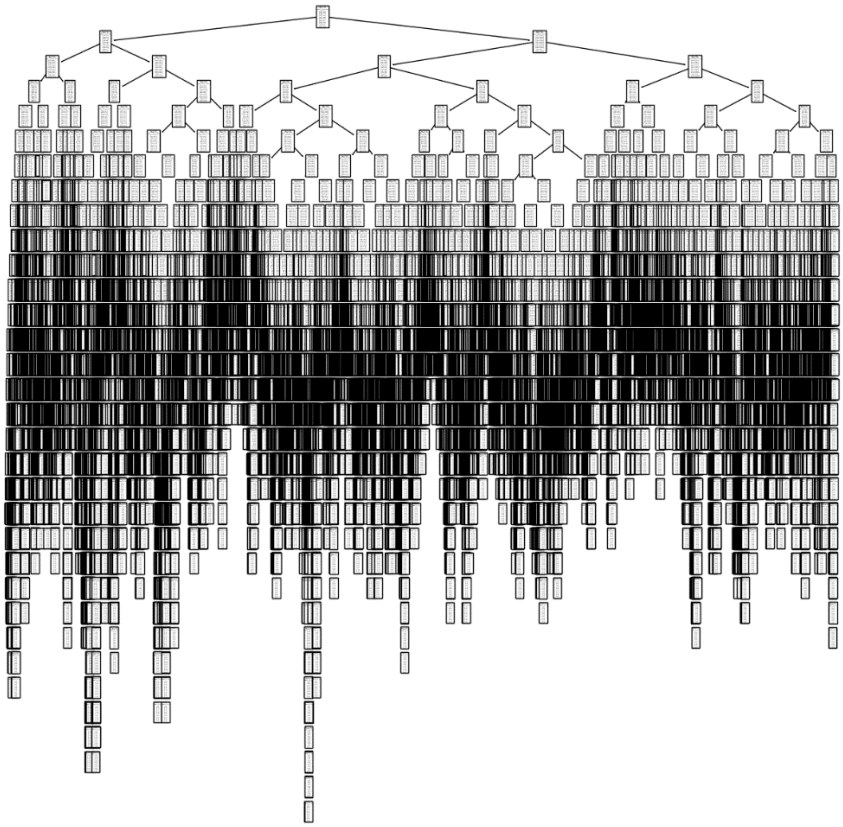
Keanu Gomes
Exercise 1.5 March 16, 2024

Basics of Machine Learning for Analysts

1.5: Supervised Learning Algorithms Part 2

Weather Data Decision Tree Model:



Weather Data Decision Tree Model - Parameters

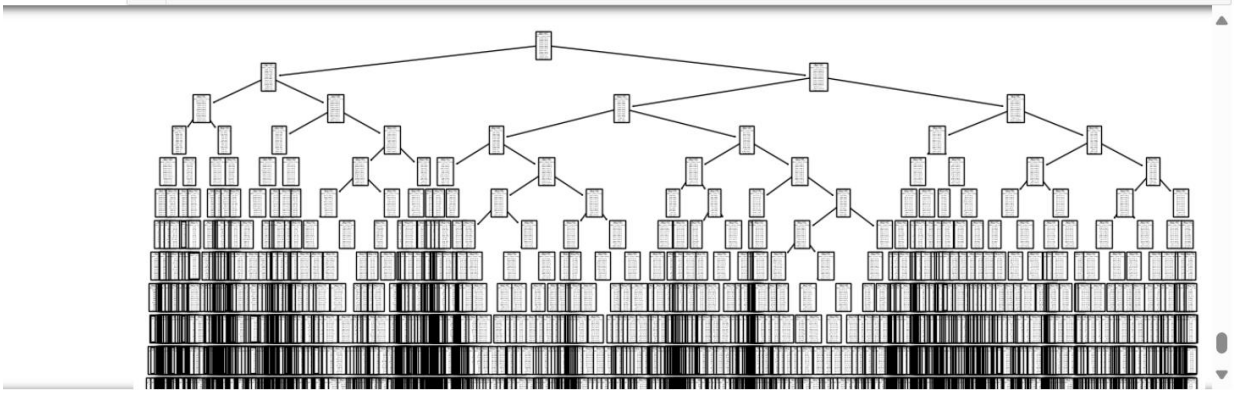
```
In [12]:
#What is the testing accuracy score? Using the cross validation method
y_pred = weather_dt.predict(X_test)
print('Test accuracy score: ', accuracy_score(y_test, y_pred))
multilabel_confusion_matrix(y_test, y_pred)

Test accuracy score:  0.4051934471941443

Out[12]: array([[[3735,  603],
                 [ 555,  845]],

                [[3143,  633],
                 [ 622, 1340]],

                [[3339,  561],
                 [ 578, 1260]],
```

- Note: *40% test accuracy*

Weather Data Decision Tree Model – Test Accuracy

4. Run decision tree model

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

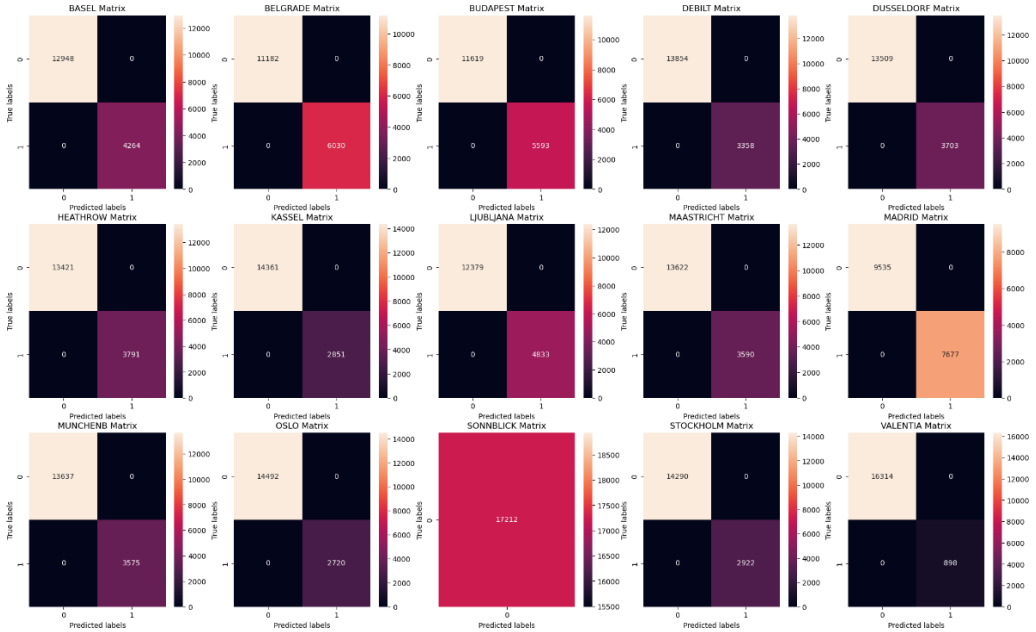


- Note: *Needs pruning*

Keanu Gomes
Exercise 1.5 March 16, 2024

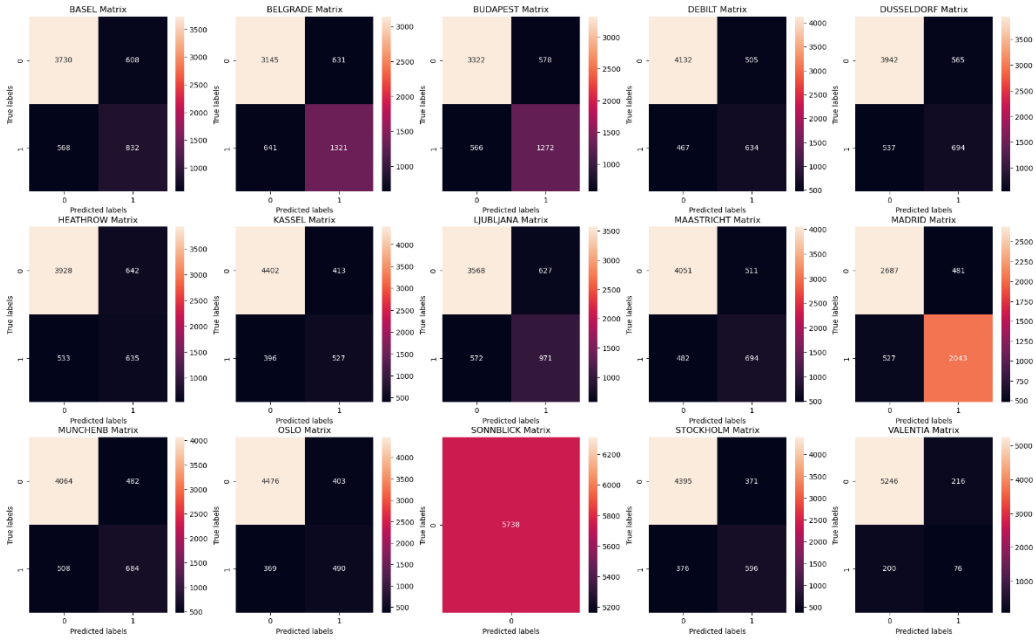
Weather Data Decision Tree Model

Training Accuracy:



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

Test Accuracy:



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

Guide for interpreting confusion matrix:

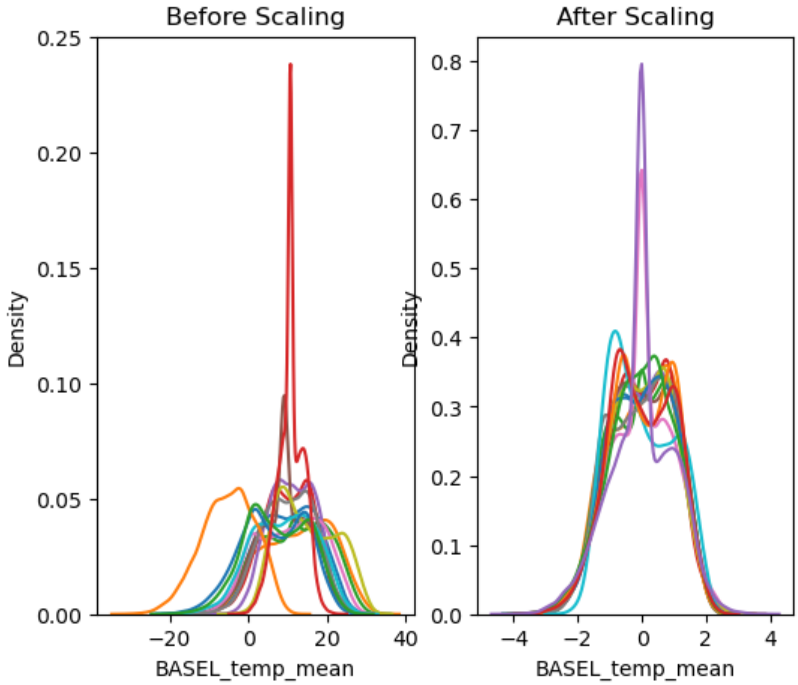
- **High TP, Low FP, Low FN:** This indicates that the model is correctly identifying most of the answers that belong to the weather station without misclassifying many others as belonging to it or misclassifying actual pleasant weather day anwers as not belonging.

- **Low TP, High FP, High FN:** This suggests that the model is struggling to correctly identify the pleasant weather days belonging to the weather station, as it is misclassifying many other pleasant weather days as belonging to it (FP) and failing to identify many actual pleasant or non-pleasant days belonging to a specific weather station (FN).

- **High TN:** If the model is correctly classifying many pleasant weather days as not belonging to the group, it may indicate that the group is well-distinguished from others in the dataset.

Weather Data - ANN MODEL

Scaled vs. Unscaled Data:



- Notes: The scaling doesn't seem to be making any significant difference between scaled and unscaled data, the unscaled data might be preferred, as it has less density than the scaled weather data. (Q: how?)

Weather ANN Model 1 - Parameters

In [22]:
```
# 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 eac
#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)
```

Out[22]:
```
MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500)
```

In [23]:
```
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))
```

```
0.4398675342784104
0.4438828860230045
```

- Note: *44*% 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}")
```

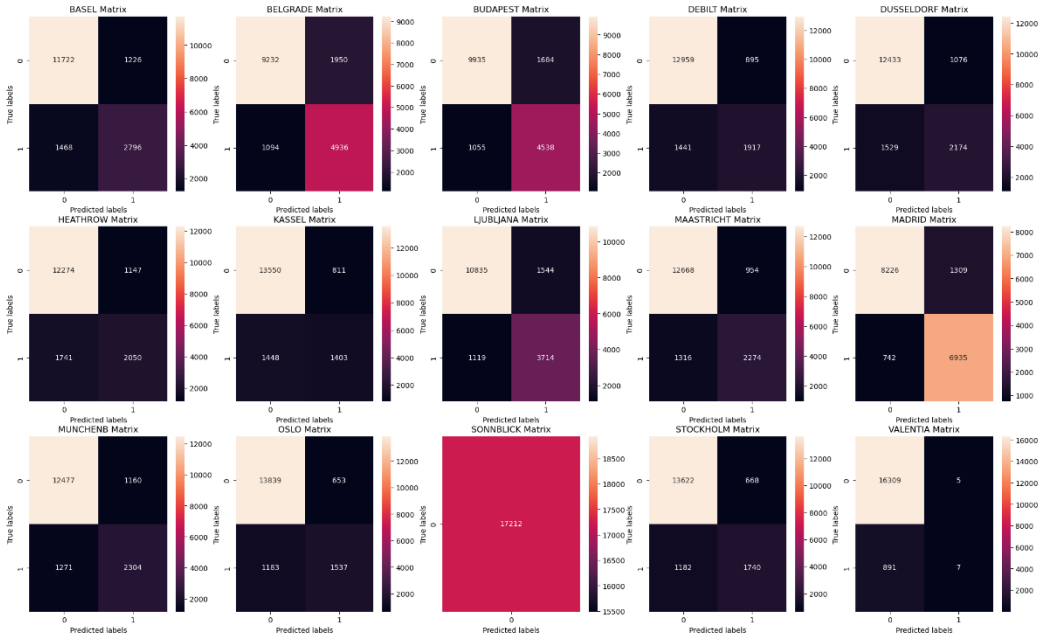
```
Accuracy scores for each group:
Group 1: 0.8419
Group 2: 0.8242
Group 3: 0.8336
Group 4: 0.8658
Group 5: 0.8397
Group 6: 0.8276
Group 7: 0.8747
Group 8: 0.8364
Group 9: 0.8628
Group 10: 0.8804
Group 11: 0.8628
Group 12: 0.8925
Group 13: 1.0000
Group 14: 0.8808
Group 15: 0.9517
```

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

Keanu Gomes
Exercise 1.5 March 16, 2024

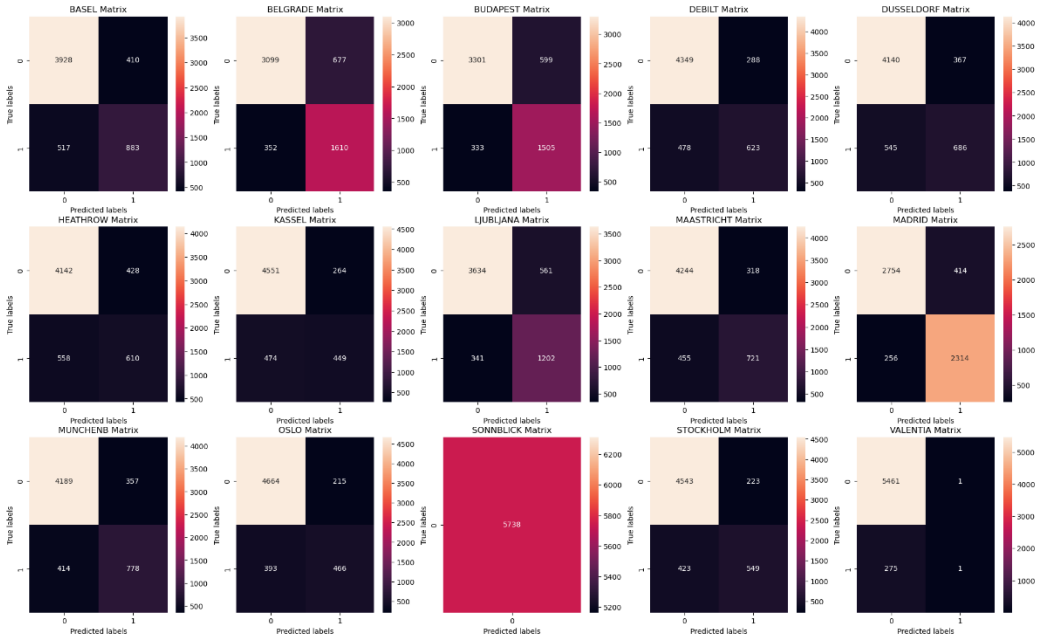
Weather ANN Model 1 - Confusion Matrix

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

In [27]:
```
# Create the ANN
mlp = MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=500, tol=0.0001) #increasing hidden layers
#Fit the data to the model
mlp.fit(X_train, y_train)
```

Out[27]:
```
MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=500)
```

In [28]:
```
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))
```

```
0.44486404833836857
0.45155106308818405
```

- *45% 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}")
```

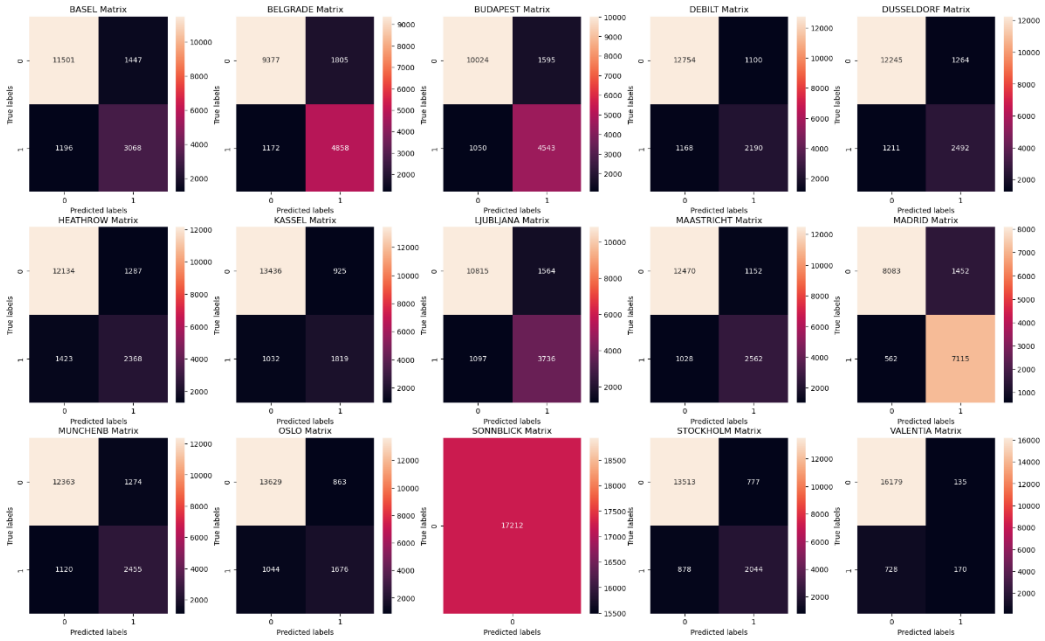
```
Accuracy scores for each group:
Group 1: 0.8484
Group 2: 0.8233
Group 3: 0.8451
Group 4: 0.8669
Group 5: 0.8390
Group 6: 0.8383
Group 7: 0.8930
Group 8: 0.8365
Group 9: 0.8655
Group 10: 0.8886
Group 11: 0.8686
Group 12: 0.8933
Group 13: 1.0000
Group 14: 0.8904
Group 15: 0.9526
```

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

Keanu Gomes
Exercise 1.5 March 16, 2024

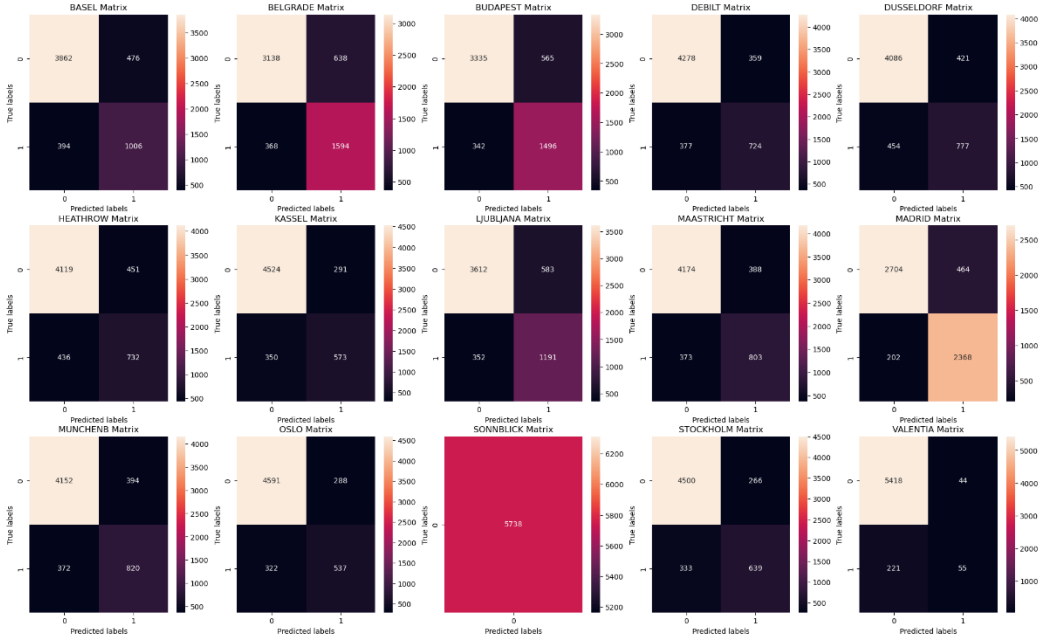
Weather ANN Model 2 - Confusion Matrix

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

In [31]:
```
# Create the ANN
mlp = MLPClassifier(hidden_layer_sizes=(20, 10, 10), max_iter=1000, tol=0.0001) #increasing hidden layers and iterations
#Fit the data to the model
mlp.fit(X_train, y_train)
```

Out[31]:
```
MLPClassifier(hidden_layer_sizes=(20, 10, 10), max_iter=1000)
```

In [32]:
```
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))
```

```
0.45044155240529865
0.4527710003485535
```

- Note: 45% 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}")
```

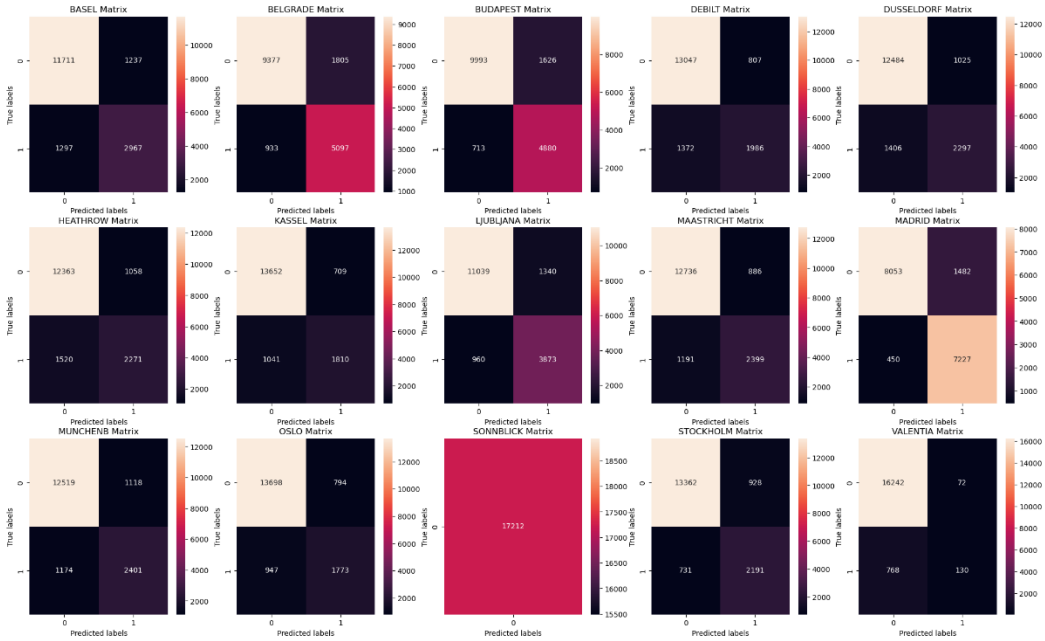
```
Accuracy scores for each group:
Group 1: 0.8520
Group 2: 0.8294
Group 3: 0.8513
Group 4: 0.8717
Group 5: 0.8536
Group 6: 0.8487
Group 7: 0.9001
Group 8: 0.8526
Group 9: 0.8761
Group 10: 0.8890
Group 11: 0.8736
Group 12: 0.8996
Group 13: 1.0000
Group 14: 0.8961
Group 15: 0.9540
```

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

Keanu Gomes
Exercise 1.5 March 16, 2024

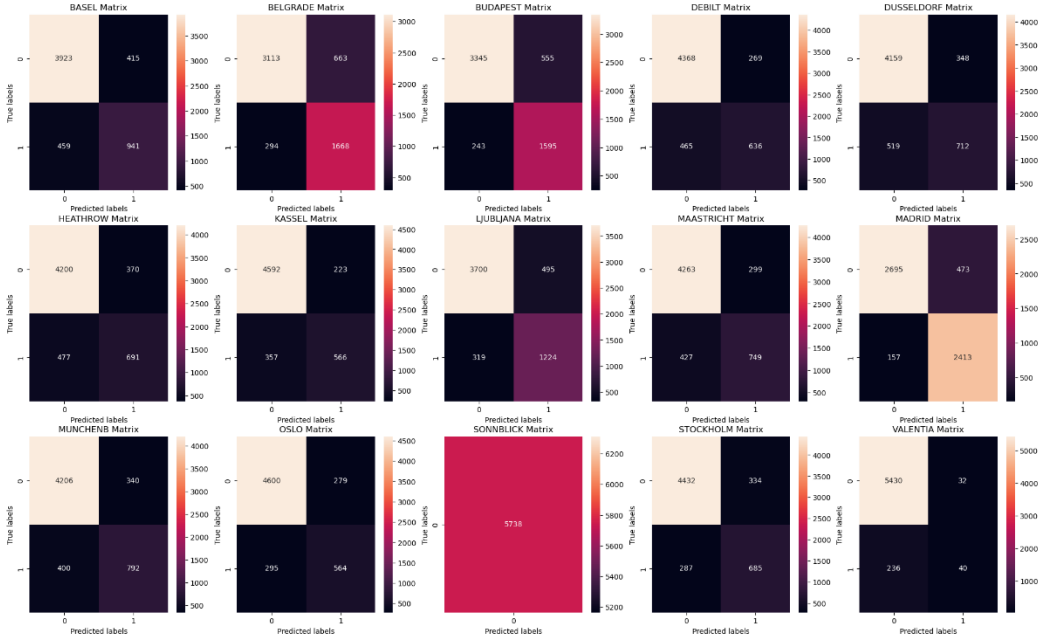
Weather ANN Model 3 - Confusion Matrix

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:

In [22]:
```
# 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 eac
#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)
```

Out[22]:
```
MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500)
```

In [23]:
```
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))
```

```
0.4398675342784104
0.4438828860230045
```

Ending Iterations:

In [35]:
```
# Create the ANN
mlp = MLPClassifier(hidden_layer_sizes=(30, 15, 15), max_iter=1200, tol=0.0001) #testing for plateau in data with increas
#Fit the data to the model
mlp.fit(X_train, y_train)
```

Out[35]:
```
MLPClassifier(hidden_layer_sizes=(30, 15, 15), max_iter=1200)
```

In [36]:
```
# Check 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))
```

```
0.45578666046943994
0.4550365981178111
```

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.

Ending Iterations – Individual weather station scores:

```
Accuracy scores for each group:
Group 1: 0.8550
Group 2: 0.8459
Group 3: 0.8660
Group 4: 0.8716
Group 5: 0.8513
Group 6: 0.8459
Group 7: 0.8996
Group 8: 0.8625
Group 9: 0.8752
Group 10: 0.8900
Group 11: 0.8759
Group 12: 0.9069
Group 13: 1.0000
Group 14: 0.9019
Group 15: 0.9510
```

NOTE: Overall, the model has a low accuracy score avg.

- Among the three models, ANN Model 3 performs the best in terms of accuracy.
  - Increasing the complexity of the model architecture (adding more hidden layers and neurons) tends to improve performance, as seen from the trend of increasing accuracy scores from Model 1 to Model 3.
- When the model achieves similar scores with slight improvement, it suggests that the model is approaching a plateau in performance.
- In terms of individual weather stations, the model performs the best with VALENTIA (Group 15), as it has the highest accuracy score of all weather stations not accounting for SONNBLICK(Group 10), as there is only 'unpleasant' weather days for that weather station.

- **Final Note:** Overall, the model has a low accuracy score avg.

- Among the three models, ANN Model 3 performs the best in terms of accuracy.

  - Increasing the complexity of the model architecture (adding more hidden layers and neurons) tends to improve performance, as seen from the trend of increasing accuracy scores from Model 1 to Model 3.

- When the model achieves similar scores with slight improvement, it suggests that the model is approaching a plateau in performance.

- In terms of individual weather stations, the model performs the best with VALENTIA (Group 15), as it has the highest accuracy score of all weather stations not accounting for SONNBLICK(Group 10), as there is only 'unpleasant' weather days for that weather station.

- Among the supervised algorithms used for this weather data set, (Gradient Decent, KNN, Decision Trees, ANN), I would recommend using either gradient decent or ANN, as it is so far the most optimal choice for converging successful predictions.