%% -- Data Importing

clc, clear all, close all, warning off

% Loading the data from local file

% reference - https://uk.mathworks.com/help/matlab/ref/readtable.html

data = readtable('seattle-weather.csv');

% Displaying the first few rows of the dataset

disp(['First couple rows of the data:']);

head(data);

% Displaying the last few rows of the dataset

disp(['Last couple rows of the data:']);

tail(data);

% Displaying the column names

disp(['Data column names']);

disp(data.Properties.VariableNames);

%Displaying number of rows and columns

disp(['Number of Rows and Columns:']);

disp(size(data));

%Displaying the timeframe of data

disp(['The dataset is from:']);

disp(min(data.date));

disp(['Till:']);

disp(max(data.date));

%This Dataset is a weather dataset of Seattle, USA. which contains.....

%'precipitation','temp\_max','temp\_min','wind','weather' values and...

%types for Seattle from 01/01/2012 - 31/12/2015 everyday

%%

%% -- Data Processing

%General Data Statistics

%Displaying summary statistics of dataset

disp('Summary Statistics:' )

summary(data);

%4 numeric weather type columns

%1 timeframe column (date)

%1 categorical column (weather)

%Reference: https://uk.mathworks.com/help/matlab/ref/std.html

%Computing Standard deviations

disp('Standard dev of percipitation column:' )

disp(std(data.precipitation));

disp('Standard dev of temp\_max column:' )

disp(std(data.temp\_max));

disp('Standard dev of temp\_min column:' )

disp(std(data.temp\_min));

disp('Standard dev of wind speed column:' )

disp(std(data.wind));

%These were the stds

%Percipitation: 6.6802

%temp\_max: 7.3498

%temp\_min:5.0230

%wind speed 1.4378

%From this we can observe wind column has the lowest std = low variability

%temp\_max has the highest

%standardization should reduce this variability

%Checking for duplicates through the date column

%Reference: https://uk.mathworks.com/matlabcentral/answers/688889-how-to-convert-a-column-in-a-table-to-date-format-for-plotting-a-time-series

data.date = datetime(data.date, 'InputFormat', 'yyyy-MM-dd');

%Reference: https://uk.mathworks.com/matlabcentral/answers/19042-finding-duplicate-values-per-column

uniqueDate = unique(data.date);

[countOfDate] = histcounts(data.date, uniqueDate);

indexToRepeatedValue = (countOfDate ~= 1);

repeatedValues = uniqueDate(indexToRepeatedValue);

numberOfAppearancesOfRepeatedValues = countOfDate(indexToRepeatedValue);

%Theres no duplicates, and the dataset is a continuous time series

% Create a copy of the original data

%This is where we will alter the data

data\_copy = readtable('seattle-weather.csv');

%Reference: https://blogs.mathworks.com/student-lounge/2023/01/11/weather-forecasting-in-matlab-for-wids-datathon-2023/

%Extract Day, month, year to make compatible for ml algorithms

data\_copy.Day = data\_copy.date.Day;

data\_copy.Month = data\_copy.date.Month;

data\_copy.Year = data\_copy.date.Year;

%Removing date column

data\_copy.date = [];

%Reference: https://uk.mathworks.com/help/matlab/ref/table.movevars.html

%Moving weather column to the end

data\_copy = movevars(data\_copy, "weather", "After", "Year");

head(data\_copy)

% Checking for missing values

disp(['Missing Values Count for Each Variable:']);

disp(sum(ismissing(data)));

%Since there is no missing values we dont need to replace/remove anything

%Displaying all the different weather types in the dataset

unique\_weather = unique(data\_copy.weather);

disp('These are the different weathers')

disp(unique\_weather);

%The unique weather types are

%drizzle

%fog

%snow

%rain

%sun

%%

% Asigning all unique weather types to a number

%since there are 5 weather types i'm assinging all from 1-5

% Reference: https://www.mathworks.com/help/matlab/ref/containers.map.html

weather\_mapping = containers.Map(unique\_weather, [1, 2, 3, 4, 5]);

% Assign numeric labels to the 'weather\_labels' column

% Reference: https://www.mathworks.com/help/matlab/ref/cell2mat.html

data\_copy.weather\_labels = cell2mat(values(weather\_mapping, data\_copy.weather));

%removing weather column as we dont want categorical data

data\_copy.weather= []

%Creating a column called temp\_range which calculates temp range everyday

data\_copy.temp\_range = data\_copy.temp\_max - data\_copy.temp\_min;

% Display the updated data\_copy table

disp(['Updated data\_copy with temp\_range and weather)labels:']);

% Check for zero values in numeric columns of data\_copy

% Reference : https://uk.mathworks.com/matlabcentral/answers/838378-how-to-delect-the-zero-values-in-table

disp("Columns with Zero Values in data:");

% Display columns with zero values in data\_copy

disp(sum(data\_copy{:, {'precipitation', 'temp\_max', 'temp\_min', 'wind','Day','Month','Year','weather\_labels','temp\_range'}} == 0));

%there is 838 zero values in percipitation column

%2 temp\_max column

%16 temp\_min column

%zero values dont need to be removed as its a weather dataset

% Create binary columns for each season to improve model

%Reference: https://uk.mathworks.com/help/matlab/ref/double.ismember.html

%Winter is assigned to months 1,2,3,12

data\_copy.Winter = double(ismember(data\_copy.Month, [1, 2, 3, 12]));

%Summer is assigned to months 6,7,8

data\_copy.Summer = double(ismember(data\_copy.Month, [6, 7, 8]));

%Autumn is assigned to months 9,10,11

data\_copy.Autumn = double(ismember(data\_copy.Month, [9, 10, 11]));

%Spring is assigned to months 4,5

data\_copy.Spring = double(ismember(data\_copy.Month, [4, 5]));

%information obtained : https://www.timeanddate.com/calendar/seasons.html?n=234

%reference: https://uk.mathworks.com/help/matlab/ref/table.movevars.html

%moving the target column to the end of the table

data\_copy = movevars(data\_copy, "weather\_labels", "After", "temp\_range");

% Display the updated data\_copy table

disp(['Updated data\_copy with standarised columsns, temp\_range column,weather)labels, and additional binary season columns:']);

head(data\_copy)

%%

%%Standardizing the weather predictor columns

% Reference: https://github.com/vighnesh32/Machine-Learning-Project/blob/main/diabholdout.m

% Standardization of precipitation column

mean\_precipitation = mean(data\_copy.precipitation);

std\_precipitation = std(data\_copy.precipitation);

stan\_precipitation = (data\_copy.precipitation - mean\_precipitation) / std\_precipitation;

data\_copy.precipitation = stan\_precipitation;

% Reference: https://github.com/vighnesh32/Machine-Learning-Project/blob/main/diabholdout.m

% Standardization of temp\_max column

mean\_temp\_max = mean(data\_copy.temp\_max);

std\_temp\_max = std(data\_copy.temp\_max);

stan\_temp\_max = (data\_copy.temp\_max - mean\_temp\_max) / std\_temp\_max;

data\_copy.temp\_max = stan\_temp\_max;

% Reference: https://github.com/vighnesh32/Machine-Learning-Project/blob/main/diabholdout.m

% Standardization of temp\_min column

mean\_temp\_min = mean(data\_copy.temp\_min);

std\_temp\_min = std(data\_copy.temp\_min);

stan\_temp\_min = (data\_copy.temp\_min - mean\_temp\_min) / std\_temp\_min;

data\_copy.temp\_min = stan\_temp\_min;

% Reference: https://github.com/vighnesh32/Machine-Learning-Project/blob/main/diabholdout.m

% Standardization of wind column

mean\_wind = mean(data\_copy.wind);

std\_wind = std(data\_copy.wind);

stan\_wind = (data\_copy.wind - mean\_wind) / std\_wind;

data\_copy.wind = stan\_wind;

% Displaying the first few rows of the updated dataset

head(data\_copy);

%% -- Data Visualization

%Reference: https://uk.mathworks.com/videos/how-to-make-subplots-in-matlab-using-tiledlayout-1599239984171.html

%Plotting scatter plots for all variables

tiledlayout('flow')

% Precipitation vs Temperature Min

nexttile

scatter(data\_copy.precipitation, data\_copy.temp\_min)

xlabel('Precipitation')

ylabel('Temperature Min')

title('Precipitation vs Temperature Min')

%Precipitation vs Temperature Max

nexttile

scatter(data\_copy.precipitation, data\_copy.temp\_max)

xlabel('Precipitation')

ylabel('Temperature Max')

title('Precipitation vs Temperature Max')

%Precipitation vs Wind Speed

nexttile

scatter(data\_copy.precipitation, data\_copy.wind)

xlabel('Precipitation')

ylabel('Wind Speed')

title('Precipitation vs Wind Speed')

%Temperature Max vs Temperature Min

nexttile

scatter(data\_copy.temp\_max, data\_copy.temp\_min)

xlabel('Temperature Max')

ylabel('Temperature Min')

title('Temperature Max vs Temperature Min')

%Temperature Max vs Wind Speed

nexttile

scatter(data\_copy.temp\_max, data\_copy.wind)

xlabel('Temperature Max')

ylabel('Wind Speed')

title('Temperature Max vs Wind Speed')

%Wind Speed vs Temperature Min

nexttile

scatter(data\_copy.wind, data\_copy.temp\_min)

xlabel('Wind Speed')

ylabel('Temperature Min')

title('Wind Speed vs Temperature Min')

%Boxplot for different weather types and temp min vs temp max

% Boxplot for temp\_min for different weather types

figure;

boxplot(data.temp\_min, data.weather, 'Labels', unique(data.weather));

xlabel('Weather Type');

ylabel('Temperature Min');

title('Boxplot: Temperature Min across Weather Types');

% Boxplot for temp\_max for different weather types

figure;

boxplot(data.temp\_max, data.weather, 'Labels', unique(data.weather));

xlabel('Weather Type');

ylabel('Temperature Max');

title('Boxplot: Temperature Max across Weather Types');

%Correlation Matrix between predictors

%Reference: https://uk.mathworks.com/help/matlab/ref/heatmap.html

numeric\_columns\_copy = data\_copy{:, {'precipitation', 'temp\_max', 'temp\_min', 'wind'}};

figure(3)

corr = corr(numeric\_columns\_copy);

xvalues = {'precipitation', 'temp\_max', 'temp\_min', 'wind'};

yvalues = {'precipitation', 'temp\_max', 'temp\_min', 'wind'};

h = heatmap(xvalues, yvalues,corr);

%Subplots histograms for every variable count

%Reference: https://uk.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html

%Reference: https://uk.mathworks.com/help/matlab/ref/subplot.html

figure(4)

subplot(2, 2, 1);

histogram(data.precipitation);

xlabel('Precipitation');

ylabel('Counts');

title('Precipitation Count');

subplot(2, 2, 2);

histogram(data.temp\_max);

xlabel('Temp Max');

ylabel('Counts');

title('Temp Max Count');

subplot(2, 2, 3);

histogram(data.temp\_min);

xlabel('Temp Min');

ylabel('Counts');

title('Temp Min count ');

subplot(2, 2, 4);

histogram(data.wind);

xlabel('Wind');

ylabel('Counts');

title('Wind Count');

%%

%Temp\_range over time

figure(5)

plot(data\_copy.Year, data\_copy.temp\_range, 'o-', 'LineWidth', 2);

xlabel('Year');

ylabel('Temperature Range');

title('Temperature Range Over Years');

%%

%%

%Boxplots for each predictors

%Box plot for precipitation

figure(6)

subplot(2,3,1)

boxplot(data\_copy.precipitation);

title('Box Plot for Precipitation');

%Box plot for temp\_max

subplot(2,3,2)

boxplot(data\_copy.temp\_max);

title('Box Plot for Temperature Max');

%Box plot for temp\_min

subplot(2,3,3)

boxplot(data\_copy.temp\_min);

title('Box Plot for Temperature Min');

%Box plot for wind

subplot(2,3,4)

boxplot(data\_copy.wind);

title('Box Plot for Wind');

%Box plot for temp\_range

subplot(2,3,5)

boxplot(data\_copy.temp\_range);

title('Box Plot for Temperature range');

%%

%Reference: https://uk.mathworks.com/matlabcentral/answers/377839-split-training-data-and-testing-data%

% Reference: https://www.mathworks.com/help/stats/cvpartition.html

%Reference: https://uk.mathworks.com/help/matlab/ref/rng.html

%Using random number generator so data results is reproducible

rng(1)

%creating a crossvalidation partition using 'holdout' method

cv = cvpartition(size(data\_copy,1), 'HoldOut', 0.2)

%Assinging the index of the test set to to variable name idx

idx = cv.test;

% Splitting the data into training and testing sets using the partition

% ~idx takes the negative

trainingData = data\_copy(~idx,:);

testingData = data\_copy(idx,:);

%Saving Testing data into a matlab file for later use when predicting

save('test\_data.mat', 'testingData');

% Defining feature columns and target column

X = {'Year', 'Month', 'Day', 'precipitation', 'temp\_max', 'temp\_min', 'wind', 'temp\_range', 'Winter', 'Summer', 'Spring', 'Autumn'};

Y = 'weather\_labels';

% Separate features (X) and labels (Y) in the training set

XTrain = trainingData(:, X);

YTrain = trainingData.(Y);

% Separate features (X) and labels (Y) in the testing set

XTest = testingData(:, X);

YTest = testingData.(Y);

% Display the sizes of the training and testing sets

disp('Number of samples in the training set: ');

disp(size(trainingData));

disp('Number of samples in the testing set: ');

disp(size(testingData));

%%

%Training decision tree model

%Training model decision tree using fitctree

%tic toc takes the time

%Reference: https://uk.mathworks.com/help/matlab/ref/tic.html

tic

rng(1);

%Training the decision tree using fictree on the training data

dtTrain = fitctree(XTrain, YTrain);

toc

%%

%Predictions on the 20% testing set

predictions\_dtTrain = predict(dtTrain, XTest);

%displaying first couple rows of predictions

head(predictions\_dtTrain);

%Saving the training model predictions in a csv

%writematrix(predictions\_dtTrain, 'predictions\_dtTrain.csv');

%%

%Reference: https://uk.mathworks.com/help/matlab/ref/eq.html

%Reference: https://uk.mathworks.com/help/matlab/ref/sum.html

%Accuracy

%Summing all correct predictions by comparing to YTest (True values)

correctPredictions\_dtTrain = sum(YTest == predictions\_dtTrain);

%Total number

totalPredictions\_dtTrain = length(YTest);

% Calculate test accuracy

%By diving number of corect predictions by

%Number of correct predictions/(lenght of test set = 292)

testAccuracy\_dtTrain = correctPredictions\_dtTrain /292;

AccuracyPercentage\_dtTrain = testAccuracy\_dtTrain\*100

%Reference: https://uk.mathworks.com/help/matlab/ref/num2str.html

disp(['Test Accuracy: ' num2str(testAccuracy\_dtTrain)]);

%Calculating the error (Amount of incorrect predictions)

error\_dtTrain = 100- AccuracyPercentage\_dtTrain;

error\_dtTrain

%%

%Results

%Reference: https://uk.mathworks.com/help/stats/confusionmat.html

%calculating the confusion matrix given the true labels and predicted

%Where dt stands for Decision Tree

results\_dtTrain = confusionmat(YTest, predictions\_dtTrain);

%Displaying

results\_dtTrain

%Total number of predictions made by model which should equate to the..

%lenght of Test set

results\_sum\_dtTrain = sum(sum(results\_dtTrain));

results\_sum\_dtTrain

%Heatmap visualization of the confusion matrix

figure;

dtTrain\_Heatmap= heatmap(results\_dtTrain);

%%

unique\_labels = unique(data\_copy.weather\_labels);

unique\_labels;

%The unique labels correspond as follows:

%1 -> Drizzle

%2 -> Fog

%3 -> Rain

%4 -> snow

%5 -> Sun

%%

% Performance metrics

%dt stands for Decision Tree

% Class 1 (Drizzle)

% True Positive

TP\_Class1\_dtTrain = results\_dtTrain(1, 1);

% False Negative

FN\_Class1\_dtTrain = sum(results\_dtTrain(:, 1)) - TP\_Class1\_dtTrain;

% False Positive

FP\_Class1\_dtTrain = sum(results\_dtTrain(1, :)) - TP\_Class1\_dtTrain;

% True Negative

TN\_Class1\_dtTrain = sum(results\_dtTrain(:)) - (TP\_Class1\_dtTrain + FP\_Class1\_dtTrain + FN\_Class1\_dtTrain);

% Precision

precision\_Class1\_dtTrain = TP\_Class1\_dtTrain / (TP\_Class1\_dtTrain + FP\_Class1\_dtTrain);

% Recall (Sensitivity)

recall\_Class1\_dtTrain = TP\_Class1\_dtTrain / (TP\_Class1\_dtTrain + FN\_Class1\_dtTrain);

% F1 Score

f1Score\_Class1\_dtTrain = 2 \* (precision\_Class1\_dtTrain \* recall\_Class1\_dtTrain) / (precision\_Class1\_dtTrain + recall\_Class1\_dtTrain);

% Accuracy

accuracy\_Class1\_dtTrain = (TP\_Class1\_dtTrain + TN\_Class1\_dtTrain) / sum(results\_dtTrain(:));

% Displaying results for Class 1 (drizzle)

disp(['Class 1 (Drizzle)']);

disp(['True Positive: ', num2str(TP\_Class1\_dtTrain)]);

disp(['False Negative: ', num2str(FN\_Class1\_dtTrain)]);

disp(['False Positive: ', num2str(FP\_Class1\_dtTrain)]);

disp(['True Negative: ', num2str(TN\_Class1\_dtTrain)]);

disp(['Precision: ', num2str(precision\_Class1\_dtTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class1\_dtTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class1\_dtTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class1\_dtTrain)]);

disp('----------------------');

%---------------------------------------------

%Class 2 (Fog)

TP\_Class2\_dtTrain = results\_dtTrain(2, 2);

FN\_Class2\_dtTrain = sum(results\_dtTrain(:, 2)) - TP\_Class2\_dtTrain;

FP\_Class2\_dtTrain = sum(results\_dtTrain(2, :)) - TP\_Class2\_dtTrain;

TN\_Class2\_dtTrain = sum(results\_dtTrain(:)) - (TP\_Class2\_dtTrain + FP\_Class2\_dtTrain + FN\_Class2\_dtTrain);

precision\_Class2\_dtTrain = TP\_Class2\_dtTrain / (TP\_Class2\_dtTrain + FP\_Class2\_dtTrain);

recall\_Class2\_dtTrain = TP\_Class2\_dtTrain / (TP\_Class2\_dtTrain + FN\_Class2\_dtTrain);

f1Score\_Class2\_dtTrain = 2 \* (precision\_Class2\_dtTrain \* recall\_Class2\_dtTrain) / (precision\_Class2\_dtTrain + recall\_Class2\_dtTrain);

accuracy\_Class2\_dtTrain = (TP\_Class2\_dtTrain + TN\_Class2\_dtTrain) / sum(results\_dtTrain(:));

% Displaying results for Class 2 (fog)

disp(['Class 2 (Fog)']);

disp(['True Positive: ', num2str(TP\_Class2\_dtTrain)]);

disp(['False Negative: ', num2str(FN\_Class2\_dtTrain)]);

disp(['False Positive: ', num2str(FP\_Class2\_dtTrain)]);

disp(['True Negative: ', num2str(TN\_Class2\_dtTrain)]);

disp(['Precision: ', num2str(precision\_Class2\_dtTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class2\_dtTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class2\_dtTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class2\_dtTrain)]);

disp('----------------------');

%---------------------------------------------

%Class 3 (rain)

TP\_Class3\_dtTrain = results\_dtTrain(3, 3);

FN\_Class3\_dtTrain = sum(results\_dtTrain(:, 3)) - TP\_Class3\_dtTrain;

FP\_Class3\_dtTrain = sum(results\_dtTrain(3, :)) - TP\_Class3\_dtTrain;

TN\_Class3\_dtTrain = sum(results\_dtTrain(:)) - (TP\_Class3\_dtTrain + FP\_Class3\_dtTrain + FN\_Class3\_dtTrain);

precision\_Class3\_dtTrain = TP\_Class3\_dtTrain / (TP\_Class3\_dtTrain + FP\_Class3\_dtTrain);

recall\_Class3\_dtTrain = TP\_Class3\_dtTrain / (TP\_Class3\_dtTrain + FN\_Class3\_dtTrain);

f1Score\_Class3\_dtTrain = 2 \* (precision\_Class3\_dtTrain \* recall\_Class3\_dtTrain) / (precision\_Class3\_dtTrain + recall\_Class3\_dtTrain);

accuracy\_Class3\_dtTrain = (TP\_Class3\_dtTrain + TN\_Class3\_dtTrain) / sum(results\_dtTrain(:));

% Displaying results for Class 3 (rain)

disp(['Class 3 (Rain)']);

disp(['True Positive: ', num2str(TP\_Class3\_dtTrain)]);

disp(['False Negative: ', num2str(FN\_Class3\_dtTrain)]);

disp(['False Positive: ', num2str(FP\_Class3\_dtTrain)]);

disp(['True Negative: ', num2str(TN\_Class3\_dtTrain)]);

disp(['Precision: ', num2str(precision\_Class3\_dtTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class3\_dtTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class3\_dtTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class3\_dtTrain)]);

disp('----------------------');

%---------------------------------------------

%Class 4 (snow)

TP\_Class4\_dtTrain = results\_dtTrain(4, 4);

FN\_Class4\_dtTrain = sum(results\_dtTrain(:, 4)) - TP\_Class4\_dtTrain;

FP\_Class4\_dtTrain = sum(results\_dtTrain(4, :)) - TP\_Class4\_dtTrain;

TN\_Class4\_dtTrain = sum(results\_dtTrain(:)) - (TP\_Class4\_dtTrain + FP\_Class4\_dtTrain + FN\_Class4\_dtTrain);

precision\_Class4\_dtTrain = TP\_Class4\_dtTrain / (TP\_Class4\_dtTrain + FP\_Class4\_dtTrain);

recall\_Class4\_dtTrain = TP\_Class4\_dtTrain / (TP\_Class4\_dtTrain + FN\_Class4\_dtTrain);

f1Score\_Class4\_dtTrain = 2 \* (precision\_Class4\_dtTrain \* recall\_Class4\_dtTrain) / (precision\_Class4\_dtTrain + recall\_Class4\_dtTrain);

accuracy\_Class4\_dtTrain = (TP\_Class4\_dtTrain + TN\_Class4\_dtTrain) / sum(results\_dtTrain(:));

% Displaying results for Class 4 (snow)

disp(['Class 4 (Snow)']);

disp(['True Positive: ', num2str(TP\_Class4\_dtTrain)]);

disp(['False Negative: ', num2str(FN\_Class4\_dtTrain)]);

disp(['False Positive: ', num2str(FP\_Class4\_dtTrain)]);

disp(['True Negative: ', num2str(TN\_Class4\_dtTrain)]);

disp(['Precision: ', num2str(precision\_Class4\_dtTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class4\_dtTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class4\_dtTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class4\_dtTrain)]);

disp('----------------------');

%---------------------------------------------

%Class 5 (Sun)

TP\_Class5\_dtTrain = results\_dtTrain(5, 5);

FN\_Class5\_dtTrain = sum(results\_dtTrain(:, 5)) - TP\_Class5\_dtTrain;

FP\_Class5\_dtTrain = sum(results\_dtTrain(5, :)) - TP\_Class5\_dtTrain;

TN\_Class5\_dtTrain = sum(results\_dtTrain(:)) - (TP\_Class5\_dtTrain + FP\_Class5\_dtTrain + FN\_Class5\_dtTrain);

precision\_Class5\_dtTrain = TP\_Class5\_dtTrain / (TP\_Class5\_dtTrain + FP\_Class5\_dtTrain);

recall\_Class5\_dtTrain = TP\_Class5\_dtTrain / (TP\_Class5\_dtTrain + FN\_Class5\_dtTrain);

f1Score\_Class5\_dtTrain = 2 \* (precision\_Class5\_dtTrain \* recall\_Class5\_dtTrain) / (precision\_Class5\_dtTrain + recall\_Class5\_dtTrain);

accuracy\_Class5\_dtTrain = (TP\_Class5\_dtTrain + TN\_Class5\_dtTrain) / sum(results\_dtTrain(:));

% Displaying results for Class 5 (sun)

disp(['Class 5 (Sun)']);

disp(['True Positive: ', num2str(TP\_Class5\_dtTrain)]);

disp(['False Negative: ', num2str(FN\_Class5\_dtTrain)]);

disp(['False Positive: ', num2str(FP\_Class5\_dtTrain)]);

disp(['True Negative: ', num2str(TN\_Class5\_dtTrain)]);

disp(['Precision: ', num2str(precision\_Class5\_dtTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class5\_dtTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class5\_dtTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class5\_dtTrain)]);

disp('----------------------');

%%

tic

rng(1)

%Decision Tree Hyperparameter optimization

decisionTree\_HP = fitctree(XTrain, YTrain, 'OptimizeHyperparameters','auto');

toc

%%

%saving and loading

% Save the decision tree model

save('decisionTree\_HP.mat', 'decisionTree\_HP');

rng(1)

% Load the saved decision tree model

load('decisionTree\_HP.mat');

%%

%loading test data

load('test\_data.mat');

%cv2 = crossval(decisiontreeHP, 'Holdout', 0.2);

%Splitting

cv = cvpartition(size(data\_copy,1), 'HoldOut', 0.2)

idx = cv.test;

testingData = data\_copy(idx,:);

% Define feature columns

X = {'Year', 'Month', 'Day', 'precipitation', 'temp\_max', 'temp\_min', 'wind', 'temp\_range', 'Winter', 'Summer', 'Spring', 'Autumn'};

Y = 'weather\_labels';

XTest = testingData(:, X);

YTest = testingData.(Y);

%%

predictions\_dtTrainHP = predict(decisionTree\_HP, XTest);

%displaying first couple rows of predictions

head(predictions\_dtTrainHP);

%Saving the training model predictions in a csv

%writematrix(predictions\_dtTrainHP, 'predictions\_dtTrain.csv');

%%

%Accuracy

%Summing all correct predictions by comparing to YTest (True values)

correctPredictions\_dtTrainHP = sum(YTest == predictions\_dtTrainHP);

%Total number

totalPredictions\_dtTrainHP = length(YTest);

testAccuracy\_dtTrainHP = correctPredictions\_dtTrainHP /292;

AccuracyPercentage\_dtTrainHP = testAccuracy\_dtTrainHP \*100

disp(['Test Accuracy: ' num2str(testAccuracy\_dtTrainHP)]);

%%

%Results

%Where dtTrainHP stands for Decision tree Training Hyper Parameter

results\_dtTrainHP = confusionmat(YTest, predictions\_dtTrainHP);

results\_dtTrainHP

results\_sum\_dtTrainHP = sum(sum(results\_dtTrainHP));

results\_sum\_dtTrainHP

figure;

dtTrain\_HeatmapHP= heatmap(results\_dtTrainHP);

%%

%Evaluation Metrics

%Drizzle

%Where dtTrainHP stands for Decision tree Training Hyper Parameter

% True Positive

TP\_Class1\_dtTrainHP = results\_dtTrainHP(1, 1);

% False Negative

FN\_Class1\_dtTrainHP = sum(results\_dtTrainHP(:, 1)) - TP\_Class1\_dtTrainHP;

% False Positive

FP\_Class1\_dtTrainHP = sum(results\_dtTrainHP(1, :)) - TP\_Class1\_dtTrainHP;

% True Negative

TN\_Class1\_dtTrainHP = sum(results\_dtTrainHP(:)) - (TP\_Class1\_dtTrainHP + FP\_Class1\_dtTrainHP + FN\_Class1\_dtTrainHP);

% Precision

precision\_Class1\_dtTrainHP = TP\_Class1\_dtTrainHP / (TP\_Class1\_dtTrainHP + FP\_Class1\_dtTrainHP);

% Recall (Sensitivity)

recall\_Class1\_dtTrainHP = TP\_Class1\_dtTrainHP / (TP\_Class1\_dtTrainHP + FN\_Class1\_dtTrainHP);

% F1 Score

f1Score\_Class1\_dtTrainHP = 2 \* (precision\_Class1\_dtTrainHP \* recall\_Class1\_dtTrainHP) / (precision\_Class1\_dtTrainHP + recall\_Class1\_dtTrainHP);

% Accuracy

accuracy\_Class1\_dtTrainHP = (TP\_Class1\_dtTrainHP + TN\_Class1\_dtTrainHP) / sum(results\_dtTrainHP(:));

disp(['Class 1 (Drizzle)']);

disp(['True Positive: ', num2str(TP\_Class1\_dtTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class1\_dtTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class1\_dtTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class1\_dtTrainHP)]);

disp(['Precision: ', num2str(precision\_Class1\_dtTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class1\_dtTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class1\_dtTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class1\_dtTrainHP)]);

disp('----------------------');

%------------------------------------------

%Fog

TP\_Class2\_dtTrainHP = results\_dtTrainHP(2, 2);

FN\_Class2\_dtTrainHP = sum(results\_dtTrainHP(:, 2)) - TP\_Class2\_dtTrainHP;

FP\_Class2\_dtTrainHP = sum(results\_dtTrainHP(2, :)) - TP\_Class2\_dtTrainHP;

TN\_Class2\_dtTrainHP = sum(results\_dtTrainHP(:)) - (TP\_Class2\_dtTrainHP + FP\_Class2\_dtTrainHP + FN\_Class2\_dtTrainHP);

precision\_Class2\_dtTrainHP = TP\_Class2\_dtTrainHP / (TP\_Class2\_dtTrainHP + FP\_Class2\_dtTrainHP);

recall\_Class2\_dtTrainHP = TP\_Class2\_dtTrainHP / (TP\_Class2\_dtTrainHP + FN\_Class2\_dtTrainHP);

f1Score\_Class2\_dtTrainHP = 2 \* (precision\_Class2\_dtTrainHP \* recall\_Class2\_dtTrainHP) / (precision\_Class2\_dtTrainHP + recall\_Class2\_dtTrainHP);

accuracy\_Class2\_dtTrainHP = (TP\_Class2\_dtTrainHP + TN\_Class2\_dtTrainHP) / sum(results\_dtTrainHP(:));

disp('Class 2 (Fog)');

disp(['True Positive: ', num2str(TP\_Class2\_dtTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class2\_dtTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class2\_dtTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class2\_dtTrainHP)]);

disp(['Precision: ', num2str(precision\_Class2\_dtTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class2\_dtTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class2\_dtTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class2\_dtTrainHP)]);

disp('----------------------');

%--------------------------------------------------

%Rain

TP\_Class3\_dtTrainHP = results\_dtTrainHP(3, 3);

FN\_Class3\_dtTrainHP = sum(results\_dtTrainHP(:, 3)) - TP\_Class3\_dtTrainHP;

FP\_Class3\_dtTrainHP = sum(results\_dtTrainHP(3, :)) - TP\_Class3\_dtTrainHP;

TN\_Class3\_dtTrainHP = sum(results\_dtTrainHP(:)) - (TP\_Class3\_dtTrainHP + FP\_Class3\_dtTrainHP + FN\_Class3\_dtTrainHP);

precision\_Class3\_dtTrainHP = TP\_Class3\_dtTrainHP / (TP\_Class3\_dtTrainHP + FP\_Class3\_dtTrainHP);

recall\_Class3\_dtTrainHP = TP\_Class3\_dtTrainHP / (TP\_Class3\_dtTrainHP + FN\_Class3\_dtTrainHP);

f1Score\_Class3\_dtTrainHP = 2 \* (precision\_Class3\_dtTrainHP \* recall\_Class3\_dtTrainHP) / (precision\_Class3\_dtTrainHP + recall\_Class3\_dtTrainHP);

accuracy\_Class3\_dtTrainHP = (TP\_Class3\_dtTrainHP + TN\_Class3\_dtTrainHP) / sum(results\_dtTrainHP(:));

disp('Class 3 (Rain)');

disp(['True Positive: ', num2str(TP\_Class3\_dtTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class3\_dtTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class3\_dtTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class3\_dtTrainHP)]);

disp(['Precision: ', num2str(precision\_Class3\_dtTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class3\_dtTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class3\_dtTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class3\_dtTrainHP)]);

disp('----------------------');

%--------------------------------------------------

%Snow

TP\_Class4\_dtTrainHP = results\_dtTrainHP(4, 4);

FN\_Class4\_dtTrainHP = sum(results\_dtTrainHP(:, 4)) - TP\_Class4\_dtTrainHP;

FP\_Class4\_dtTrainHP = sum(results\_dtTrainHP(4, :)) - TP\_Class4\_dtTrainHP;

TN\_Class4\_dtTrainHP = sum(results\_dtTrainHP(:)) - (TP\_Class4\_dtTrainHP + FP\_Class4\_dtTrainHP + FN\_Class4\_dtTrainHP);

precision\_Class4\_dtTrainHP = TP\_Class4\_dtTrainHP / (TP\_Class4\_dtTrainHP + FP\_Class4\_dtTrainHP);

recall\_Class4\_dtTrainHP = TP\_Class4\_dtTrainHP / (TP\_Class4\_dtTrainHP + FN\_Class4\_dtTrainHP);

f1Score\_Class4\_dtTrainHP = 2 \* (precision\_Class4\_dtTrainHP \* recall\_Class4\_dtTrainHP) / (precision\_Class4\_dtTrainHP + recall\_Class4\_dtTrainHP);

accuracy\_Class4\_dtTrainHP = (TP\_Class4\_dtTrainHP + TN\_Class4\_dtTrainHP) / sum(results\_dtTrainHP(:));

disp('Class 4 (Snow)');

disp(['True Positive: ', num2str(TP\_Class4\_dtTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class4\_dtTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class4\_dtTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class4\_dtTrainHP)]);

disp(['Precision: ', num2str(precision\_Class4\_dtTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class4\_dtTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class4\_dtTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class4\_dtTrainHP)]);

disp('----------------------');

%--------------------------------------------------

%Sun

TP\_Class5\_dtTrainHP = results\_dtTrainHP(5, 5);

FN\_Class5\_dtTrainHP = sum(results\_dtTrainHP(:, 5)) - TP\_Class5\_dtTrainHP;

FP\_Class5\_dtTrainHP = sum(results\_dtTrainHP(5, :)) - TP\_Class5\_dtTrainHP;

TN\_Class5\_dtTrainHP = sum(results\_dtTrainHP(:)) - (TP\_Class5\_dtTrainHP + FP\_Class5\_dtTrainHP + FN\_Class5\_dtTrainHP);

precision\_Class5\_dtTrainHP = TP\_Class5\_dtTrainHP / (TP\_Class5\_dtTrainHP + FP\_Class5\_dtTrainHP);

recall\_Class5\_dtTrainHP = TP\_Class5\_dtTrainHP / (TP\_Class5\_dtTrainHP + FN\_Class5\_dtTrainHP);

f1Score\_Class5\_dtTrainHP = 2 \* (precision\_Class5\_dtTrainHP \* recall\_Class5\_dtTrainHP) / (precision\_Class5\_dtTrainHP + recall\_Class5\_dtTrainHP);

accuracy\_Class5\_dtTrainHP = (TP\_Class5\_dtTrainHP + TN\_Class5\_dtTrainHP) / sum(results\_dtTrainHP(:));

disp(['Class 5 (Sun)']);

disp(['True Positive: ', num2str(TP\_Class5\_dtTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class5\_dtTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class5\_dtTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class5\_dtTrainHP)]);

disp(['Precision: ', num2str(precision\_Class5\_dtTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class5\_dtTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class5\_dtTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class5\_dtTrainHP)]);

disp('----------------------');

%--------------------------------------------------

%%

%Comparison between HyperParameter Tuning and Original

AccuracyPercentage\_dtTrainHP

AccuracyPercentage\_dtTrain

% Compare in a bar chart

figure;

bar([AccuracyPercentage\_dtTrainHP, AccuracyPercentage\_dtTrain]);

xticks(1:2);

xticklabels({'Hyperparameter Tuning', 'Original Model'});

ylabel('Accuracy');

title('Decision Tree');

%%

%Model 2: Random Forest

%Training Random forest model

%Training Random forest using fitenemble

tic

%Random Number generator

%Reference: https://uk.mathworks.com/help/matlab/ref/rng.html

rng(1);

%Reference:https://uk.mathworks.com/help/stats/select-predictors-for-random-forests.html

%Where rf stands for Random Forest

rfTrain = fitensemble(XTrain, YTrain, 'Bag', 100, 'Tree', 'Type', 'classification');

toc

%%

%Predictions on the 20% testing set

predictions\_rfTrain = predict(rfTrain, XTest);

%displaying first couple rows of predictions

head(predictions\_rfTrain);

%Saving the training model predictions in a csv

%writematrix(predictions\_dtTrain, 'predictions\_dtTrain.csv');

%%

%Accuracy

%Summing all correct predictions by comparing to YTest (True values)

correctPredictions\_rfTrain = sum(YTest == predictions\_rfTrain);

%Total number

totalPredictions\_rfTrain = length(YTest);

%By diving number of corect predictions by

%Number of correct predictions/(lenght of test set = 292)

testAccuracy\_rfTrain = correctPredictions\_rfTrain /292;

AccuracyPercentage\_rfTrain = testAccuracy\_rfTrain\*100

disp(['Test Accuracy: ' num2str(testAccuracy\_rfTrain)]);

%%

%Results

results\_rfTrain = confusionmat(YTest, predictions\_rfTrain);

results\_rfTrain

results\_sum\_rfTrain = sum(sum(results\_rfTrain));

results\_sum\_rfTrain

figure;

rfTrain\_Heatmap= heatmap(results\_rfTrain);

%%

%Evaluation metrics

%Rain

TP\_Class1\_rfTrain = results\_rfTrain(1, 1);

FN\_Class1\_rfTrain = sum(results\_rfTrain(:, 1)) - TP\_Class1\_rfTrain;

FP\_Class1\_rfTrain = sum(results\_rfTrain(1, :)) - TP\_Class1\_rfTrain;

TN\_Class1\_rfTrain = sum(results\_rfTrain(:)) - (TP\_Class1\_rfTrain + FP\_Class1\_rfTrain + FN\_Class1\_rfTrain);

precision\_Class1\_rfTrain = TP\_Class1\_rfTrain / (TP\_Class1\_rfTrain + FP\_Class1\_rfTrain);

recall\_Class1\_rfTrain = TP\_Class1\_rfTrain / (TP\_Class1\_rfTrain + FN\_Class1\_rfTrain);

f1Score\_Class1\_rfTrain = 2 \* (precision\_Class1\_rfTrain \* recall\_Class1\_rfTrain) / (precision\_Class1\_rfTrain + recall\_Class1\_rfTrain);

accuracy\_Class1\_rfTrain = (TP\_Class1\_rfTrain + TN\_Class1\_rfTrain) / sum(results\_rfTrain(:));

disp(['Class 1 (Drizzle)']);

disp(['True Positive: ', num2str(TP\_Class1\_rfTrain)]);

disp(['False Negative: ', num2str(FN\_Class1\_rfTrain)]);

disp(['False Positive: ', num2str(FP\_Class1\_rfTrain)]);

disp(['True Negative: ', num2str(TN\_Class1\_rfTrain)]);

disp(['Precision: ', num2str(precision\_Class1\_rfTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class1\_rfTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class1\_rfTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class1\_rfTrain)]);

disp('----------------------');

%----------------------------------------------------

%Fog

TP\_Class2\_rfTrain = results\_rfTrain(2, 2);

FN\_Class2\_rfTrain = sum(results\_rfTrain(:, 2)) - TP\_Class2\_rfTrain;

FP\_Class2\_rfTrain = sum(results\_rfTrain(2, :)) - TP\_Class2\_rfTrain;

TN\_Class2\_rfTrain = sum(results\_rfTrain(:)) - (TP\_Class2\_rfTrain + FP\_Class2\_rfTrain + FN\_Class2\_rfTrain);

precision\_Class2\_rfTrain = TP\_Class2\_rfTrain / (TP\_Class2\_rfTrain + FP\_Class2\_rfTrain);

recall\_Class2\_rfTrain = TP\_Class2\_rfTrain / (TP\_Class2\_rfTrain + FN\_Class2\_rfTrain);

f1Score\_Class2\_rfTrain = 2 \* (precision\_Class2\_rfTrain \* recall\_Class2\_rfTrain) / (precision\_Class2\_rfTrain + recall\_Class2\_rfTrain);

accuracy\_Class2\_rfTrain = (TP\_Class2\_rfTrain + TN\_Class2\_rfTrain) / sum(results\_rfTrain(:));

disp(['Class 2 (Fog)']);

disp(['True Positive: ', num2str(TP\_Class2\_rfTrain)]);

disp(['False Negative: ', num2str(FN\_Class2\_rfTrain)]);

disp(['False Positive: ', num2str(FP\_Class2\_rfTrain)]);

disp(['True Negative: ', num2str(TN\_Class2\_rfTrain)]);

disp(['Precision: ', num2str(precision\_Class2\_rfTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class2\_rfTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class2\_rfTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class2\_rfTrain)]);

disp('----------------------');

%----------------------------------------------------

%Rain

TP\_Class3\_rfTrain = results\_rfTrain(3, 3);

FN\_Class3\_rfTrain = sum(results\_rfTrain(:, 3)) - TP\_Class3\_rfTrain;

FP\_Class3\_rfTrain = sum(results\_rfTrain(3, :)) - TP\_Class3\_rfTrain;

TN\_Class3\_rfTrain = sum(results\_rfTrain(:)) - (TP\_Class3\_rfTrain + FP\_Class3\_rfTrain + FN\_Class3\_rfTrain);

precision\_Class3\_rfTrain = TP\_Class3\_rfTrain / (TP\_Class3\_rfTrain + FP\_Class3\_rfTrain);

recall\_Class3\_rfTrain = TP\_Class3\_rfTrain / (TP\_Class3\_rfTrain + FN\_Class3\_rfTrain);

f1Score\_Class3\_rfTrain = 2 \* (precision\_Class3\_rfTrain \* recall\_Class3\_rfTrain) / (precision\_Class3\_rfTrain + recall\_Class3\_rfTrain);

accuracy\_Class3\_rfTrain = (TP\_Class3\_rfTrain + TN\_Class3\_rfTrain) / sum(results\_rfTrain(:));

disp(['Class 3 (Raim)']);

disp(['True Positive: ', num2str(TP\_Class3\_rfTrain)]);

disp(['False Negative: ', num2str(FN\_Class3\_rfTrain)]);

disp(['False Positive: ', num2str(FP\_Class3\_rfTrain)]);

disp(['True Negative: ', num2str(TN\_Class3\_rfTrain)]);

disp(['Precision: ', num2str(precision\_Class3\_rfTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class3\_rfTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class3\_rfTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class3\_rfTrain)]);

disp('----------------------');

%----------------------------------------------------

%Snow

TP\_Class4\_rfTrain = results\_rfTrain(4, 4);

FN\_Class4\_rfTrain = sum(results\_rfTrain(:, 4)) - TP\_Class4\_rfTrain;

FP\_Class4\_rfTrain = sum(results\_rfTrain(4, :)) - TP\_Class4\_rfTrain;

TN\_Class4\_rfTrain = sum(results\_rfTrain(:)) - (TP\_Class4\_rfTrain + FP\_Class4\_rfTrain + FN\_Class4\_rfTrain);

precision\_Class4\_rfTrain = TP\_Class4\_rfTrain / (TP\_Class4\_rfTrain + FP\_Class4\_rfTrain);

recall\_Class4\_rfTrain = TP\_Class4\_rfTrain / (TP\_Class4\_rfTrain + FN\_Class4\_rfTrain);

f1Score\_Class4\_rfTrain = 2 \* (precision\_Class4\_rfTrain \* recall\_Class4\_rfTrain) / (precision\_Class4\_rfTrain + recall\_Class4\_rfTrain);

accuracy\_Class4\_rfTrain = (TP\_Class4\_rfTrain + TN\_Class4\_rfTrain) / sum(results\_rfTrain(:));

disp(['Class 4 (Snow)']);

disp(['True Positive: ', num2str(TP\_Class4\_rfTrain)]);

disp(['False Negative: ', num2str(FN\_Class4\_rfTrain)]);

disp(['False Positive: ', num2str(FP\_Class4\_rfTrain)]);

disp(['True Negative: ', num2str(TN\_Class4\_rfTrain)]);

disp(['Precision: ', num2str(precision\_Class4\_rfTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class4\_rfTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class4\_rfTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class4\_rfTrain)]);

disp('----------------------');

%----------------------------------------------------

%Sun

TP\_Class5\_rfTrain = results\_rfTrain(5, 5);

FN\_Class5\_rfTrain = sum(results\_rfTrain(:, 5)) - TP\_Class5\_rfTrain;

FP\_Class5\_rfTrain = sum(results\_rfTrain(5, :)) - TP\_Class5\_rfTrain;

TN\_Class5\_rfTrain = sum(results\_rfTrain(:)) - (TP\_Class5\_rfTrain + FP\_Class5\_rfTrain + FN\_Class5\_rfTrain);

precision\_Class5\_rfTrain = TP\_Class5\_rfTrain / (TP\_Class5\_rfTrain + FP\_Class5\_rfTrain);

recall\_Class5\_rfTrain = TP\_Class5\_rfTrain / (TP\_Class5\_rfTrain + FN\_Class5\_rfTrain);

f1Score\_Class5\_rfTrain = 2 \* (precision\_Class5\_rfTrain \* recall\_Class5\_rfTrain) / (precision\_Class5\_rfTrain + recall\_Class5\_rfTrain);

accuracy\_Class5\_rfTrain = (TP\_Class5\_rfTrain + TN\_Class5\_rfTrain) / sum(results\_rfTrain(:));

disp(['Class 5 (Sun)']);

disp(['True Positive: ', num2str(TP\_Class5\_rfTrain)]);

disp(['False Negative: ', num2str(FN\_Class5\_rfTrain)]);

disp(['False Positive: ', num2str(FP\_Class5\_rfTrain)]);

disp(['True Negative: ', num2str(TN\_Class5\_rfTrain)]);

disp(['Precision: ', num2str(precision\_Class5\_rfTrain)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class5\_rfTrain)]);

disp(['F1 Score: ', num2str(f1Score\_Class5\_rfTrain)]);

disp(['Accuracy: ', num2str(accuracy\_Class5\_rfTrain)]);

disp('----------------------');

%----------------------------------------------------

%%

tic

rng(1)

RandomForest\_HP = fitcensemble(XTrain, YTrain, 'OptimizeHyperparameters', 'auto', 'Method', 'bag');

toc

%%

%saving and loading

% Save the decision tree model

%Reference:

save('RandomForest\_HP.mat', 'RandomForest\_HP');

rng(1)

% Load the saved decision tree model

load('RandomForest\_HP.mat');

%%

%loading test data

load('test\_data.mat');

%cv2 = crossval(decisiontreeHP, 'Holdout', 0.2);

%Splitting

cv = cvpartition(size(data\_copy,1), 'HoldOut', 0.2)

idx = cv.test;

testingData = data\_copy(idx,:);

% Define feature columns

X = {'Year', 'Month', 'Day', 'precipitation', 'temp\_max', 'temp\_min', 'wind', 'temp\_range', 'Winter', 'Summer', 'Spring', 'Autumn'};

Y = 'weather\_labels';

XTest = testingData(:, X);

YTest = testingData.(Y);

%%

%Where rfTrainHP stands for Random Forest Training Hyper Parameter

predictions\_rfTrainHP = predict(RandomForest\_HP, XTest);

%displaying first couple rows of predictions

head(predictions\_rfTrainHP);

%Saving the training model predictions in a csv

%writematrix(predictions\_dtTrainHP, 'predictions\_dtTrain.csv');

%%

%Accuracy

%Summing all correct predictions by comparing to YTest (True values)

correctPredictions\_rfTrainHP = sum(YTest == predictions\_rfTrainHP);

%Total number

totalPredictions\_rfTrainHP = length(YTest);

% Calculate test accuracy

%By diving number of corect predictions by

%Number of correct predictions/(lenght of test set = 292)

testAccuracy\_rfTrainHP = correctPredictions\_rfTrainHP /292;

AccuracyPercentage\_rfTrainHP = testAccuracy\_rfTrainHP \*100

disp(['Test Accuracy: ' num2str(testAccuracy\_rfTrainHP)]);

%%

%Results

%

results\_rfTrainHP = confusionmat(YTest, predictions\_rfTrainHP);

results\_rfTrainHP

results\_sum\_rfTrainHP = sum(sum(results\_rfTrainHP));

results\_sum\_rfTrainHP

figure;

rfTrain\_HeatmapHP= heatmap(results\_rfTrainHP);

%%

%Drizzle

TP\_Class1\_rfTrainHP = results\_rfTrainHP(1, 1);

FN\_Class1\_rfTrainHP = sum(results\_rfTrainHP(:, 1)) - TP\_Class1\_rfTrainHP;

FP\_Class1\_rfTrainHP = sum(results\_rfTrainHP(1, :)) - TP\_Class1\_rfTrainHP;

TN\_Class1\_rfTrainHP = sum(results\_rfTrainHP(:)) - (TP\_Class1\_rfTrainHP + FP\_Class1\_rfTrainHP + FN\_Class1\_rfTrainHP);

precision\_Class1\_rfTrainHP = TP\_Class1\_rfTrainHP / (TP\_Class1\_rfTrainHP + FP\_Class1\_rfTrainHP);

recall\_Class1\_rfTrainHP = TP\_Class1\_rfTrainHP / (TP\_Class1\_rfTrainHP + FN\_Class1\_rfTrainHP);

f1Score\_Class1\_rfTrainHP = 2 \* (precision\_Class1\_rfTrainHP \* recall\_Class1\_rfTrainHP) / (precision\_Class1\_rfTrainHP + recall\_Class1\_rfTrainHP);

accuracy\_Class1\_rfTrainHP = (TP\_Class1\_rfTrainHP + TN\_Class1\_rfTrainHP) / sum(results\_rfTrainHP(:));

disp(['Class 1 (Drizzle)']);

disp(['True Positive: ', num2str(TP\_Class1\_rfTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class1\_rfTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class1\_rfTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class1\_rfTrainHP)]);

disp(['Precision: ', num2str(precision\_Class1\_rfTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class1\_rfTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class1\_rfTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class1\_rfTrainHP)]);

disp('----------------------');

%----------------------------------------------------

%Fog

TP\_Class2\_rfTrainHP = results\_rfTrainHP(2, 2);

FN\_Class2\_rfTrainHP = sum(results\_rfTrainHP(:, 2)) - TP\_Class2\_rfTrainHP;

FP\_Class2\_rfTrainHP = sum(results\_rfTrainHP(2, :)) - TP\_Class2\_rfTrainHP;

TN\_Class2\_rfTrainHP = sum(results\_rfTrainHP(:)) - (TP\_Class2\_rfTrainHP + FP\_Class2\_rfTrainHP + FN\_Class2\_rfTrainHP);

precision\_Class2\_rfTrainHP = TP\_Class2\_rfTrainHP / (TP\_Class2\_rfTrainHP + FP\_Class2\_rfTrainHP);

recall\_Class2\_rfTrainHP = TP\_Class2\_rfTrainHP / (TP\_Class2\_rfTrainHP + FN\_Class2\_rfTrainHP);

f1Score\_Class2\_rfTrainHP = 2 \* (precision\_Class2\_rfTrainHP \* recall\_Class2\_rfTrainHP) / (precision\_Class2\_rfTrainHP + recall\_Class2\_rfTrainHP);

accuracy\_Class2\_rfTrainHP = (TP\_Class2\_rfTrainHP + TN\_Class2\_rfTrainHP) / sum(results\_rfTrainHP(:));

disp(['Class 2 (Fog)']);

disp(['True Positive: ', num2str(TP\_Class2\_rfTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class2\_rfTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class2\_rfTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class2\_rfTrainHP)]);

disp(['Precision: ', num2str(precision\_Class2\_rfTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class2\_rfTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class2\_rfTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class2\_rfTrainHP)]);

disp('----------------------');

%----------------------------------------------------

%Rain

TP\_Class3\_rfTrainHP = results\_rfTrainHP(3, 3);

FN\_Class3\_rfTrainHP = sum(results\_rfTrainHP(:, 3)) - TP\_Class3\_rfTrainHP;

FP\_Class3\_rfTrainHP = sum(results\_rfTrainHP(3, :)) - TP\_Class3\_rfTrainHP;

TN\_Class3\_rfTrainHP = sum(results\_rfTrainHP(:)) - (TP\_Class3\_rfTrainHP + FP\_Class3\_rfTrainHP + FN\_Class3\_rfTrainHP);

precision\_Class3\_rfTrainHP = TP\_Class3\_rfTrainHP / (TP\_Class3\_rfTrainHP + FP\_Class3\_rfTrainHP);

recall\_Class3\_rfTrainHP = TP\_Class3\_rfTrainHP / (TP\_Class3\_rfTrainHP + FN\_Class3\_rfTrainHP);

f1Score\_Class3\_rfTrainHP = 2 \* (precision\_Class3\_rfTrainHP \* recall\_Class3\_rfTrainHP) / (precision\_Class3\_rfTrainHP + recall\_Class3\_rfTrainHP);

accuracy\_Class3\_rfTrainHP = (TP\_Class3\_rfTrainHP + TN\_Class3\_rfTrainHP) / sum(results\_rfTrainHP(:));

disp(['Class 3 (Rain)']);

disp(['True Positive: ', num2str(TP\_Class3\_rfTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class3\_rfTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class3\_rfTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class3\_rfTrainHP)]);

disp(['Precision: ', num2str(precision\_Class3\_rfTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class3\_rfTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class3\_rfTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class3\_rfTrainHP)]);

disp('----------------------');

%----------------------------------------------------

%Snow

TP\_Class4\_rfTrainHP = results\_rfTrainHP(4, 4);

FN\_Class4\_rfTrainHP = sum(results\_rfTrainHP(:, 4)) - TP\_Class4\_rfTrainHP;

FP\_Class4\_rfTrainHP = sum(results\_rfTrainHP(4, :)) - TP\_Class4\_rfTrainHP;

TN\_Class4\_rfTrainHP = sum(results\_rfTrainHP(:)) - (TP\_Class4\_rfTrainHP + FP\_Class4\_rfTrainHP + FN\_Class4\_rfTrainHP);

precision\_Class4\_rfTrainHP = TP\_Class4\_rfTrainHP / (TP\_Class4\_rfTrainHP + FP\_Class4\_rfTrainHP);

recall\_Class4\_rfTrainHP = TP\_Class4\_rfTrainHP / (TP\_Class4\_rfTrainHP + FN\_Class4\_rfTrainHP);

f1Score\_Class4\_rfTrainHP = 2 \* (precision\_Class4\_rfTrainHP \* recall\_Class4\_rfTrainHP) / (precision\_Class4\_rfTrainHP + recall\_Class4\_rfTrainHP);

accuracy\_Class4\_rfTrainHP = (TP\_Class4\_rfTrainHP + TN\_Class4\_rfTrainHP) / sum(results\_rfTrainHP(:));

disp(['Class 4 (Snow)']);

disp(['True Positive: ', num2str(TP\_Class4\_rfTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class4\_rfTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class4\_rfTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class4\_rfTrainHP)]);

disp(['Precision: ', num2str(precision\_Class4\_rfTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class4\_rfTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class4\_rfTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class4\_rfTrainHP)]);

disp('----------------------');

%----------------------------------------------------

%Sun

TP\_Class5\_rfTrainHP = results\_rfTrainHP(5, 5);

FN\_Class5\_rfTrainHP = sum(results\_rfTrainHP(:, 5)) - TP\_Class5\_rfTrainHP;

FP\_Class5\_rfTrainHP = sum(results\_rfTrainHP(5, :)) - TP\_Class5\_rfTrainHP;

TN\_Class5\_rfTrainHP = sum(results\_rfTrainHP(:)) - (TP\_Class5\_rfTrainHP + FP\_Class5\_rfTrainHP + FN\_Class5\_rfTrainHP);

precision\_Class5\_rfTrainHP = TP\_Class5\_rfTrainHP / (TP\_Class5\_rfTrainHP + FP\_Class5\_rfTrainHP);

recall\_Class5\_rfTrainHP = TP\_Class5\_rfTrainHP / (TP\_Class5\_rfTrainHP + FN\_Class5\_rfTrainHP);

f1Score\_Class5\_rfTrainHP = 2 \* (precision\_Class5\_rfTrainHP \* recall\_Class5\_rfTrainHP) / (precision\_Class5\_rfTrainHP + recall\_Class5\_rfTrainHP);

accuracy\_Class5\_rfTrainHP = (TP\_Class5\_rfTrainHP + TN\_Class5\_rfTrainHP) / sum(results\_rfTrainHP(:));

disp(['Class 5 (Sun)']);

disp(['True Positive: ', num2str(TP\_Class5\_rfTrainHP)]);

disp(['False Negative: ', num2str(FN\_Class5\_rfTrainHP)]);

disp(['False Positive: ', num2str(FP\_Class5\_rfTrainHP)]);

disp(['True Negative: ', num2str(TN\_Class5\_rfTrainHP)]);

disp(['Precision: ', num2str(precision\_Class5\_rfTrainHP)]);

disp(['Recall (Sensitivity): ', num2str(recall\_Class5\_rfTrainHP)]);

disp(['F1 Score: ', num2str(f1Score\_Class5\_rfTrainHP)]);

disp(['Accuracy: ', num2str(accuracy\_Class5\_rfTrainHP)]);

disp('----------------------');

%----------------------------------------------------

%%

%Comparison between HyperParameter Tuning and Original

AccuracyPercentage\_rfTrainHP

AccuracyPercentage\_rfTrain

% Compare in a bar chart

figure;

bar([AccuracyPercentage\_rfTrainHP, AccuracyPercentage\_rfTrain]);

xticks(1:2);

xticklabels({'Hyperparameter Tuning', 'Original Model'});

ylabel('Accuracy');

title('Random Forest');

%%

%Model Comparison

figure;

bar([AccuracyPercentage\_rfTrainHP, AccuracyPercentage\_dtTrain, AccuracyPercentage\_rfTrain, AccuracyPercentage\_dtTrainHP]);

xticks(1:4);

xticklabels({'Random forest Hyperparameter', 'Decision Tree', 'Random Forest Original', 'Decision Tree Hyper Parameter'});

xtickangle(90);

ylabel('Accuracy');

title('Comparison of models');

%%

%subplots of each weather

%Drizzle

figure;

subplot(2,3,1)

bar([precision\_Class1\_rfTrainHP,precision\_Class1\_rfTrain, precision\_Class1\_dtTrainHP, precision\_Class1\_dtTrain,recall\_Class1\_rfTrainHP,recall\_Class1\_rfTrain,recall\_Class1\_dtTrainHP,recall\_Class1\_dtTrain,f1Score\_Class1\_rfTrainHP,f1Score\_Class1\_rfTrain,f1Score\_Class1\_dtTrainHP,f1Score\_Class1\_dtTrain]);

xticklabels({'PrecisionRF','PrecisionRFHP.','PrecisionDT','PrecisionDTHP','RecallRFHP','RecallRF','RecallDTHP','RecallDT','F1 Score RFHP','F1 Score RF','F1 Score DTHP','F1 Score DT'});

xtickangle(90);

title('Drizzle');

%Fog

subplot(2,3,2)

bar([precision\_Class2\_rfTrainHP, precision\_Class2\_rfTrain, precision\_Class2\_dtTrainHP, precision\_Class2\_dtTrain,recall\_Class2\_rfTrainHP, recall\_Class2\_rfTrain, recall\_Class2\_dtTrainHP, recall\_Class2\_dtTrain,f1Score\_Class2\_rfTrainHP, f1Score\_Class2\_rfTrain, f1Score\_Class2\_dtTrainHP, f1Score\_Class2\_dtTrain]);

xticklabels({'Precision RFHP','Precision RF','Precision DTHP','Precision DT','Recall RFHP','Recall RF','Recall DTHP','Recall DT','F1 Score RFHP','F1 Score RF','F1 Score DTHP','F1 Score DT'});

xtickangle(90);

title('Fog');

%Rain

subplot(2,3,3);

bar([precision\_Class3\_rfTrainHP, precision\_Class3\_rfTrain, precision\_Class3\_dtTrainHP, precision\_Class3\_dtTrain,recall\_Class3\_rfTrainHP, recall\_Class3\_rfTrain, recall\_Class3\_dtTrainHP, recall\_Class3\_dtTrain,f1Score\_Class3\_rfTrainHP, f1Score\_Class3\_rfTrain, f1Score\_Class3\_dtTrainHP, f1Score\_Class3\_dtTrain]);

xticklabels({'Precision RFHP','Precision RF','Precision DTHP','Precision DT','Recall RFHP','Recall RF','Recall DTHP','Recall DT','F1 Score RFHP','F1 Score RF','F1 Score DTHP','F1 Score DT'});

xtickangle(90);

title('Rain');

%Snow

subplot(2,3,4);

bar([precision\_Class4\_rfTrainHP, precision\_Class4\_rfTrain, precision\_Class4\_dtTrainHP, precision\_Class4\_dtTrain,recall\_Class4\_rfTrainHP, recall\_Class4\_rfTrain, recall\_Class4\_dtTrainHP, recall\_Class4\_dtTrain,f1Score\_Class4\_rfTrainHP, f1Score\_Class4\_rfTrain, f1Score\_Class4\_dtTrainHP, f1Score\_Class4\_dtTrain]);

xticklabels({'Precision RFHP','Precision RF','Precision DTHP','Precision DT','Recall RFHP','Recall RF','Recall DTHP','Recall DT','F1 Score RFHP','F1 Score RF','F1 Score DTHP','F1 Score DT'});

xtickangle(90);

title('Snow');

%Sun

subplot(2,3,5);

bar([precision\_Class5\_rfTrainHP, precision\_Class5\_rfTrain, precision\_Class5\_dtTrainHP, precision\_Class5\_dtTrain,recall\_Class5\_rfTrainHP, recall\_Class5\_rfTrain, recall\_Class5\_dtTrainHP, recall\_Class5\_dtTrain,f1Score\_Class5\_rfTrainHP, f1Score\_Class5\_rfTrain, f1Score\_Class5\_dtTrainHP, f1Score\_Class5\_dtTrain]);

xticklabels({'Precision RFHP','Precision RF','Precision DTHP','Precision DT','Recall RFHP','Recall RF','Recall DTHP','Recall DT','F1 Score RFHP','F1 Score RF','F1 Score DTHP','F1 Score DT'});

xtickangle(90);

title('Sun');

%%

AccuracyPercentage\_rfTrainHP

AccuracyPercentage\_dtTrainHP

AccuracyPercentage\_rfTrain

AccuracyPercentage\_dtTrain

%%