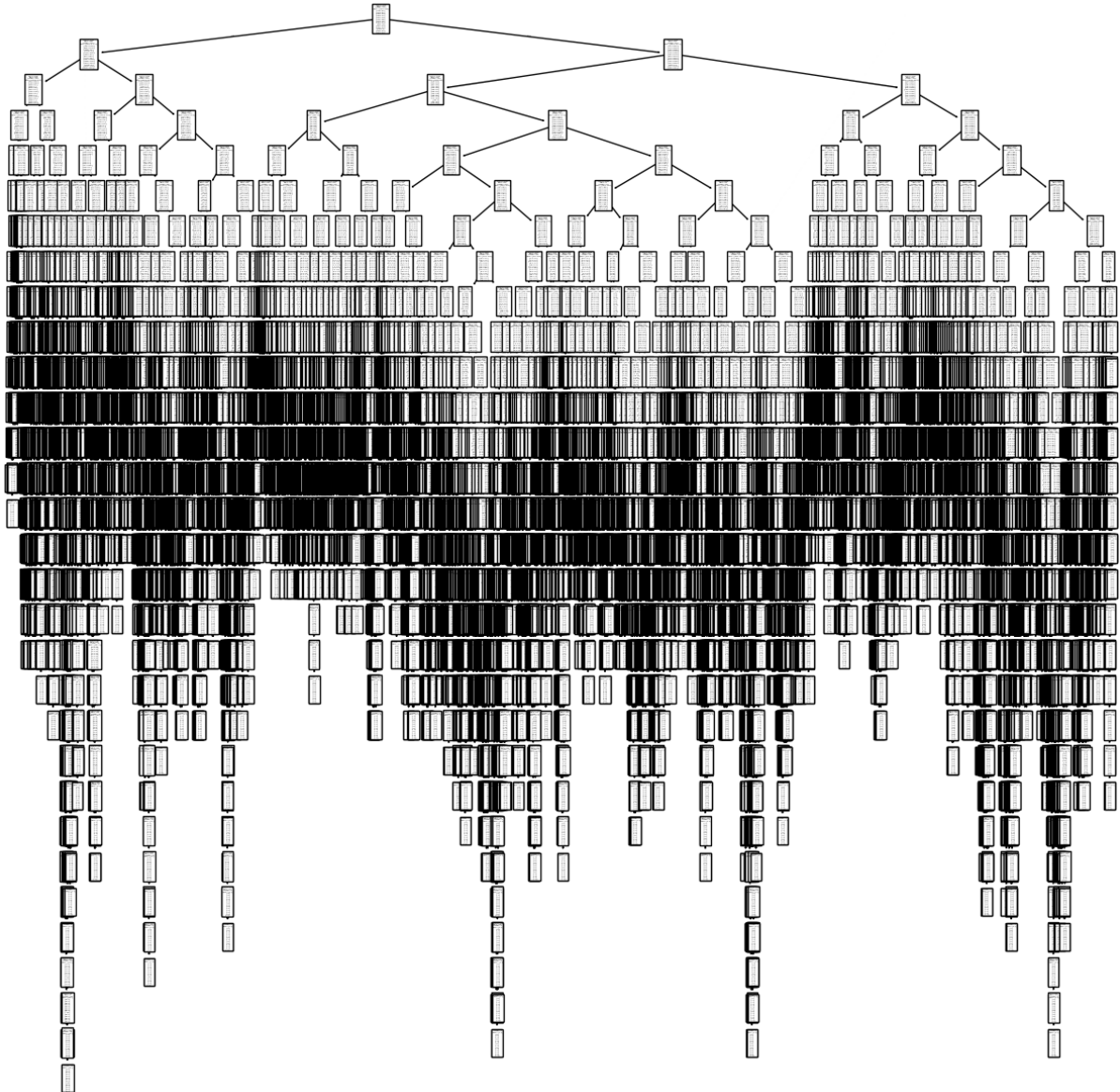


1.5: SUPERVISED LEARNING ALGORITHMS PART 2

Weather Data Decision Tree Model:



Weather Data Decision Tree Model – Parameters

```
In [ ]: #What is the testing accuracy score? Using the cross-validation method
y_pred = weather_dt.predict(X_test)
print('Test accuracy score: ', accuracy_score(y_test, y_pred))
multilabel_confusion_matrix(y_test, y_pred)
```

Test accuracy score: 0.47385848727779717

```
In [ ]: array([[3870, 468],
              [ 438, 962]],

              [[3200, 576],
               [ 530, 1432]],

              [[3397, 503],
               [ 516, 1322]],

              [[4273, 364],
               [ 347, 754]],

              ...])
```

Note: 47% test accuracy

```
[122]: #What is the training accuracy score? Using the cross validation method
y_pred_train = weather_dt.predict(X_train)
print('Train accuracy score: ', cross_val_score(weather_dt, X_train, y_train, cv = 3, scoring='accuracy').mean())
multilabel_confusion_matrix(y_train, y_pred_train)
```

Train accuracy score: 0.46153827226214633

```
[122]: array([[12948, 0],
              [ 0, 4264]],

              [[11182, 0],
               [ 0, 6030]],

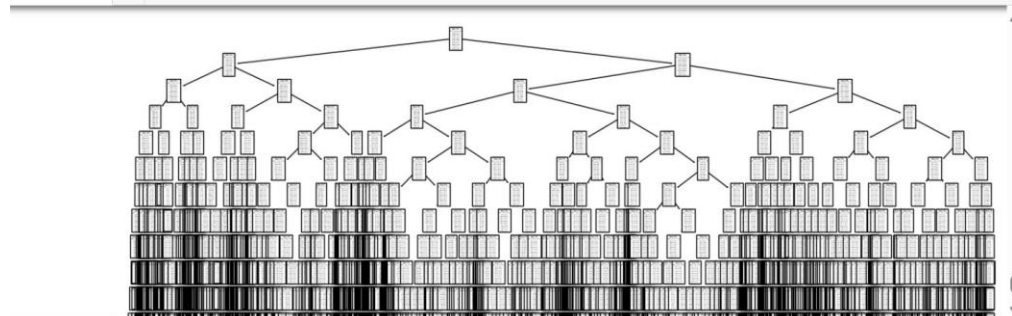
              ...])
```

Note: 46% test accuracy

Weather Data Decision Tree Model – Test Accuracy

4. Run decision tree model

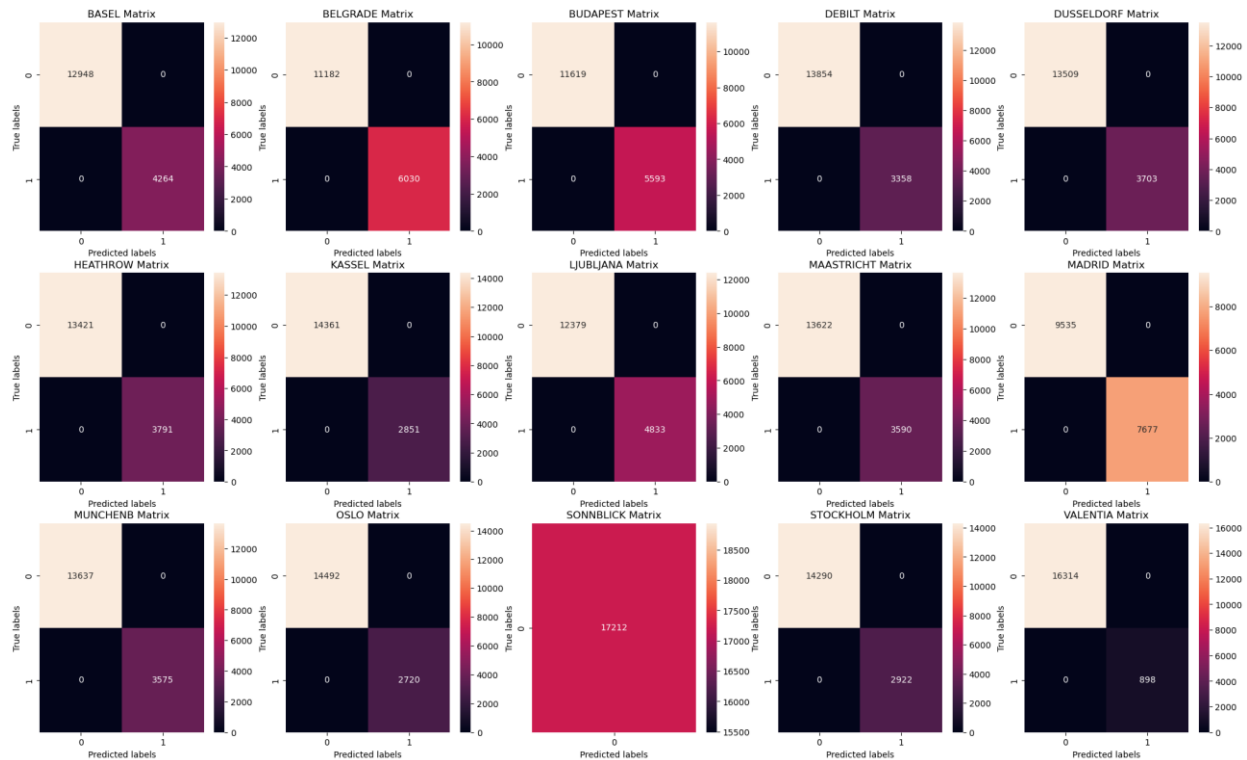
```
In [10]: 1 #Run Decision Tree classifier
2 weather_dt = DecisionTreeClassifier(criterion='gini', min_samples_split=2)
3 weather_dt.fit(X_train, y_train)
4 figure(figsize=(15,15))
5 tree.plot_tree(weather_dt)
```



Note: Needs pruning

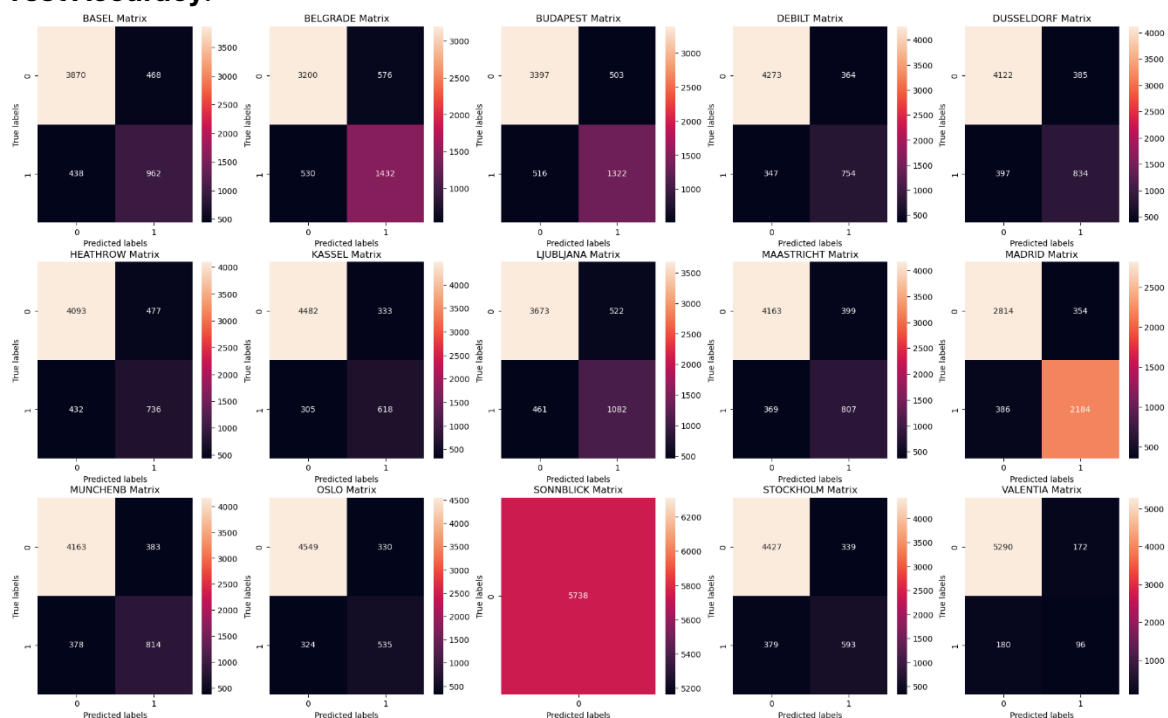
Weather Data Decision Tree Model

Training Accuracy:



Note: Visualization of ‘training’ predictions for decision tree model.

Test Accuracy:



Note: Visualization of ‘test’ predictions for decision tree model.

Weather ANN Model 1 – Parameters

```
#Create the ANN
#hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
#and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
mlp = MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500, tol=0.0001)
#Fit the data to the model
mlp.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500)
```

```
#testing ANN accuracy
y_pred = mlp.predict(X_train)
print(accuracy_score(y_pred, y_train))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_pred_test, y_test))
```

```
0.4571810364861724
0.46357615894039733
```

ANN model for y_pred_test/y_test has an accuracy score of 0.463%, the train data has an accuracy of 0.457%

Note: 46% testing accuracy

Weather ANN Model 1 – Individual Scores

```
print("Accuracy scores for each group:")
for i, accuracy in enumerate(accuracy_scores):
    print(f"Group {i + 1}: {accuracy:.4f}")
```

Accuracy scores for each group:

```
Group 1: 0.8419
Group 2: 0.8242
Group 3: 0.8336
Group 4: 0.8658
Group 5: 0.8397
Group 6: 0.8276
Group 7: 0.8747
Group 8: 0.8364
Group 9: 0.8628
Group 10: 0.8804
Group 11: 0.8628
Group 12: 0.8925
Group 13: 1.0000
Group 14: 0.8808
Group 15: 0.9517
```

Note: Group 15 (VALENTIA), 95% accuracy on this weather station

Weather ANN Model 1 - Confusion Matrix

Training Accuracy:

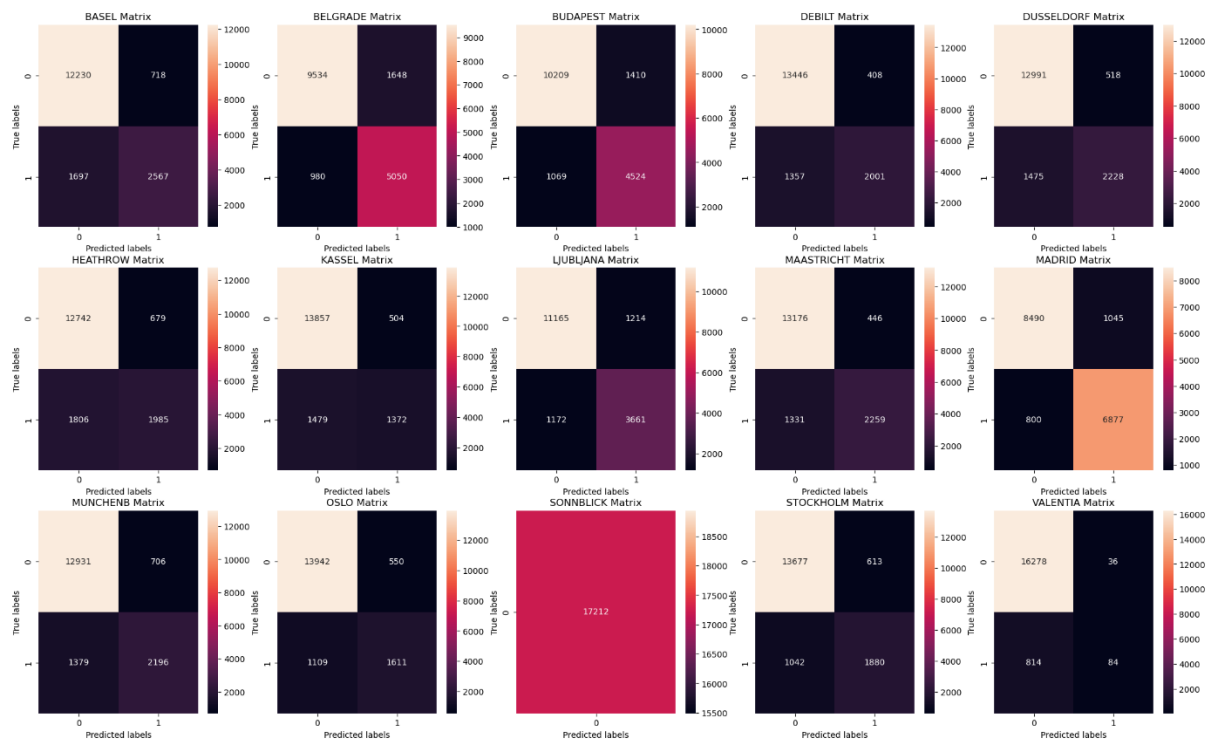


Fig1: Visualization of ‘training’ predictions for artificial neural network (ANN) model 1.

Test Accuracy:

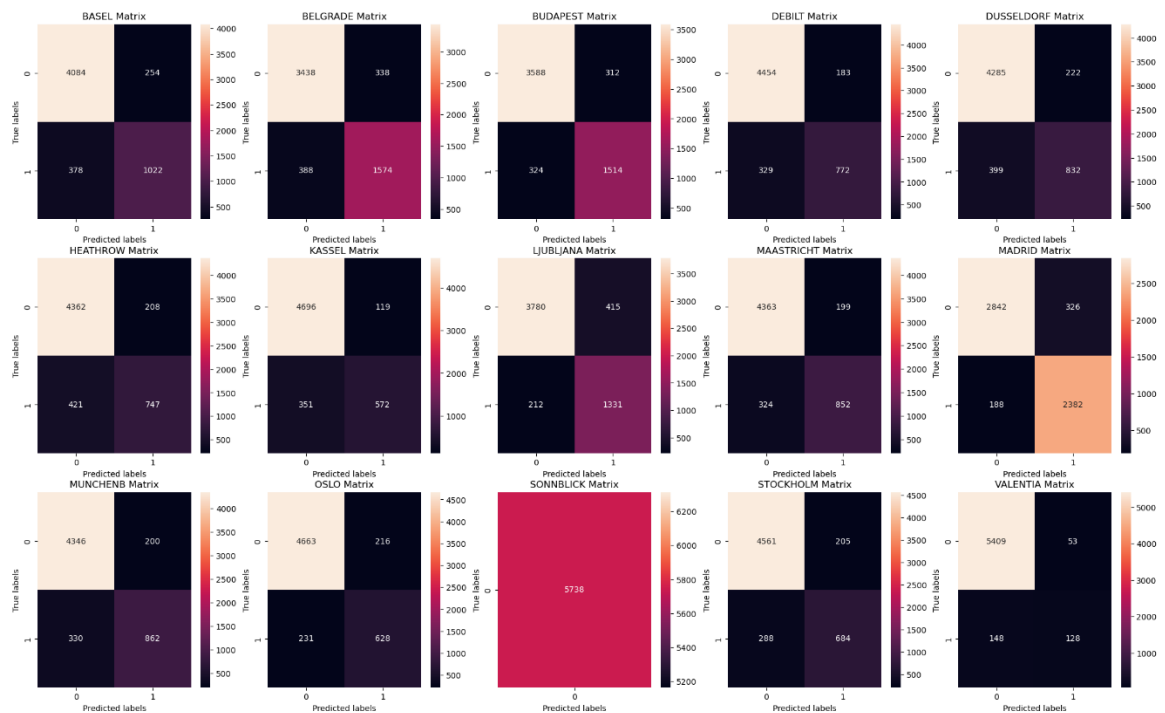


Fig2: Visualization of ‘test’ predictions for artificial neural network (ANN) model 1.

Weather ANN Model 2 – Parameters

```
.20]: #Create the ANN
      #hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
      #and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
      mlp = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=500, tol=0.0001)
      #Fit the data to the model
      mlp.fit(X_train, y_train)

.20]: ▼ MLPClassifier ⓘ ⓘ
      MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=500)

.21]: y_pred = mlp.predict(X_train)
      print(accuracy_score(y_pred, y_train))
      y_pred_test = mlp.predict(X_test)
      print(accuracy_score(y_pred_test, y_test))

      0.4699047176388566
      0.4775182990589055
```

- 47% test accuracy.

Weather ANN Model 2 – Individual Scores

```
print("Accuracy scores for each group:")
for i, accuracy in enumerate(accuracy_scores):
    print(f"Group {i + 1}: {accuracy:.4f}")
```

Accuracy scores for each group:

Group 1: 0.8484
Group 2: 0.8233
Group 3: 0.8451
Group 4: 0.8669
Group 5: 0.8390
Group 6: 0.8383
Group 7: 0.8930
Group 8: 0.8365
Group 9: 0.8655
Group 10: 0.8886
Group 11: 0.8686
Group 12: 0.8933
Group 13: 1.0000
Group 14: 0.8904
Group 15: 0.9526

Group 15 (VALENTIA), 95% accuracy on this weather station.

Weather ANN Model 2 - Confusion Matrix

Training Accuracy:

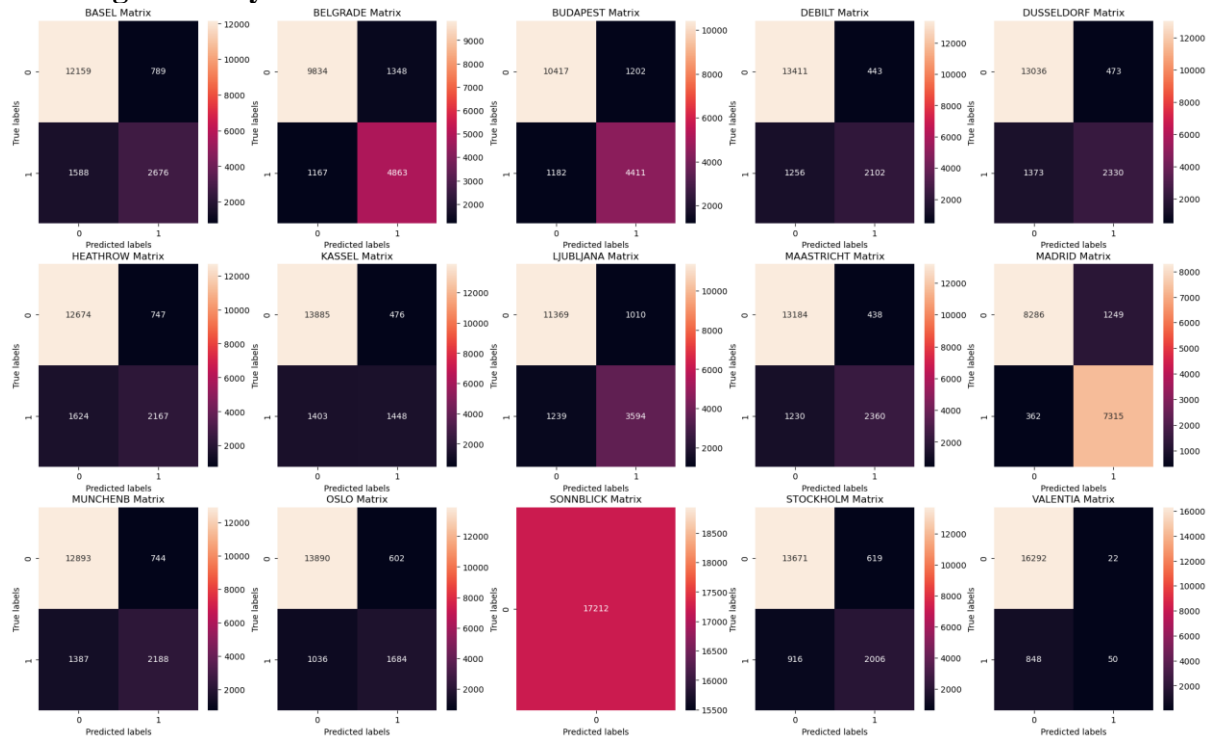


Fig3 Visualization of ‘training’ predictions for artificial neural network (ANN) model 2.

Test Accuracy:

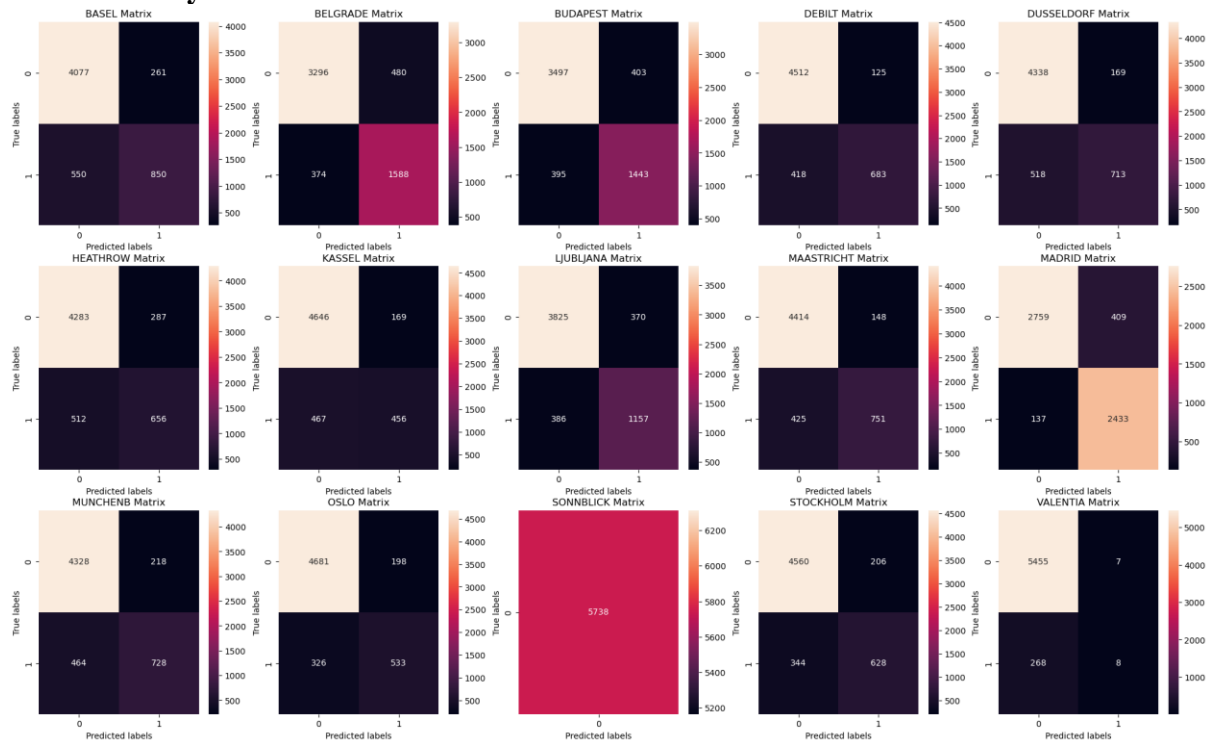
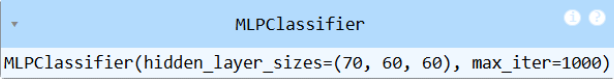


Fig4: Visualization of ‘test’ predictions for artificial neural network (ANN) model 2.

Weather ANN Model 3 – Parameters

```
]: #Create the ANN
#hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
#and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
mlp = MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000, tol=0.0001)
#Fit the data to the model
mlp.fit(X_train, y_train)

]: 
MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000)

]: y_pred = mlp.predict(X_train)
print(accuracy_score(y_pred, y_train))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_pred_test, y_test))

0.5153962351847549
0.4949459742070408
```

Note: 49% testing accuracy

Weather ANN Model 3 – Individual Scores

```
print("Accuracy scores for each group:")
for i, accuracy in enumerate(accuracy_scores):
    print(f"Group {i + 1}: {accuracy:.4f}")
```

Accuracy scores for each group:

Group 1: 0.8520
Group 2: 0.8294
Group 3: 0.8513
Group 4: 0.8717
Group 5: 0.8536
Group 6: 0.8487
Group 7: 0.9001
Group 8: 0.8526
Group 9: 0.8761
Group 10: 0.8890
Group 11: 0.8736
Group 12: 0.8996
Group 13: 1.0000
Group 14: 0.8961
Group 15: 0.9540

Note: Group 15 (VALENTIA), 95% accuracy on this weather station.

Weather ANN Model 3 - Confusion Matrix

Training Accuracy:

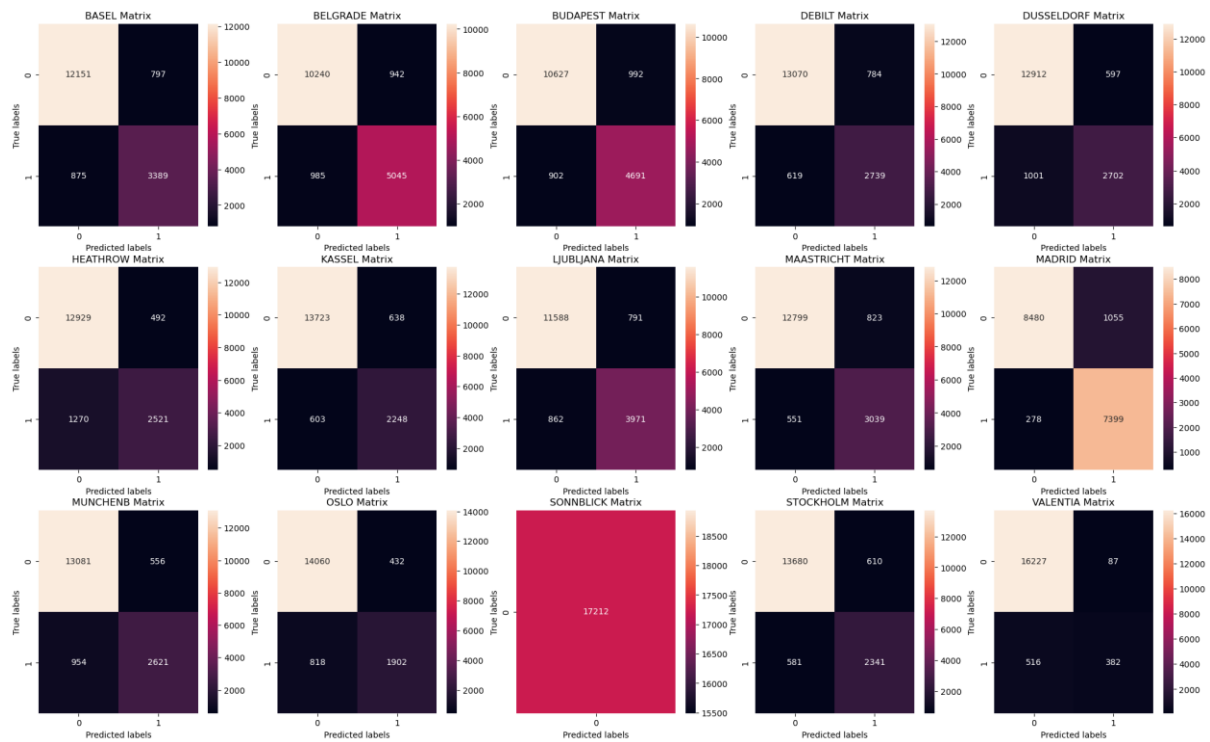


Fig5: Visualization of ‘training’ predictions for artificial neural network (ANN) model 3.

Test Accuracy:

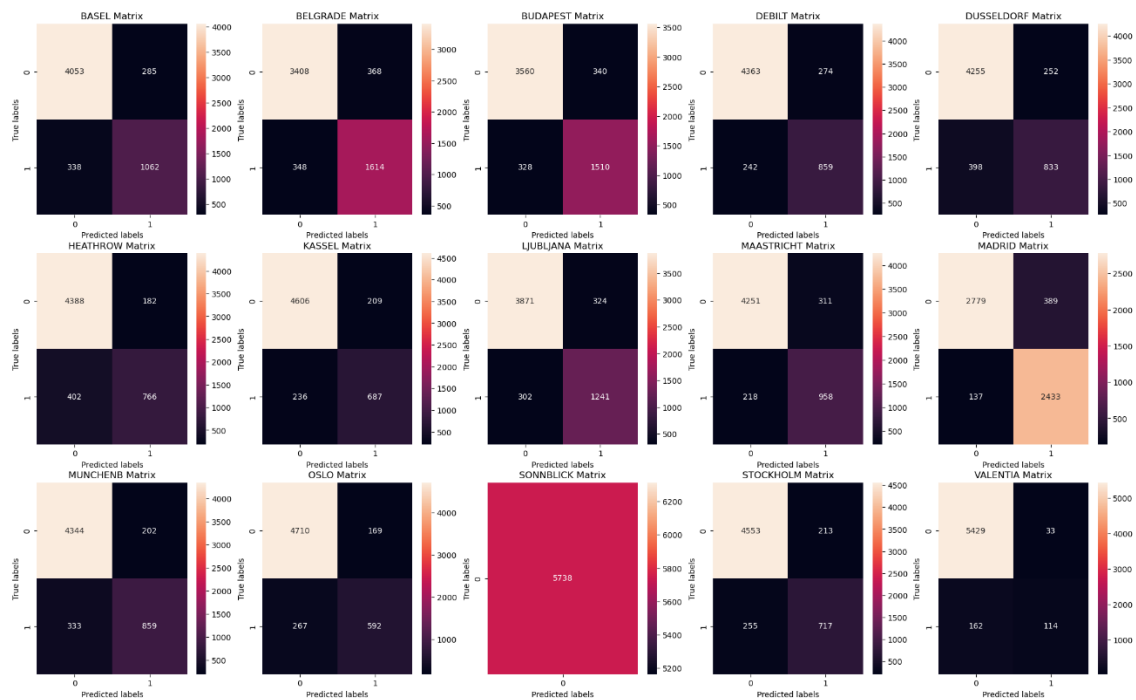


Fig6: Visualization of ‘test’ predictions for artificial neural network (ANN) model 3.

Weather ANN Model – Iterations

Starting Iterations:

```
#Create the ANN
#hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
#and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
mlp = MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500, tol=0.0001)
#Fit the data to the model
mlp.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500)
```

```
#testing ANN accuracy
y_pred = mlp.predict(X_train)
print(accuracy_score(y_pred, y_train))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_pred_test, y_test))
```

```
0.4571810364861724
0.46357615894039733
```

ANN model for `y_pred_test/y_test` has an accuracy score of 0.463%, the train data has an accuracy of 0.457%

Ending Iterations:

```
3]: # Create the ANN
mlp = MLPClassifier(hidden_layer_sizes=(30, 15, 15), max_iter=1200, tol=0.0001) #testing for plateau in data with increased parameters
#Fit the data to the model
mlp.fit(X_train, y_train)
```

```
3]: MLPClassifier
MLPClassifier(hidden_layer_sizes=(30, 15, 15), max_iter=1200)
```

```
3]: # Check accuracy
y_pred = mlp.predict(X_train)
print(accuracy_score(y_train, y_pred))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_test, y_pred_test))
multilabel_confusion_matrix(y_test, y_pred_test)
```

```
0.4973274459679293
0.4945974207040781
```

```
3]: array([[4087, 251],
         [ 394, 1006]],

         [[3416, 360],
```

Note: Based on the provided accuracy scores, it seems that increasing the complexity of the neural network leads to incremental improvements in predictive performance for classifying pleasant weather conditions, however, the overall accuracy scores are below 50% and seem to reach a plateau on model performance; making the parameters more complex may cause overfitting.

1. Which of these algorithms (including the KNN model from Exercise 1.4) do you think best predicts the current data?

Among the various algorithms tested, the K-Nearest Neighbours (KNN) model is currently the best at predicting the weather data, as it achieves the highest accuracy compared to the Decision Tree and Artificial Neural Network (ANN) models. While the ANN model shows some improvement in accuracy with increasing complexity, its performance plateaus after a certain point, with marginal improvements from 46% to 49%. On the other hand, KNN captures patterns more effectively, making it the most reliable model for predicting the current weather data. Therefore, KNN stands out as the most effective algorithm in this case.

2. Are any weather stations fully accurate? Is there any overfitting happening?

No weather station achieves perfect accuracy in predictions. However, there is evidence of overfitting at some stations, particularly with the SONNBLICK station, which consistently predicts unpleasant weather. This overfitting is likely due to a lack of variation in the data, as SONNBLICK primarily records unpleasant weather days. This imbalance results in the model achieving high accuracy for this specific condition but makes it prone to errors when faced with diverse weather conditions. Therefore, although SONNBLICK shows high accuracy, the model may not generalize well due to its overfitting, which is a sign that more balanced and varied training data is needed to improve the model's robustness.

3. Are there certain features of the dataset that might contribute to the overall accuracy?

Key features such as temperature variations, humidity, and seasonal patterns are critical in contributing to the model's overall accuracy. Weather data is inherently complex and highly variable, making prediction challenging. Temperature fluctuations and daily weather changes introduce additional uncertainty, reducing the reliability of predictive models. Since weather patterns can be unpredictable, these features may only partially correlate with actual conditions at a specific time. This variability in the data influences the accuracy of the model, as even small fluctuations in weather conditions can have significant impacts on the predictions.

4. Which model would you recommend that ClimateWins use?

I would recommend that ClimateWins use the KNN model for predicting weather conditions. KNN provides the best balance between accuracy and simplicity, making it a reliable choice for this dataset. While the ANN model might have potential with more data and further tuning, KNN is a practical model that requires less fine-tuning and still handles the complexity of weather data well. KNN's ability to base predictions on neighbouring data points makes it an effective method for capturing weather patterns without the risk of overfitting. Therefore, KNN is the most suitable choice for ClimateWins, given its dependable accuracy and suitability for the complex and varied nature of weather data.