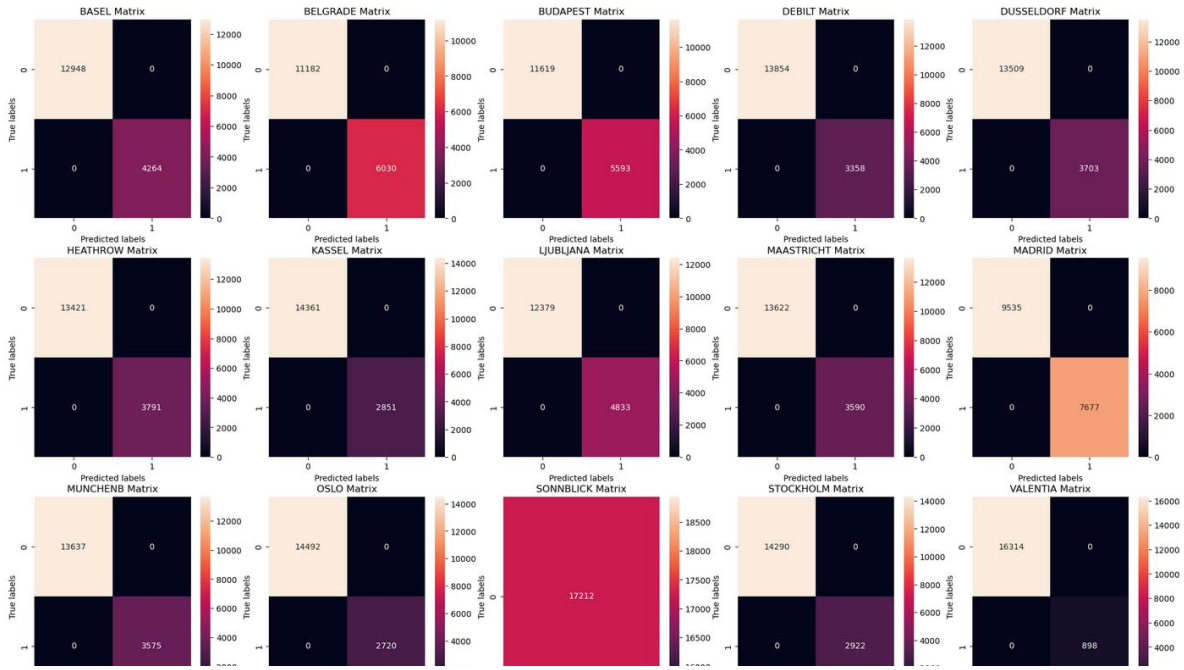
Decision tree

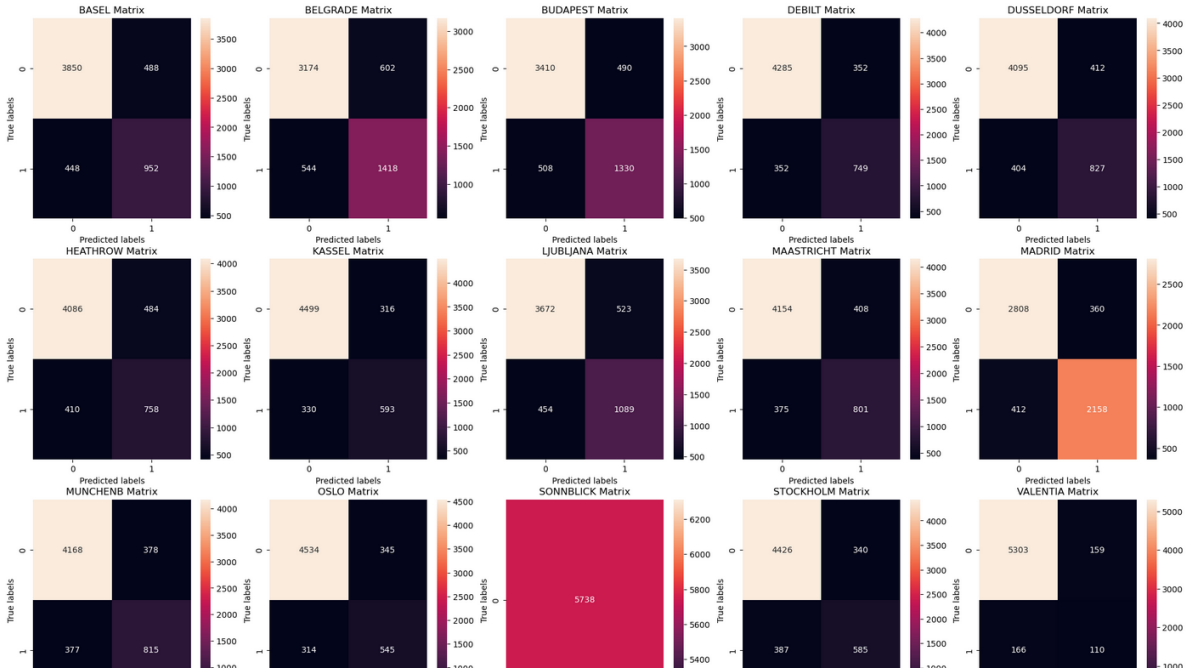
Cross-validated train accuracy: 46%

Test Data accuracy: 47%

Training data confusion matrix



Test data confusion matrix



Neural Network Model

2 layers with 5,5 nodes each

Max iteration: 1000

Tolerance: 0.0001

Accuracy for training data: 47%

Accuracy for testing data: 47%

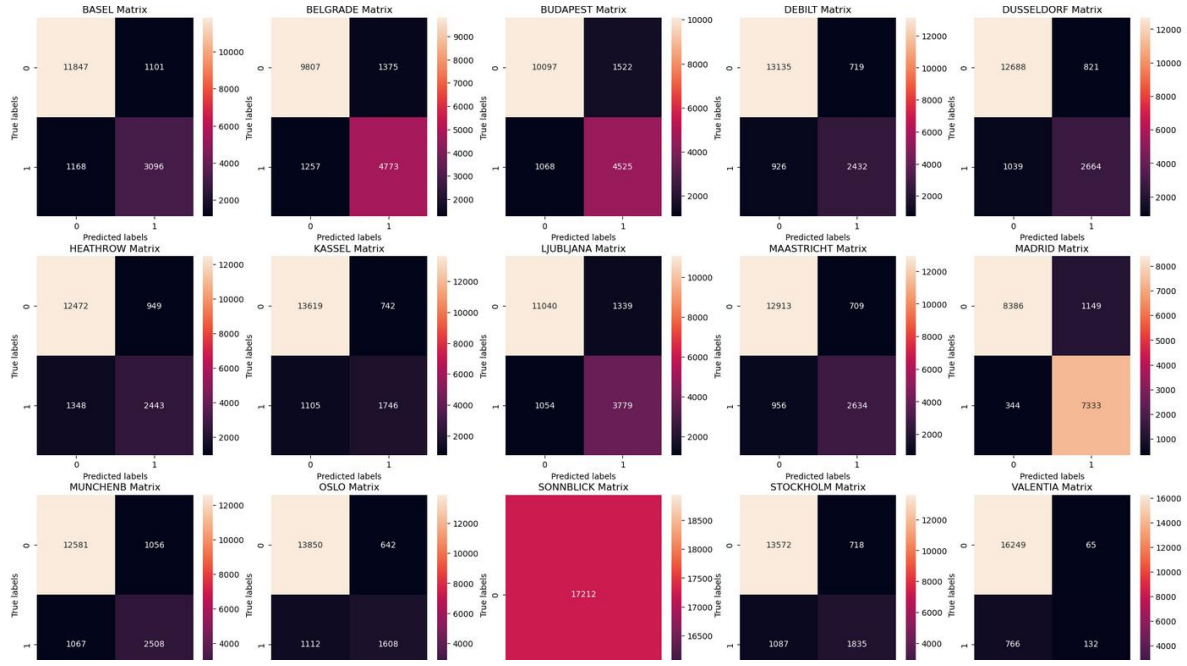
```
#Create the ANN
#hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
#and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
mlp = MLPClassifier(hidden_layer_sizes=(5, 5), max_iter=500, tol=0.0001)
#Fit the data to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

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

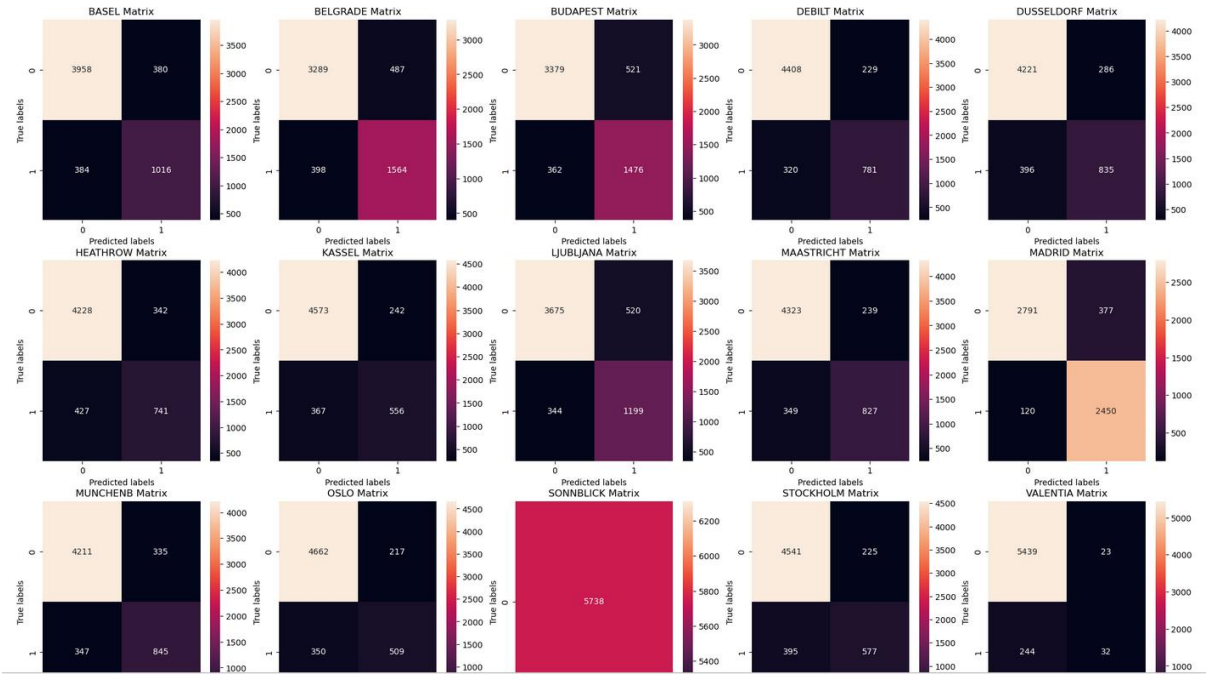
```
#testing ANN accuracy #ANN MODEL
y_pred = mlp.predict(X_train)
print(accuracy_score(y_pred, y_train))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_pred_test, y_test))
```

```
0.46967232163606787
0.4721157197629836
```

Training data confusion matrix



Test data confusion matrix



3 layers with 70, 60, 60 nodes each

Max iteration: 1000

Tolerance: 0.0001

Accuracy for training data: 58%

Accuracy for testing data: 51%

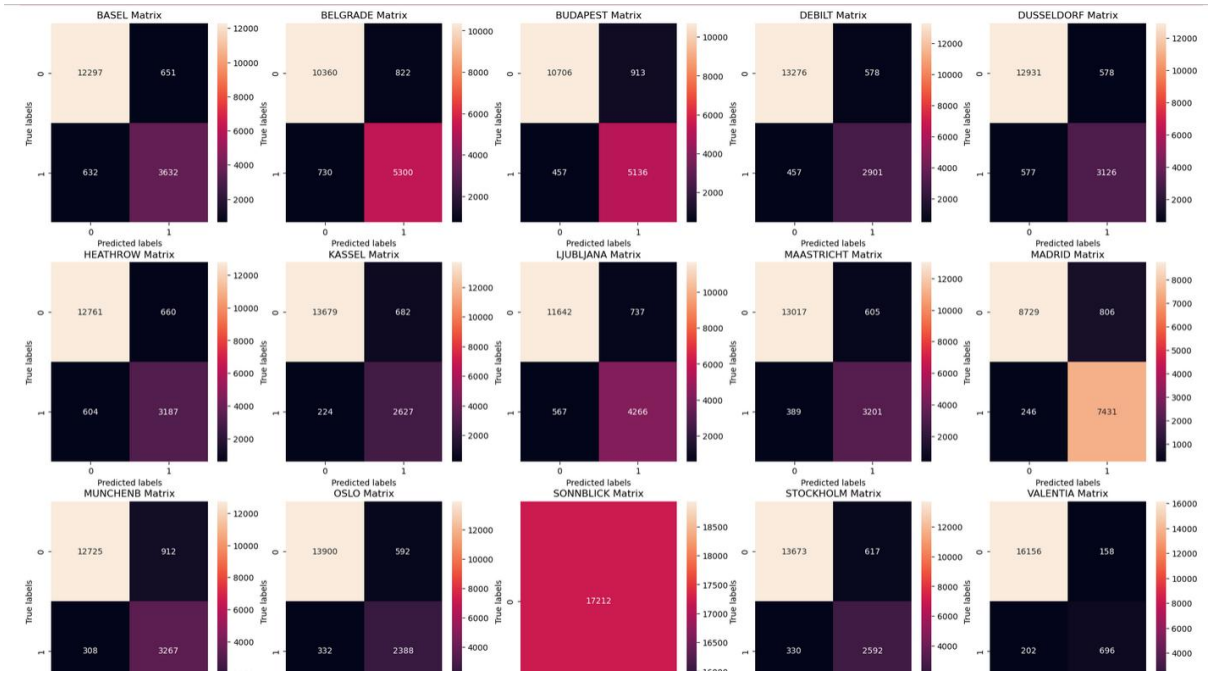
[39]:
```
#Create the ANN
#hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
#and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
mlp = MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000, tol=0.0001)
#Fit the data to the model
mlp.fit(X_train, y_train)
```

[39]:
```
MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000)
```

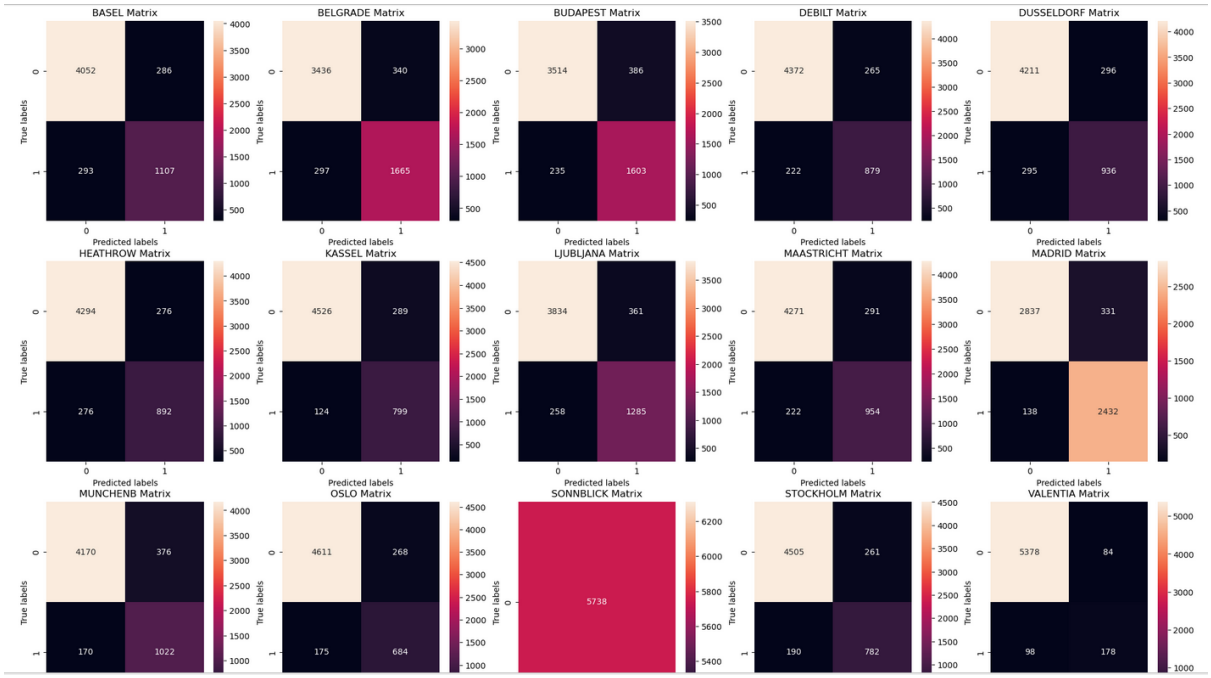
[41]:
```
y_pred = mlp.predict(X_train)
print(accuracy_score(y_pred, y_train))
y_pred_test = mlp.predict(X_test)
print(accuracy_score(y_pred_test, y_test))
```

```
0.5844178480130142
0.5172533983966539
```

Training data confusion matrix



Test data decision matrix



3 layers with 70, 60, 60 nodes each

Max iteration: 1000

Tolerance: 0.0003

Accuracy for training data: 56%

Accuracy for testing data: 50%

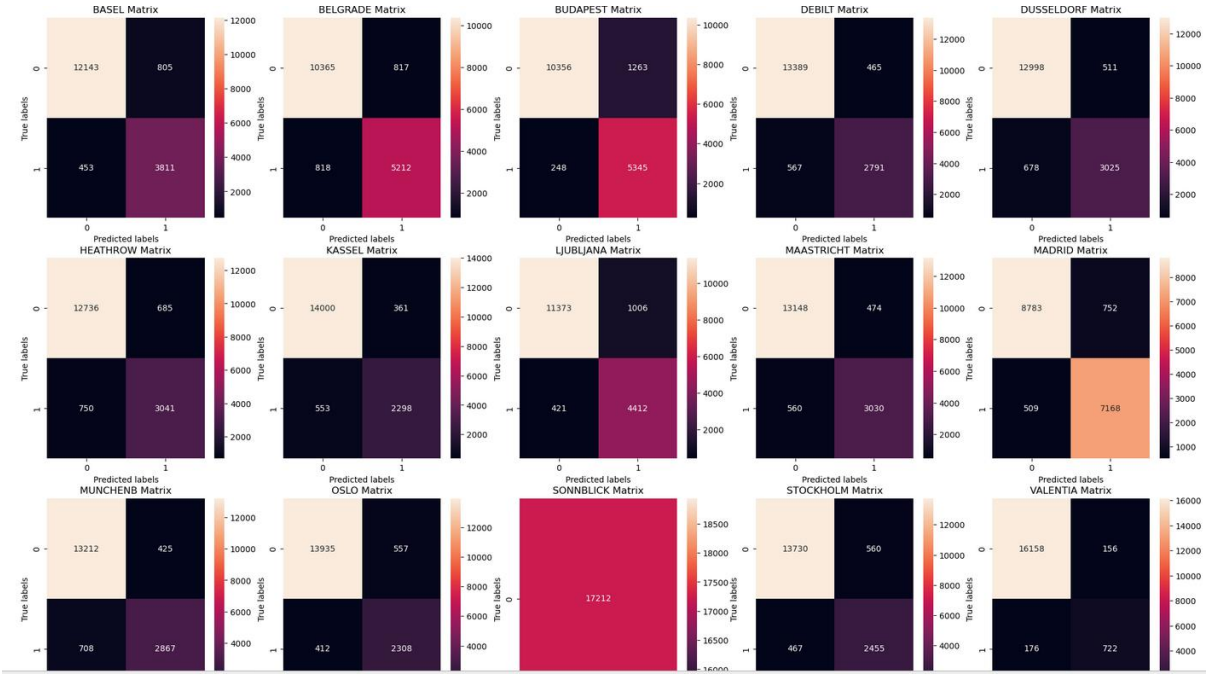
```
[47]: #Create the ANN
      #hidden_layer_sizes has up to three layers, each with a number of nodes. So (5, 5) is two hidden layers with 5 nodes each,
      #and (100, 50, 25) is three hidden layers with 100, 50, and 25 nodes.
      mlp = MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000, tol=0.0003)
      #Fit the data to the model
      mlp.fit(X_train, y_train)
```

```
[47]: MLPClassifier
      MLPClassifier(hidden_layer_sizes=(70, 60, 60), max_iter=1000, tol=0.0003)
```

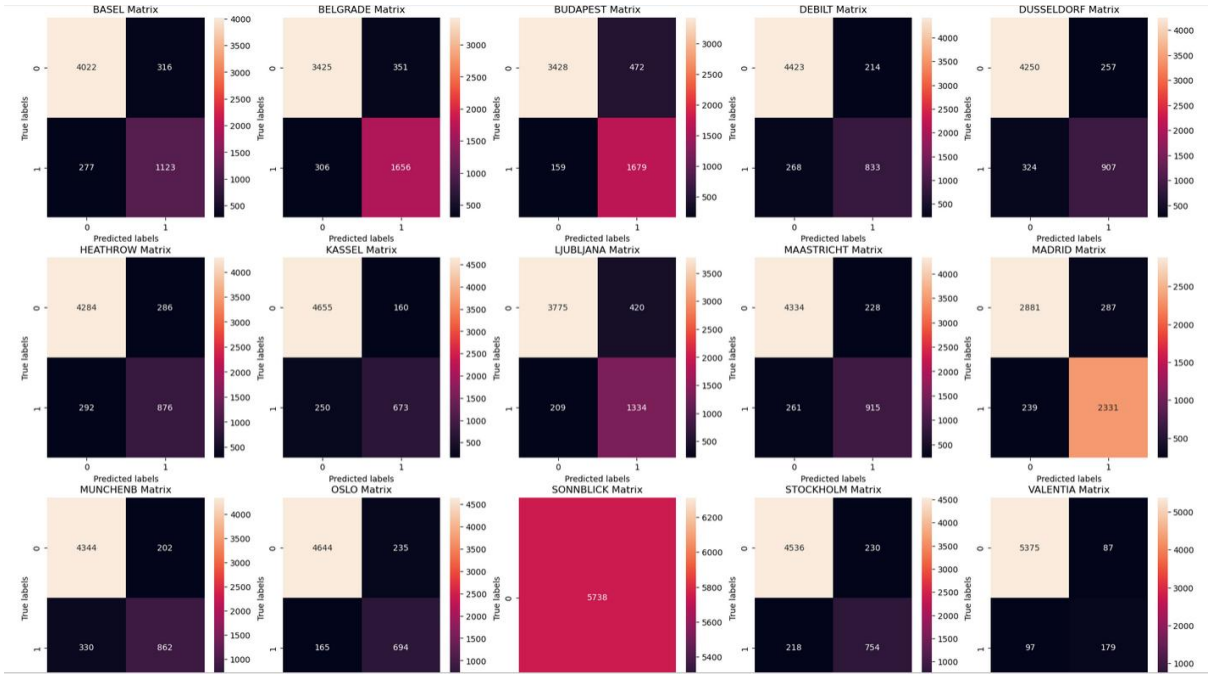
```
[49]: y_pred = mlp.predict(X_train)
      print(accuracy_score(y_pred, y_train))
      y_pred_test = mlp.predict(X_test)
      print(accuracy_score(y_pred_test, y_test))
```

```
0.5592028817104345
0.5026141512722203
```

Training data confusion matrix



Test data confusion matrix



The KNN model achieved the highest accuracy rate at 88%. However, this result may be overstated, as some cities had much higher accuracy rates than others, suggesting inconsistent model performance across locations. For this reason, I recommend using the Artificial Neural Network Model, which tends to learn more effectively from historical data and can be better tuned with human supervision to achieve higher accuracy. Additionally, the perfect accuracy observed at Sonnblick for all models (where every prediction was correct) may indicate overfitting to a station with little variability in weather outcomes. This overfitting could prevent the model from generalizing well to new data or locations with more diverse weather patterns. Finally, unpredictable features in the dataset, such as temperature and daily weather, also impact the overall accuracy of these algorithms.