'].astype('category')
df['RainToday']=df['RainToday'].astype('category')
```

```
# here we transform them to numeric values.
df['RainTomorrow']=df['RainTomorrow'].cat.codes
df['RainToday']=df['RainToday'].cat.codes
```

```
 # Identifying number of row and columns .
df.shape
```

```
(142193, 18)
```

The target of this was the "RainTomorrow" so we would need to transform all the non-numeric features and drop them beforehand to make the model.

Whilst training and testing is the next step the most important is bringing all the values to scale. This entails. The reason why we scale the values is because all distance is affected by the scale of the variables. We want a smaller magnitude otherwise higher magnitude variables will be given higher weightage and will impact the performance.

```
#we need to bring all features to the same level of magnitudes. This can be achieved by a method called feature scaling.
scaler = preprocessing.MinMaxScaler()
scaler.fit(X)
X = pd.DataFrame(scaler.transform(X), index=X.index, columns=X.columns)
X.iloc[4:10]
```

Using K=5 the n-neighbours = 5 and we will fit the KNN model and train it with the training data we have. Following this, train the test data and check the accuracy with the classification report as well.

```
# let's train our test data and check its accuracy.
y_pred = classifier.predict(X_test)
score = accuracy_score(y_test,y_pred)
print('Accuracy :',score)
```

```
Accuracy : 0.8257616247995724
```

```
# let's see the classification report .
y_test_pred = classifier.predict(X_test)
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89     27580
           1       0.70      0.39      0.50      7969

    accuracy                           0.83     35549
   macro avg       0.77      0.67      0.70     35549
weighted avg       0.81      0.83      0.81     35549
```

Trying this again with K = 8 gave an accuracy of 82.1%

Comparing the two results we can see that the accuracy slightly dropped from 5 to 8. This makes the choice of k very critical. A small value of k means that come variables will have a higher influence on the result. A large value makes it computationally expensive and defeats the basic philosophy behind KNN (that points that are near might have similar densities or classes). As both results are practically identical it shows the usage of KNN being practical here but should be used with caution as due to its sensitivity to irrelevant features of the data.

F. XGBoost

XGBoost is an algorithm that has recently been in popular use for structured or tabular data. It is an implementation of gradient boosted decision trees designed for speed and performance. The name XGBoost (Extreme Gradient Boost), refers to the engineering goal to push the limit of computations resources for boosted tree algorithms.

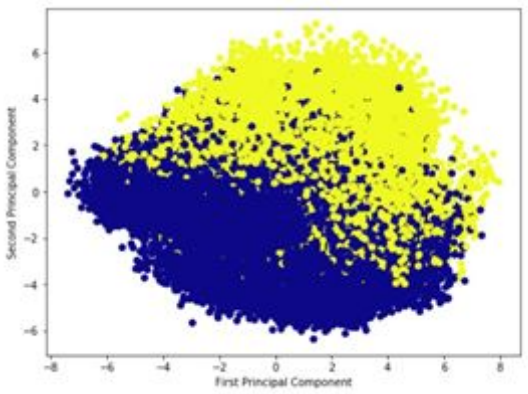
We will be using PCA to reduce the dimension of the dataset. Like before we will be changing all the Yes and No values to 1,0 respectively and dropping Date as it's an insignificant feature. We will also change all categorical features and convert all to dummy variables instead. This is because categorical independent variables (i.e., nominal and ordinal independent variables) cannot be directly entered into a multiple regression. Instead, they need to be converted into dummy variables.

Once scaling the values as with KNN, we can now use PCA however as we don't know if the component value is suitable with 2 or any other number, we will use 2 n-components and check.

```
from sklearn.decomposition import PCA

pca_model = PCA(n_components = 2)
pca = pca_model.fit_transform(scaled)
```

Upon checking the Variance ratio to ensure the dataste doesn't have too many outlier variables we plot the graph for RainTomorrow as a component against another.




```
#Predictions
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print("RMSE: %f" % (rmse))
```

```
RMSE: 0.397395
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f" % (accuracy * 100.0))
```

```
Accuracy: 84.21
```

Once this is done we train the dataset and import the XGBoost implementation, it will now be tested for Root Mean Square Error (RMSE) and Accuracy. The score shows PCA can be used whenever we have a number of dimensions/features and XGBoost is an approach where new models are created that predict the residuals or errors of prior models and then added together make a final prediction. From this the dataset is shown to have high model performance and accuracy.

2. Regression Models

A. Linear Regression

Since the Linear Regression model is only able to manipulate on continuous variables, all of categorical variables were dropped out of the dataset. Furthermore, the entire dataset was standardized to the range from 0 to 1 by "MinMaxScaler" provided by "Sklearn" library. Before the model was fit to the dataset, the correlation coefficient of each feature and target was calculated to check whether the model had the ability of fitting this dataset. As a result, the highest correlation coefficient was only around 0.37. The consequence of giving the model a trial to fit this dataset was the low accuracy score, only approximately 0.21. In conclusion, the model was extremely underfitting, so it is not able to predict tomorrow Australian weather.

B. Logistic Regression

Logistic Regression can be responsible for both numerical and categorical features, none of features would be removed out of the dataset. To avoid unnecessary loss and time-consuming process, the entire dataset was standardized into the range of 0 to 1, and split into the training set and test set in a ratio of 4 to 1. As a result, the accuracy score gained after training the model was nearly 0.85. Besides, the metrics report stated that the model predicted perfectly on class “No” instead of “Yes” since the figure for class “No” was higher “Yes” in both precision and recall.

	precision	recall	f1-score
No	0.87	0.95	0.91
Yes	0.73	0.50	0.59

Briefly, the Logistic Regression performed well in predicting whether it will rain tomorrow in Australia.

3. Neural Network

The neural networks were rather simple in their construction and showed a high quality of results from the testing that was done. The way we created them was by first preparing the data with as many possible variables and then after testing we were able to remove the variables we suspected had no effect and could use this method to prove this. It was clear from the testing that the assumptions with wind direction were justified. The calculations done by this method was one of the most accurate with the best iteration having a weighted average having a value of 1.0 when using ‘riskmm’ and 0.85 without.

	precision	recall	f1-score	support
No	0.86	0.95	0.90	27524
Yes	0.74	0.46	0.57	8025
accuracy			0.84	35549
macro avg	0.80	0.71	0.74	35549
weighted avg	0.83	0.84	0.83	35549

Base(contains wind direction, no risk mm)

	precision	recall	f1-score	support
No	0.87	0.94	0.90	27651
Yes	0.70	0.51	0.59	7898
accuracy			0.84	35549
macro avg	0.79	0.72	0.75	35549
weighted avg	0.83	0.84	0.83	35549

Contains no wind direction or risk mm

	precision	recall	f1-score	support
No	1.00	1.00	1.00	27606
Yes	0.99	1.00	1.00	7943
accuracy			1.00	35549
macro avg	1.00	1.00	1.00	35549
weighted avg	1.00	1.00	1.00	35549

Contains both wind direction and risk mm

Section 3: Conclusions

The models used through the data set all produced a accuracy above 80% except for linear regression which is really helpful and shows that any model can be used for this data set but the best model for this dataset is decision tree as the test accuracy was 0.8534 and training set 0.8561 which are both high and can predict accurately the weather for tomorrow. The worst model used is linear regression as the two methods used produced low accuracy scores and the discrepancy between 2 variables are quite big causing the model to be underfitting and is not suitable for training and predicting weather tomorrow.

Some additional things that would have helped improve our model and help achieve better results is by using multiple methods to tune the hyperparameter such as cross validation, grid search CV to ensure the best value. Another improvement is getting the model to predict data that is not part of the dataset to see if the method works after training and testing.

Overall the use of machine learning models for this dataset provides useful information and accurately solves the problem of predicting rain tomorrow in Australia.

Section 4: References

Hale, J 2019, *Scale, Standardize, or Normalize with Scikit-Learn*, Towards Data Science, Medium, retrieved 16 May 2020, <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02>.

Yiu, T 2019, *Understanding Random Forest*, Towards Data Science, Medium, retrieved 16 May 2020, <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.

Young, J 2017, *Rain in Australia*, Kaggle, retrieved 18 April 2020, <https://www.kaggle.com/jsphyg/weather-dataset-rattle-package>.