```
%% -- 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 colums ns, 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','R↙
ecallDTHP','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
%%
```