COMP 4211 PROJECT REPORT:

Rain in Australia

Taewoo KIM

2021 3873

tkimae@connect.ust.hk

Description of dataset and preprocessing done on it

Dataset chosen:

Rain in Australia from https://www.kaggle.com/kemical/kickstarter-projects

Description:

(Reference: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package)

This dataset contains daily weather observations from numerous Australian weather stations. This dataset contains 24 columns which describes the factors affecting the weather in Australia including:

Date
The date of observation

Location
The common name of the location of the weather station

MinTemp
The minimum temperature in degrees celsius

MaxTemp
The maximum temperature in degrees celsius

Rainfall
The amount of rainfall recorded for the day in mm

Evaporation
The so-called Class A pan evaporation (mm) in the 24 hours to 9am

Sunshine
The number of hours of bright sunshine in the day.

WindGustDir
The direction of the strongest wind gust in the 24 hours to midnight

WindGustSpeed
The speed (km/h) of the strongest wind gust in the 24 hours to midnight

WindDir9am

Direction of the wind at 9am

WindDir3pm
Direction of the wind at 3pm

WindSpeed9am
Wind speed (km/hr) averaged over 10 minutes prior to 9am

WindSpeed3pm
Wind speed (km/hr) averaged over 10 minutes prior to 3pm

Humidity9am
Humidity (percent) at 9am

Humidity3pm
Humidity (percent) at 3pm

Pressure9am
Atmospheric pressure (hpa) reduced to mean sea level at 9am

Pressure3pm
Atmospheric pressure (hpa) reduced to mean sea level at 3pm

Cloud9am
Fraction of sky obscured by cloud at 9am. This is measured in "oktas", which are a unit of eigths. It records how many eigths of the sky are obscured by cloud. A 0 measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.

Cloud3pm
Fraction of sky obscured by cloud (in "oktas": eighths) at 3pm. See Cload9am for a description of the values

Temp9am
Temperature (degrees C) at 9am

Temp3pm
Temperature (degrees C) at 3pm

RainToday
Boolean: 1 if precipitation (mm) in the 24 hours to 9am exceeds 1mm, otherwise 0

RISK_MM

The amount of next day rain in mm. Used to create response variable RainTomorrow. A kind of measure of the "risk".

RainTomorrow
The target variable. Did it rain tomorrow?

The target label **RainTomorrow** predicts whether it will rain tomorrow or not with the answer 'yes' or 'no'. this is done by training a binary classification model and predicting target label **RainTomorrow.**

Preprocessing done on the dataset

To begin with, I dropped "Date" column since I thought it does not affect the result on rain prediction result. Moreover, I dropped "Location" column since the purpose is to predict whether it will rain in Australia, not it will rain in some specific Location in Australia. Furthermore, since it has been mentioned by provider of dataset that "RISK_MM" column should be avoided, I also dropped this column.
Next, I have replaced categorical values to numerical values, so that I could use the dataset for machine learning model. This include replacing values in column "WindGustDir", "WindDir9am", "WindDir3pm" into the following way:

'W:0','WNW:1','WSW:2','NE:3','NNW:4','N:5','NNE:6','SW:7','ENE:8','SSE:9','S:10','NW:11','SE:12','ESE:13', 'E:14','SSW:15'

After this I have also replaced values in "RainToday" and "RainTomorrow" columns in the following way:

'No:0','Yes:1'

Furthermore, I have replaced empty slots in dataset to NaN value and replaced NaN values with the data imputation method in sklearn library called Imputer. I have chosen the 'most_frequent' as a strategy in imputer which selects the most frequent element in the column and replace NaN values with the selected most frequent value.

Lastly, I have standardized the values in the data using the library in sklearn preprocessing called MinMaxScalar(). Purpose of this is to decrease the values in dataset to the range between 0 and 1 so that values get smaller.

Presentation of the first 10 data in dataset after each step is as follows:

After dropping Date, Location, and RISK_MM columns, converting categorical values into integer values, and filling the empty slots with NaN values:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3
0	13.4	22.9	0.6	NaN	NaN	NaN	44.0	NaN	1.0	20.0	...	71.0	2
1	7.4	25.1	NaN	NaN	NaN	1.0	44.0	4.0	2.0	4.0	...	44.0	2
2	12.9	25.7	NaN	NaN	NaN	2.0	46.0	NaN	2.0	19.0	...	38.0	3
3	9.2	28.0	NaN	NaN	NaN	3.0	24.0	12.0	14.0	11.0	...	45.0	1
4	17.5	32.3	1.0	NaN	NaN	NaN	41.0	8.0	11.0	7.0	...	82.0	3
5	14.6	29.7	0.2	NaN	NaN	1.0	56.0	NaN	NaN	19.0	...	55.0	2
6	14.3	25.0	NaN	NaN	NaN	NaN	50.0	7.0	NaN	20.0	...	49.0	1
7	7.7	26.7	NaN	NaN	NaN	NaN	35.0	9.0	NaN	6.0	...	48.0	1
8	9.7	31.9	NaN	NaN	NaN	4.0	80.0	12.0	11.0	7.0	...	42.0	
9	13.1	30.1	1.4	NaN	NaN	NaN	28.0	10.0	9.0	15.0	...	58.0	2

10 rows × 21 columns

After Imputation on the data was done (after NaN values were replaced with most frequent data):

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity3
0	13.4	22.9	0.6	4.0	10.7	12.0	44.0	5.0	1.0	20.0	...	71.0	2
1	7.4	25.1	0.2	4.0	10.7	1.0	44.0	4.0	2.0	4.0	...	44.0	2
2	12.9	25.7	0.2	4.0	10.7	2.0	46.0	5.0	2.0	19.0	...	38.0	3
3	9.2	28.0	0.2	4.0	10.7	3.0	24.0	12.0	14.0	11.0	...	45.0	1
4	17.5	32.3	1.0	4.0	10.7	12.0	41.0	8.0	11.0	7.0	...	82.0	3
5	14.6	29.7	0.2	4.0	10.7	1.0	56.0	5.0	12.0	19.0	...	55.0	2
6	14.3	25.0	0.2	4.0	10.7	12.0	50.0	7.0	12.0	20.0	...	49.0	1
7	7.7	26.7	0.2	4.0	10.7	12.0	35.0	9.0	12.0	6.0	...	48.0	1
8	9.7	31.9	0.2	4.0	10.7	4.0	80.0	12.0	11.0	7.0	...	42.0	
9	13.1	30.1	1.4	4.0	10.7	12.0	28.0	10.0	9.0	15.0	...	58.0	2

10 rows × 21 columns

After standardization of the values in dataset was done(using MinMaxScalar):

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity9am	Humidity
0	0.516509	0.523629	0.001348	0.026915	0.736111	0.785714	0.294574	0.285714	0.000000	0.140625	...	0.707071	0.21
1	0.375000	0.565217	0.000270	0.026915	0.736111	0.000000	0.294574	0.214286	0.071429	0.015625	...	0.434343	0.24
2	0.504717	0.576560	0.000270	0.026915	0.736111	0.071429	0.310078	0.285714	0.071429	0.132812	...	0.373737	0.29
3	0.417453	0.620038	0.000270	0.026915	0.736111	0.142857	0.139535	0.785714	0.928571	0.070312	...	0.444444	0.15
4	0.613208	0.701323	0.002427	0.026915	0.736111	0.785714	0.271318	0.500000	0.714286	0.039062	...	0.818182	0.32
5	0.544811	0.652174	0.000270	0.026915	0.736111	0.000000	0.387597	0.285714	0.785714	0.132812	...	0.545455	0.22
6	0.537736	0.563327	0.000270	0.026915	0.736111	0.785714	0.341085	0.428571	0.785714	0.140625	...	0.484848	0.18
7	0.382075	0.595463	0.000270	0.026915	0.736111	0.785714	0.224806	0.571429	0.785714	0.031250	...	0.474747	0.18
8	0.429245	0.693762	0.000270	0.026915	0.736111	0.214286	0.573643	0.785714	0.714286	0.039062	...	0.414141	0.08
9	0.509434	0.659735	0.003505	0.026915	0.736111	0.785714	0.170543	0.642857	0.571429	0.101562	...	0.575758	0.26

Description of the machine learning task performed on the dataset

List of machine learning task performed on data set:

1. **Logistic regression**
2. **Neural Network**
3. **Support Vector Machine**
4. **Decision Tree**
5. **Random Forest**

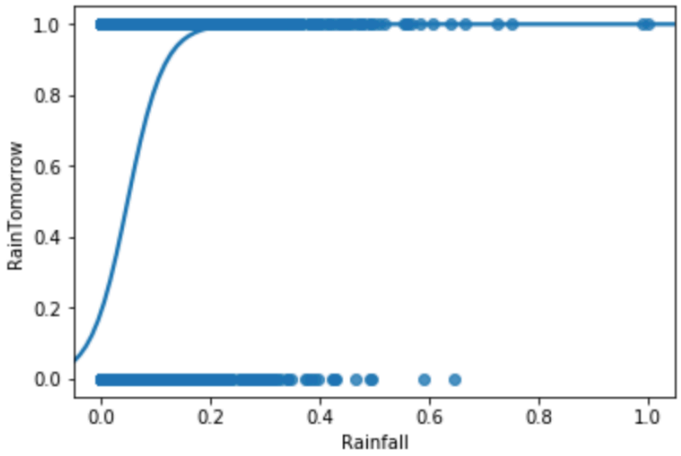
Logistic regression

Logistic regression is a machine learning method which is used to solve the classification problems. In the case of Rain in Australia dataset, we had to solve the binary classification problem, and this is the reason why I chose to use this model.

Logistic function is in the form of sigmoid function which leads the answer to be in the range of 0 and 1. When this model is trained with the train data, it is able to classify the test data set to give the result of either 0 or 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

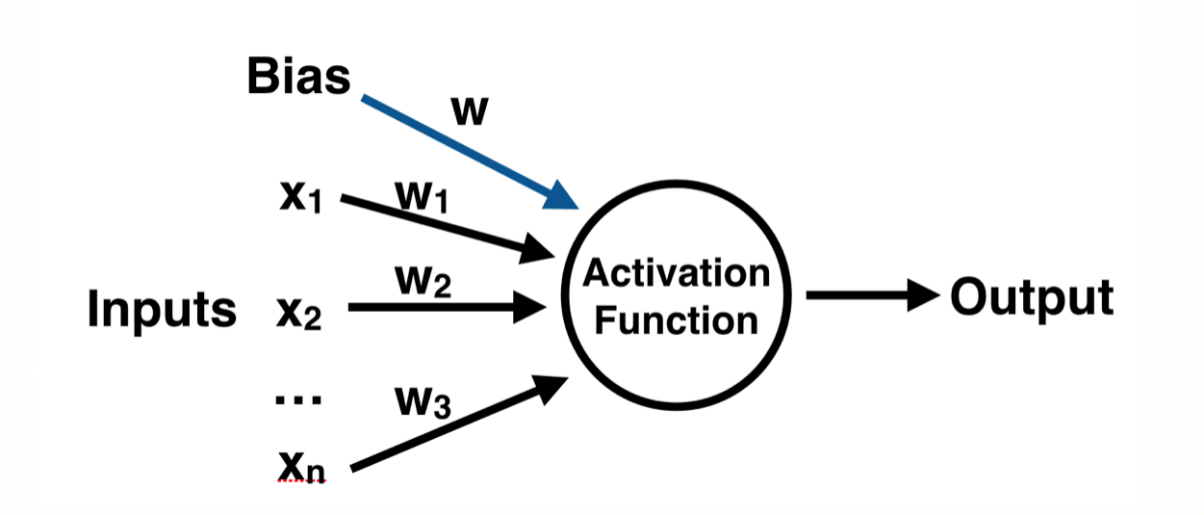
Here is the visualization of logistic regression function with the data column 'Rainfall' as x-axis and 'RainTomorrow' as y-axis for the better understanding:



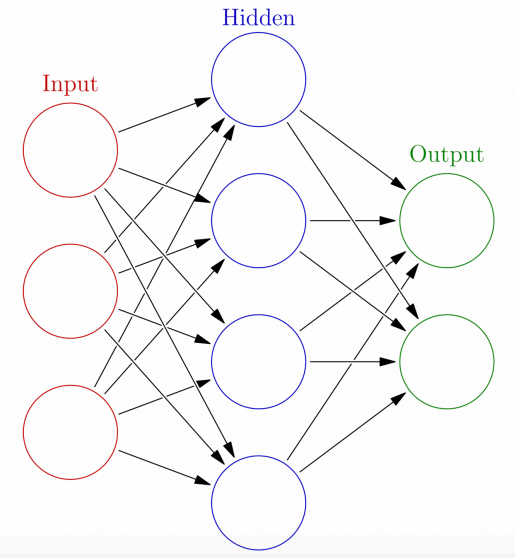
Neural Network

Neural network is a machine learning model which mimics the learning pattern of biological neural network. Neural Network layer consists of multiple perceptron and Neural Network model consists of single or multiple layers. Each perceptron has one or more inputs, a bias, an activation function and a single output.

Picture to aid the understand perceptron from website:
(https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/)



Once there is an output produced by learning from the train data set of weatherAUS.csv, it is compared with the label value in dataset and adjust the weights repeatedly until it reaches maximum number of iterations or an acceptable error rate.
Picture of the neural network model to aid the understanding from website: (https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/)



Support Vector Machine

Support Vector Machine classifies data with the hyperplanes that have been calculated and created by algorithm with train data. When creating hyperplane, margin of a separating hyperplane should be maximized.

Formula to calculate margin is:

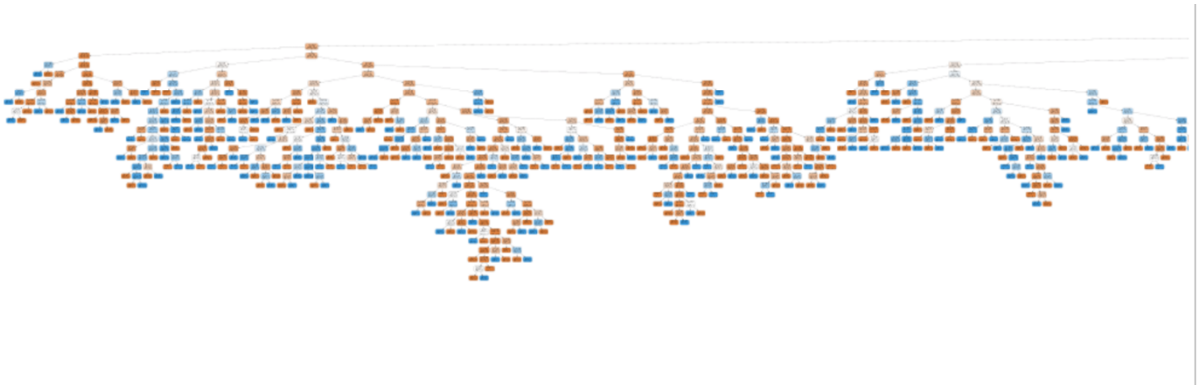
$$\gamma = \frac{1}{2}\frac{\mathbf{w}^T(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}.$$

and we have to minimize the value of ‖w‖ in order to maximize margin. Hence hyperplane is created with the algorithm when training with weatherAus.csv data, and this hyperplane is used to classify the test data set.

Decision Tree

Decision tree is somewhat like rule based system, since the algorithm creates set of rules based on the weatherAus' training dataset with label values. It is a recursive tree construction procedure with divide and conquer method. Rule is created when training with the train dataset, and later when testing, this rule is used to classify whether it will rain tomorrow or not.

Below is a screen shot of a part of the tree produced with train dataset from weatherAus.



Random Forest

This algorithm is also suitable for the dataset weatherAUS since it works well for the classification problem. It is like creating number of decision trees and make a forest. Higher the number of trees, the higher the accuracy result. Hence explanation of how it works is the same as the explanantion on the decision tree except that it is a collection of decision trees.

Below is a visualization of the part of random forest created with the weatherAUS data. From this we can see how rules are set and how it is used to classify results

Description of the computing environment, machine learning methods and parameter settings.

Computing environment:

Computer model: MacBook Pro(13-inch, 2017)
OS: macOS Mojave ver 10.14
Processor: 2.3 GHz Intel Core i5
Memory: 8GB 2133 MHz LPDDR3
Graphics: Intel Iris Plus Graphics 640 1536 MB
Program used: Jupyter notebook
Programming language: Python

Logistic regression

Logistic regression is used by employing the method linear_model.SGDClassifier() from library called sklearn. I chose this function since it employs the stochastic gradient descent learning. Moreover, it sums over a mini-batch of training example at a time.

Firstly, I have done hyperparameter selection using GridSearchCV on sklearn library. This executes cross-validation by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%.

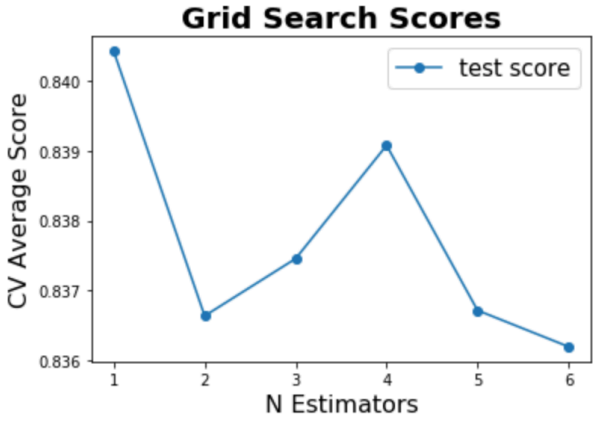
Hyperparameters to be tested are as below:

Penalty: 'l1' , 'l2'

Eta0 = 0.01, 0.05, 0.1

Among these, best hyperparameter chosen is as below:

Best parameter is:
{'eta0': 0.01, 'penalty': 'l1'}
Setting 0 : {'eta0': 0.01, 'penalty': 'l1'} test score: 0.84
Setting 1 : {'eta0': 0.01, 'penalty': 'l2'} test score: 0.837
Setting 2 : {'eta0': 0.05, 'penalty': 'l1'} test score: 0.837
Setting 3 : {'eta0': 0.05, 'penalty': 'l2'} test score: 0.839
Setting 4 : {'eta0': 0.1, 'penalty': 'l1'} test score: 0.837
Setting 5 : {'eta0': 0.1, 'penalty': 'l2'} test score: 0.836



Visual aid has been added to help visualizing the result.

With the chosen hyperparameter, model was fully trained with the train data with the parameter setting as below:

```
Model Setting:
{'alpha': 0.0001, 'average': False, 'class_weight': None,
'early_stopping': False, 'epsilon': 0.1, 'eta0': 0.01, 'fit_intercept':
True, 'l1_ratio': 0.15, 'learning_rate': 'optimal', 'loss': 'log',
'max_iter': 50, 'n_iter': None, 'n_iter_no_change': 5, 'n_jobs': None,
'penalty': 'l1', 'power_t': 0.5, 'random_state': None, 'shuffle': True,
'tol': 0.001, 'validation_fraction': 0.1, 'verbose': 1, 'warm_start':
False}
```

I have used the default settings other than the highlighted parameters. early stopping has been set to False to disable early stopping, eta0 has been set to 0.01 since it turned out to be optimal during hyperparameter selection, loss has been set to 'log' because it gives logistic regression a probabilistic classifier. max_iter is set to set the maximum number of iterations as 50 and penalty is 'l1' since it turned out to be the optimal one during hyperparameter selection.

Lastly, verbose has been set to 1 to enable verbosity of function.

Neural Network

Neural network model is built by using the method MLPClassifier in the sklearn library. This model adopts the gradient descent optimization algorithm.

Firstly, I have done hyperparameter selection using GridSearchCV on sklearn library. This executes cross-validation by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%.

Hyperparameters to be tested are as below:

```
#10 hyperparameters and going to choose best one out of them
tuned_parameters = [{'hidden_layer_sizes':[(1, )], 'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(2, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(3, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(4, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(5, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(6, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(7, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(8, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(9, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]},
                    {'hidden_layer_sizes':[(10, )],'learning_rate':['constant'], 'learning_rate_init':[0.01]}]
```

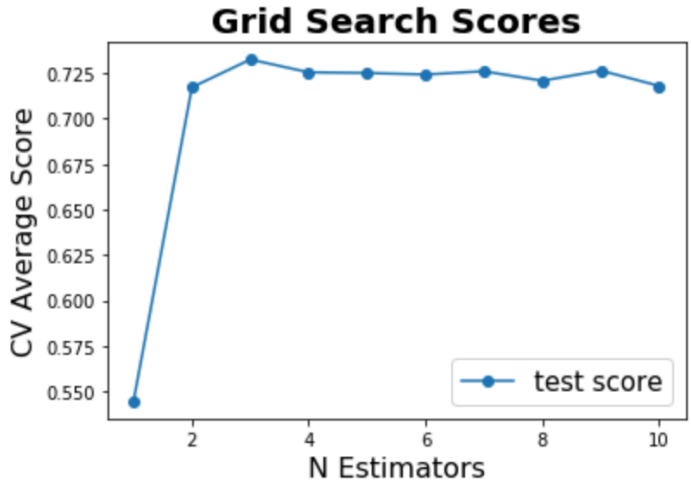
Among these, best hyperparameter chosen is as below:

```
Best parameters set found on development set:

{'hidden_layer_sizes': (3,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}

Grid scores on development set:

0.544 (+/-0.178) for {'hidden_layer_sizes': (1,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.717 (+/-0.014) for {'hidden_layer_sizes': (2,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.733 (+/-0.009) for {'hidden_layer_sizes': (3,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.725 (+/-0.008) for {'hidden_layer_sizes': (4,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.725 (+/-0.010) for {'hidden_layer_sizes': (5,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.724 (+/-0.023) for {'hidden_layer_sizes': (6,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.726 (+/-0.003) for {'hidden_layer_sizes': (7,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.721 (+/-0.014) for {'hidden_layer_sizes': (8,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.726 (+/-0.010) for {'hidden_layer_sizes': (9,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
0.718 (+/-0.021) for {'hidden_layer_sizes': (10,), 'learning_rate': 'constant', 'learning_rate_init': 0.01}
```



Visual aid has been added to help visualizing the result.

With the chosen hyperparameter, model was fully trained with the train data with the parameter setting as below:

```
Parameter setting for MLPClassifier

{'activation': 'relu', 'alpha': 0.0001, 'batch_size': 'auto', 'beta_1':
0.9, 'beta_2': 0.999, 'early_stopping': True, 'epsilon': 1e-08,
'hidden_layer_sizes': (3,), 'learning_rate': 'constant',
'learning_rate_init': 0.001, 'max_iter': 300, 'momentum': 0.9,
'n_iter_no_change': 10, 'nesterovs_momentum': True, 'power_t': 0.5,
'random_state': 1, 'shuffle': True, 'solver': 'sgd', 'tol': 0.0001,
'validation_fraction': 0.1, 'verbose': True, 'warm_start': False}
```

Highlighted parameters are the parameters that I have changed from the default setting. Other than those, parameters are default parameters. Early stopping has been set to True to terminate training when validation score is not improving and prevent overfitting of the model. Hidden layer size is set to (3,0) which was proven to be the best during cross validation procedure. Max_iter has been set to 300 to limit the number of epoch, and random_state has been set to 1 to be used as a seed for random number generator. I have chosen sgd as a solver which employs stochastic gradient descent algorithm and verbose has been set to true to enable verbosity.

Support Vector Machine

Support vector machine is employed using SVC in the sklearn.svm library.

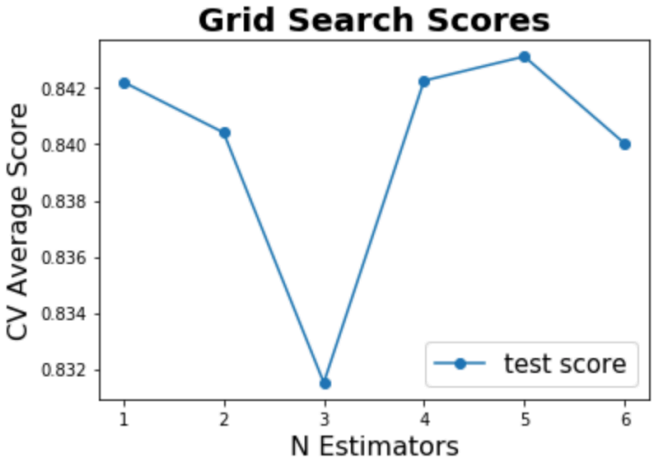
Firstly, I have done hyperparameter selection using GridSearchCV on sklearn library. This executes cross-validation by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%.

Hyperparameters to be tested are as below:

```
Setting 0 :  {'C': 1, 'kernel': 'linear'} test score: 0.842
Setting 1 :  {'C': 1, 'kernel': 'rbf'} test score: 0.84
Setting 2 :  {'C': 1, 'kernel': 'poly'} test score: 0.832
Setting 3 :  {'C': 10, 'kernel': 'linear'} test score: 0.842
Setting 4 :  {'C': 10, 'kernel': 'rbf'} test score: 0.843
Setting 5 :  {'C': 10, 'kernel': 'poly'} test score: 0.84
```

Visualization is done to see which one is the best

```
Best parameter is:
{'C': 10, 'kernel': 'rbf'}
```



With the chosen hyperparameter, model was fully trained with the train data with the parameter setting as below:

```
parameter setting is:
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

Highlighted parameters are the parameters that I have changed from the default setting. Other than those, parameters are default parameters. Penalty parameter C for the error term has been set as 10 and kernel type has been set to 'rbf' which is equivalent to the default setting.

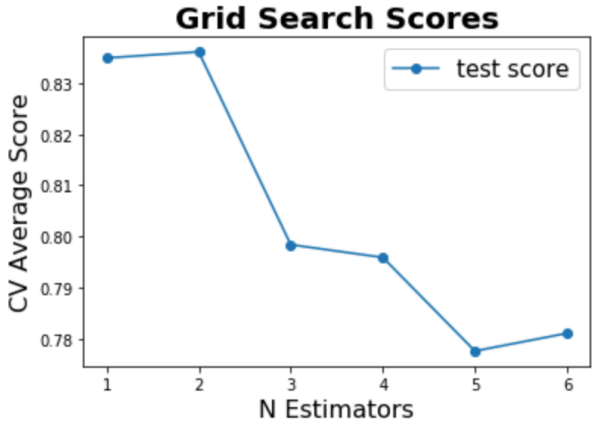
Decision Tree

Decision tree model was built using DecisionTreeClassifier() in sklearn.tree method.

First, hyperparameter tuning was done with GridSearchCV on sklearn library. This executes cross-validation by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%.

Parameters to be tested are as below:

```
Best parameter is:
{'max_depth': 10, 'splitter': 'best'}
Setting 0 :  {'max_depth': 10, 'splitter': 'random'} test score: 0.835
Setting 1 :  {'max_depth': 10, 'splitter': 'best'} test score: 0.836
Setting 2 :  {'max_depth': 20, 'splitter': 'random'} test score: 0.798
Setting 3 :  {'max_depth': 20, 'splitter': 'best'} test score: 0.796
Setting 4 :  {'max_depth': 30, 'splitter': 'random'} test score: 0.778
Setting 5 :  {'max_depth': 30, 'splitter': 'best'} test score: 0.781
```



As you can see from the graph, best parameter set is setting 0 with max_depth = 10 and splitter = 'best'. With this parameter, model was fully trained with train data:

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=None,
            splitter='best')
```

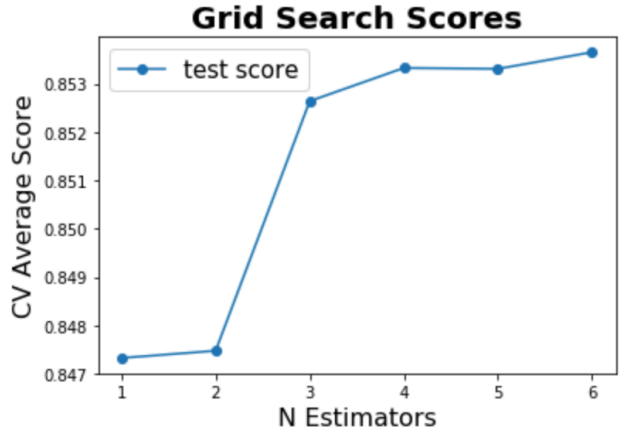
Other than tuned hyperparameters, all parameters are set as default parameters

Random Forest

Random forest was built by using RandomForestClassifier in sklearn.ensemble library

First, hyperparameter selection is performed with the parameter sets below using GridSearchCV in sklearn library, and the best parameter set is chosen. This executes cross-validation by randomly sampling 80% of the training instances to train a classifier and then validating it on the remaining 20%.

```
Best parameter is:
{'max_depth': 30, 'n_estimators': 400}
Setting 0 :  {'max_depth': 10, 'n_estimators': 200} test score: 0.847
Setting 1 :  {'max_depth': 10, 'n_estimators': 400} test score: 0.847
Setting 2 :  {'max_depth': 20, 'n_estimators': 200} test score: 0.853
Setting 3 :  {'max_depth': 20, 'n_estimators': 400} test score: 0.853
Setting 4 :  {'max_depth': 30, 'n_estimators': 200} test score: 0.853
Setting 5 :  {'max_depth': 30, 'n_estimators': 400} test score: 0.854
```



As you can see from the graph and output, third set is the best parameter set and with these parameters, model is fully trained with train data. Below is the parameter setting for this model:

```
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
            max_depth=30, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=400, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Other than highlighted parameters, parameters are set as default parameters and highlighted parameters have been set with the best parameter selected by hyperparameter tuning.

Ensemble method

Ensemble method is an algorithm which uses multiple machine learning algorithm to derive a better model than the model that could obtained from a single algorithm. In some cases, even in my experiment, performance (in mean accuracy score) can be lower than the single model, but it gives a model a power of generalization. If this model is limited to the dataset provided, it may perform worse, but if more amount of data is given, there is high possibility that it will perform better.

In my experiment, I have used the VotingClassifier from sklearn.ensemble library. Parameter setting is displayed below. In this classifier, each model makes prediction for each data point, and each prediction is considered as a vote. For example, suppose there are three models: model1, model2, and model3, and each has a prediction value as 0,1,0 accordingly. Then VotingClassifier with max voting strategy(voting='hard') predicts the majority as a value which is 0 in this case.

```
VotingClassifier(estimators=[('lr', SGDClassifier(alpha=0.0001,
average=False, class_weight=None,
      early_stopping=False, epsilon=0.1, eta0=0.01, fit_intercept=True,
      l1_ratio=0.15, learning_rate='optimal', loss='log', max_iter=50,
      n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l1',
      ...obs=None,
          oob_score=False, random_state=None, verbose=0,
          warm_start=False))],
        flatten_transform=None, n_jobs=None, voting='hard',
weights=None)
```

Parameters for this classifier is set as defaults, except for the estimators where I have put classifier objects of five previously defined models I have experimented so far, and voting parameter is set as 'hard' in order to perform Max Voting.

Description of the experiments and Visualization of the results

Before fitting the train data into the model, preprocessed dataset was divided into train and test data using train_test_split in library sklearn.model_selection. Train data has been set to 80% and test data has been set to 20% of whole data.

Logistic regression:

With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

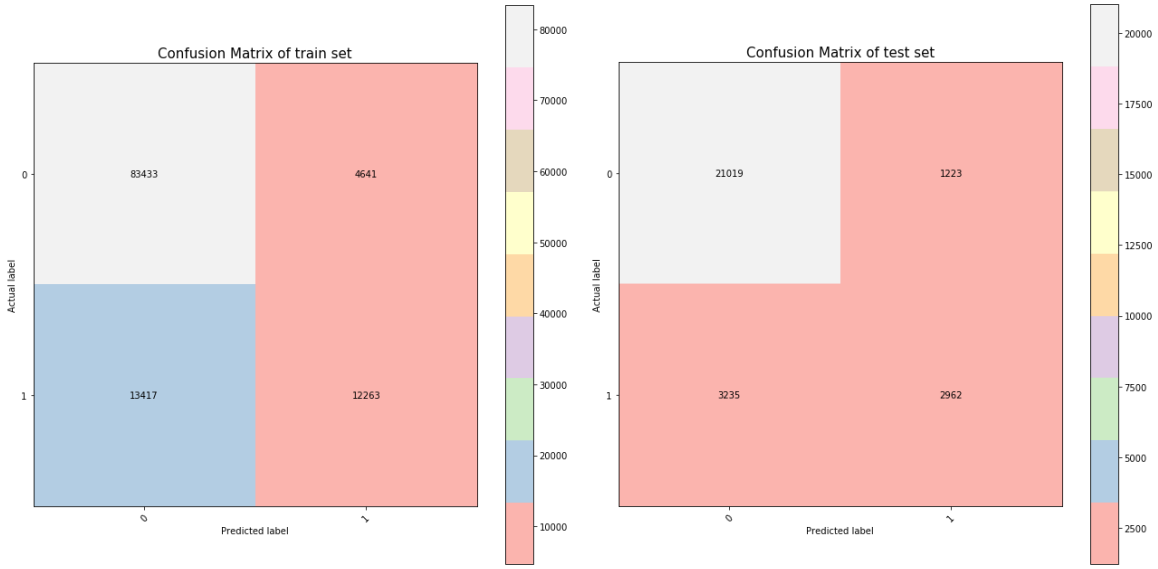
```
Mean Accuracy of training set is 0.8412539339275982 and test set is
0.8432434333134077

time taken is:
0.6145811080932617
```

Mean accuracy is calculated by adding up the number of correctly predicted sample divided by the total number of data.

$$accuracy\left(y, \hat{y}\right) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\left(\hat{y} = y_i\right).$$

Mean accuracy on train data is `0.8412539339275982` and on test data is `0.8432434333134077`. since there is not a big difference in the value, there is no over fitting of train data happening and since score is quite close to full score 1.0, we can say the performance of this model is moderate. Below is the visualization of the classification done by model on train and test data.



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification

Neural Network:

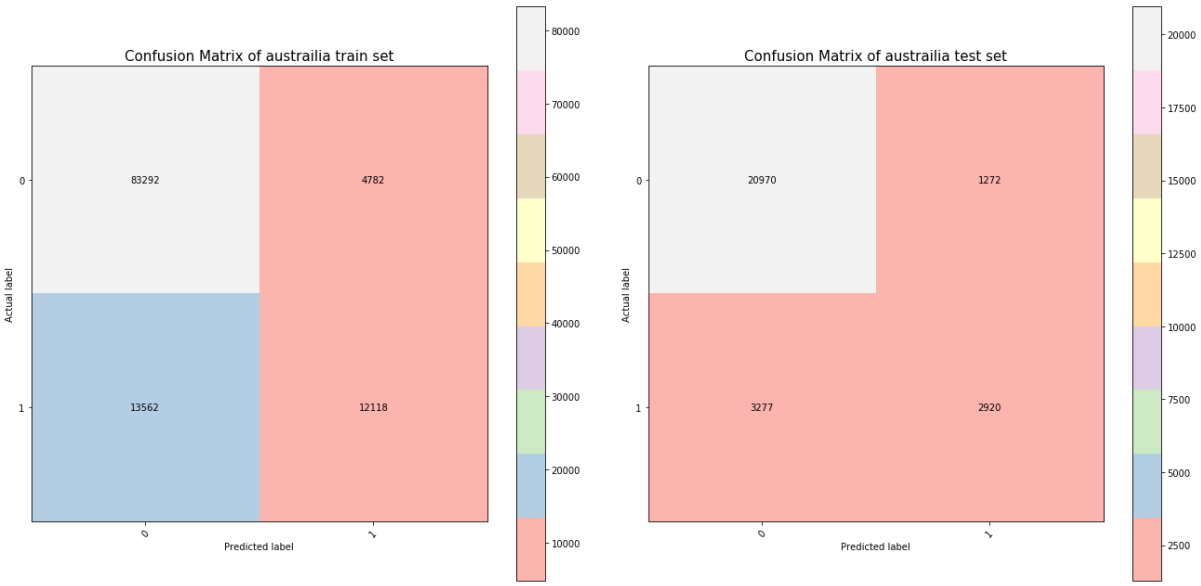
With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

```
Mean accuracy score of austrailia data training set is
0.8387397366246462 and test set is 0.8400436020957136

time taken is:
12.768558025360107
```

Mean accuracy score is calculated in the same way explained in the explanation of logistic regression.

Mean accuracy score on train data set is 0.8387397366246462 and test set is 0.8400436020957136. since there is no big difference in the score, I could say there is no overfitting or underfitting on the model, and the score is quite close to the perfect score which is 1, I could say the performance of this model is descent.



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification.

SVM:

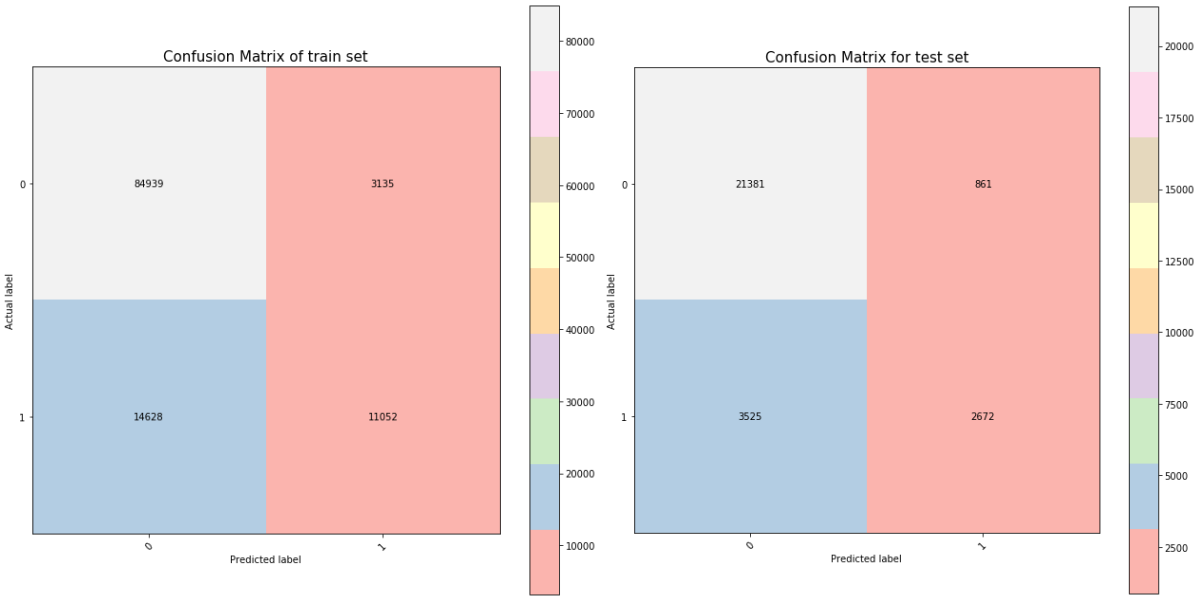
With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

```
mean accuracy score for train set of SVM model is:
0.8438472493274962
mean accuracy score for test set of SVM model is:
0.8457751679032315

Time taken :
1560.6207661628723
```

Mean accuracy score is calculated in the same way explained in the explanation of logistic regression.

Mean accuracy score on train data set is 0.8438472493274962 and test set is 0.8457751679032315. Since there is no big difference in the score, I could say there is no overfitting or underfitting on the model, and the score is quite close to the perfect score which is 1.0, I could say the performance of this model is descent.



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification.

Decision Tree:

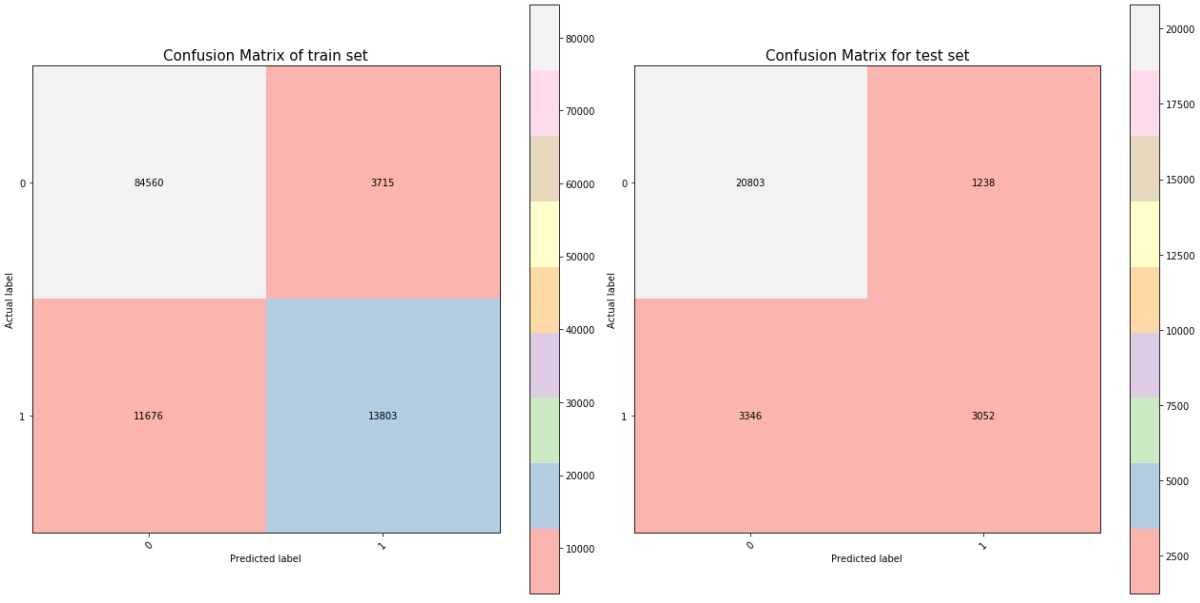
With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

```
mean accuracy score for train set of SVM model is:
0.8646992633226085
mean accuracy score for test set of SVM model is:
0.8388128977812159

Time taken :
1.0231678485870361
```

Mean accuracy score is calculated in the same way explained in the explanation of logistic regression.

Mean accuracy score on train data set is 0.8646992633226085 and test set is 0.8388128977812159. Since there is no big difference in the score, I could say there is no overfitting or underfitting on the model, and the score is quite close to the perfect score which is 1.0, I could say the performance of this model is descent.



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this confusion matrix help us to visualize the result of classification.

Random Forest:

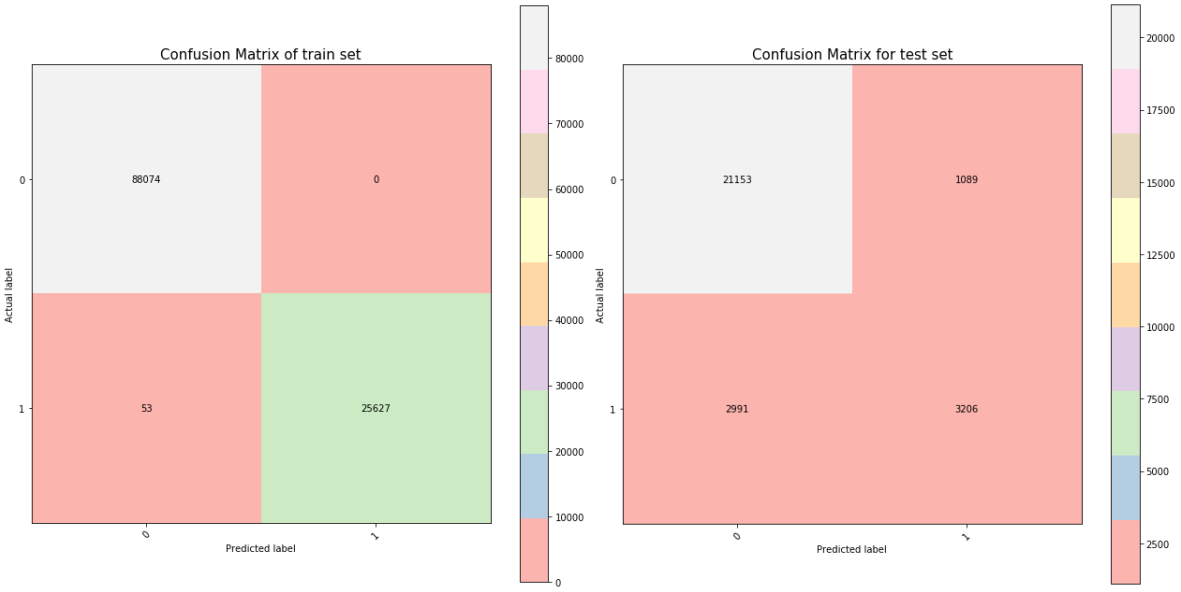
With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

```
mean accuracy score for train set of Random Forest model is:
0.9995340823179845
mean accuracy score for test set of Random Forest model is:
0.8565350399099828


Time taken :
413.93043303489685
```

Mean accuracy score is calculated in the same way explained in the explanation of logistic regression.

Mean accuracy score on train data set is 0.9995340823179845 and test set is 0.8565350399099828. Since there is quite a big difference in the score, where mean accuracy score on train set is way higher than the mean accuracy score in the test data set, I could say there is overfitting on the model. However, the mean accuracy score on the test set is quite close to 1.0, so the performance of model is descent.



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification.

Ensemble method:

Feature selection

In order to reduce the dimensionality of the data for the faster processing speed, I have implemented the feature selection algorithm which is SelectKBest method in sklearn.feature_selection library.

Parameter setting for this selector is listed below.

```
SelectKBest(k=5, score_func=<function chi2 at 0x1a1beff510>)
```

K is set as five to display 5 best parameters and score_func is set as chi2 from sklearn.feature_selection library. Selected top five features from this method is listed below:

```
Index(['Rainfall', 'Sunshine', 'Humidity3pm', 'Cloud3pm', 'RainToday'],
dtype='object')
```

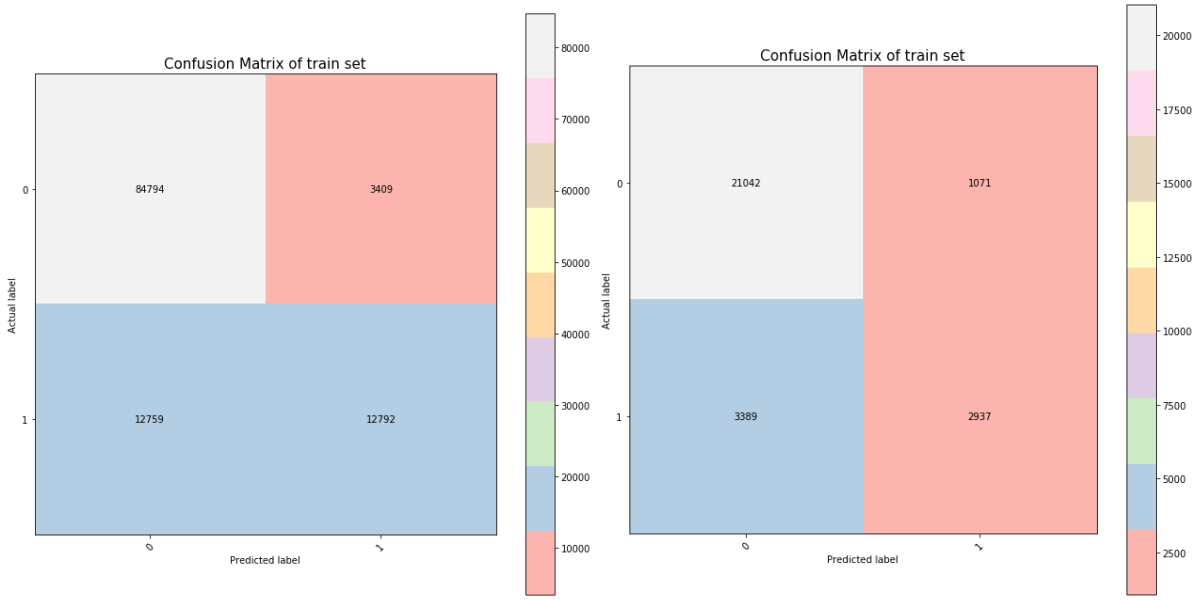
Max Voting with full data

With randomly divided train and test data, model was trained with the parameter settings mentioned in the above section with the help of fit() function. After this, mean accuracy score on train and test set were obtained using the score() function. Scores calculated on train, test data, and time taken(in seconds) is displayed below:

```
mean accuracy score of Max voting ensemble method on train set is
0.8578687342862669
mean accuracy score of Max voting ensemble method on test set is
0.8431731073525792
confusion matrix is:
time taken for this method is:
715.0562710762024
```

Mean accuracy score is calculated in the same way explained in the explanation of logistic regression.

Mean accuracy score on train data set is 0.8578687342862669 and test set is 0.8431731073525792. Since there is no big difference in the score, I could say there is no overfitting or underfitting on the model. Since the mean accuracy score on the test set is quite close to 1.0, I can say the performance of model is descent.



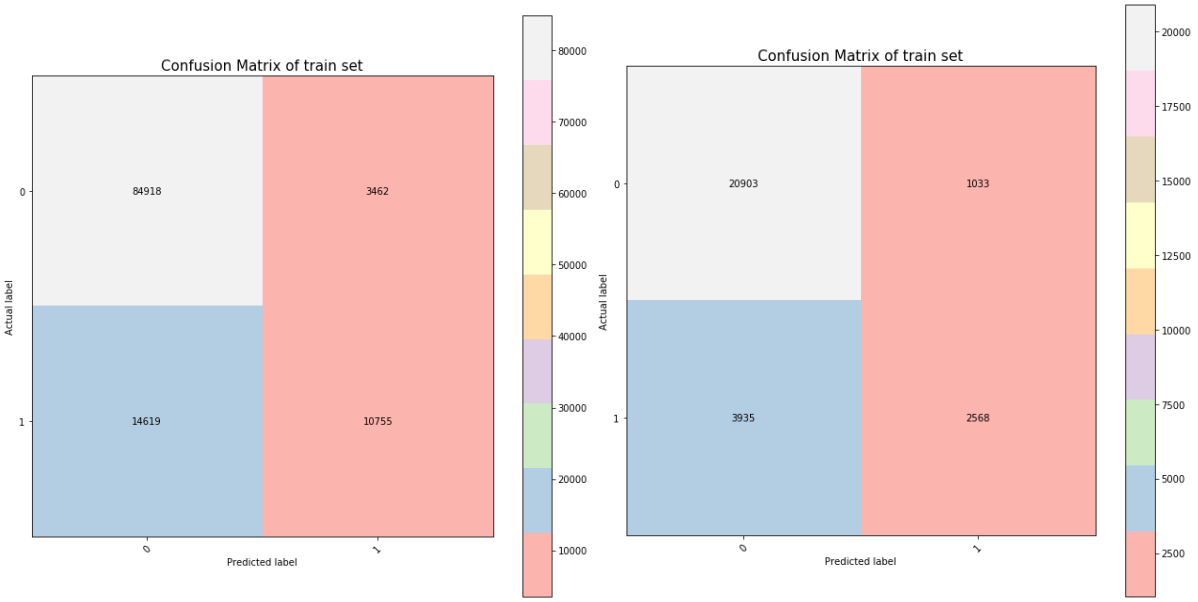
X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification.

Max Voting with partial data

train_X and test_X data's dimensionality is reduced into the feature selected by SelectKBest, and the same experiment as above was performed. Calculated mean accuracy scores and time taken is displayed below

```
mean accuracy score of Max voting ensemble method on train set is
0.8410517432354027
mean accuracy score of Max voting ensemble method on test set is
0.8253103133021555
confusion matrix is:
time taken for this method is:
535.9432029724121
```

Mean accuracy score on train data set is 0.8410517432354027 and test set is 0.8253103133021555. Since there is no big difference in the score, I could say there is no overfitting or underfitting on the model. Since the mean accuracy score on the test set is quite close to 1.0, I can say the performance of model is descent



X axis of These confusion matrices denote predicted label, and y axis denotes actual label. So number written on the top left square shows number of correctly classified data with labels 0 and number written on the bottom right square denotes number of correctly classified data with labels 1. Like this, confusion matrix help us to visualize the result of classification

Comparison of the results obtained

1. Comparison between classifiers

- **Performance wise(with mean accuracy score on test data)**

  I will list the names of classifiers with the mean accuracy score in the descending order of mean accuracy score.

  1.Random Forest - 0.8565350399099828
  2.SVM - 0.8457751679032315
  3.Logistic Regression - 0.8432434333134077
  4.Voting classifier(with full data) - 0.8431731073525792
  5. Neural Network - 0.8400436020957136
  6.Decision Tree - 0.8388128977812159

- **Timewise(with time taken)**

  Here I will list the names of classifiers with the mean accuracy score in the ascending order of time taken.

  1.Logistic regression - 0.6145811080932617
  2.Decision Tree - 1.0231678485870361
  3.Neural network - 12.768558025360107
  4.Random Forest - 413.93043303489685
  5.Voting classifier(with full data) - 715.0562710762024
  6.SVM - 1560.6207661628723

- **Overall**

  In terms of the mean accuracy score, we can say Random Forest is the best model since it had a best predicting performance on the test data set. However, if we see the time it took, it is around 400 seconds and it is 4th fastest model among six models. In terms of time taken, logistic regression model is the best model since it took the shortest time for training, and testing. If we see the performance ranking of logistic regression, it is the 3rd best among six models. Hence, among six models, we can say logistic regression is the best model, considering the performance and the time taken together. However, if we have better resources to use, time taken will be reduced to the great extent and Random Forest can be the better choice in that case. Moreover, since Max Voting classifier which uses ensemble method is created out of rest of the five models, so this model could perform better in the bigger dataset.

2. Comparison between ensemble classifier trained with full data versus partial data

- **Performance wise(with mean accuracy score)**

  I will list the names of classifiers with the mean accuracy score in the descending order of mean accuracy score.

  1.Voting classifier(with full dataset) - 0.8431731073525792
  2.Voting classifier(with partial dataset) - 0.8253103133021555

- **Timewise(with time taken)**

  Here I will list the names of classifiers with the mean accuracy score in the ascending order of time taken.

  1.Voting classifier(with partial dataset) - 535.9432029724121
  2.Voting classifier(with full dataset) - 715.0562710762024

- **Overall**

  We can see by reducing the features to 5 can dramatically reduce the time taken (by about 180 seconds), and the difference between the mean accuracy score is not huge. Hence, if one looks for better accuracy, it is recommended to take the full dataset to train, but efficiency wise, it is better to take only partial features.