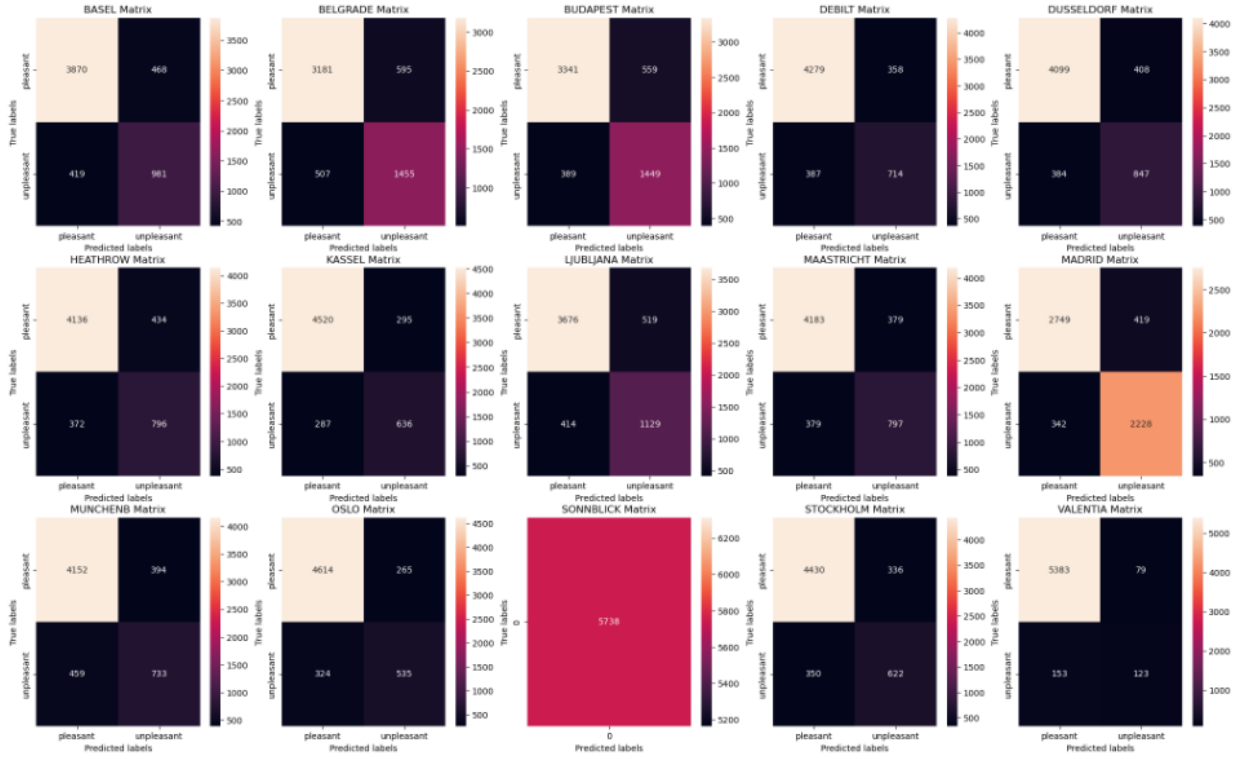
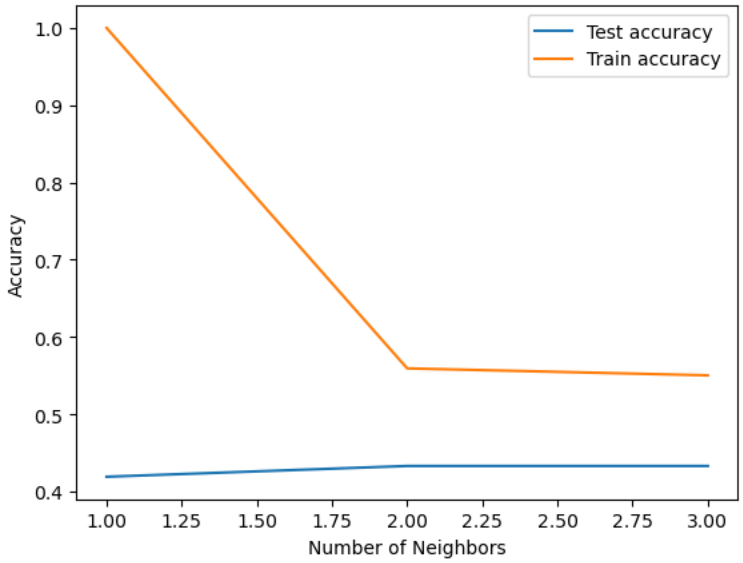
K-Nearest Neighbors: Reflections on Model Performance

Iteration 1: 4 neighbor KNN

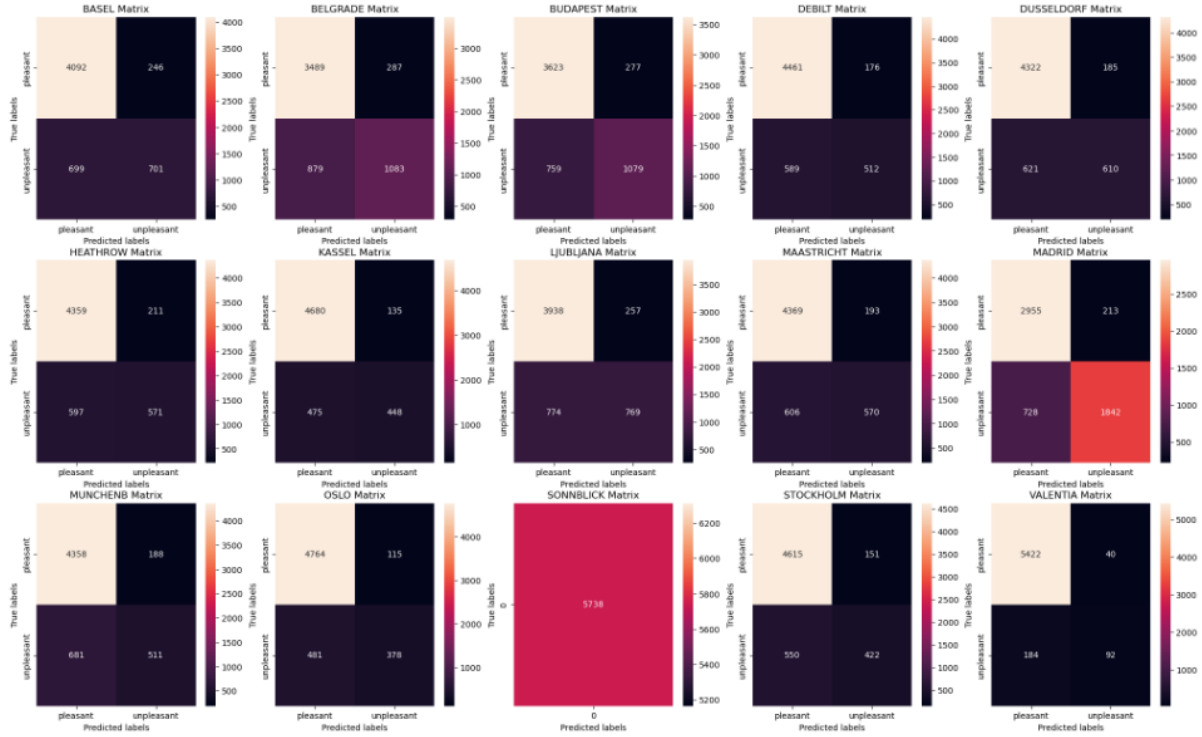


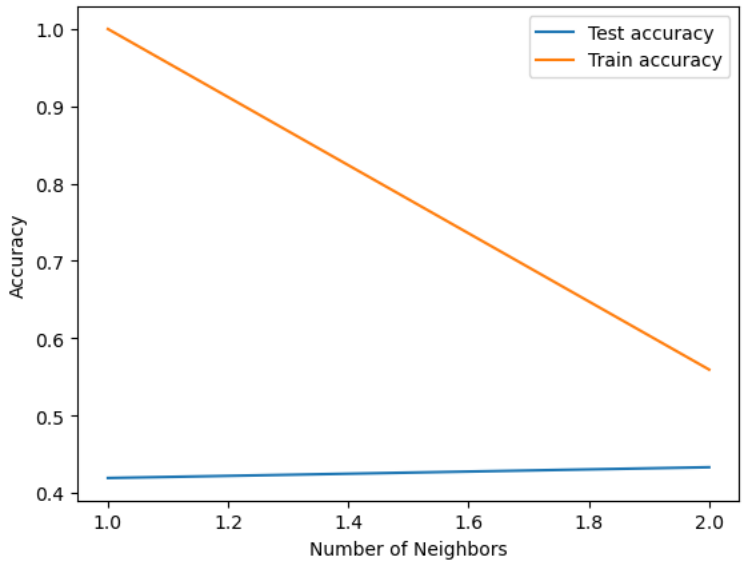


The top picture was the confusion matrix for the first iteration of KNN, and the second one was the resulting graph showing the accuracy of the train and test data. In the first iteration,

we can see the accuracy for both datasets was quite low. After 1 neighbor, the accuracy plummets for the training data, suggesting that the model was overfitting (it knows the dataset too well and is capturing more noise with the signal). Approaching 4 neighbors, the accuracy for the training data was about **55%** and the test data about **42%**. Looking at the **Sonnblick** weather station, we can see a problem: it shows that there were no pleasant days, and all 5738 days of collecting were unpleasant. This can hurt the model's performance since it fails to capture any variation in the pleasant weather at this station. **Valentia** also shows a high imbalance, with a lot more unpleasant than pleasant days, and for the same reasons as Sonnblick, this can affect model performance. Conversely, **Basel, Belgrade, Budapest, Kassel, Heathrow, and Maastricht** have a much more balanced makeup, and for this reason, the model performs much better and has higher accuracy with these stations. Lastly, **Madrid** and **Stockholm** had weaker performances showing a high number of misclassifications when it comes to predicting pleasant days. This can negatively affect overall model performance.

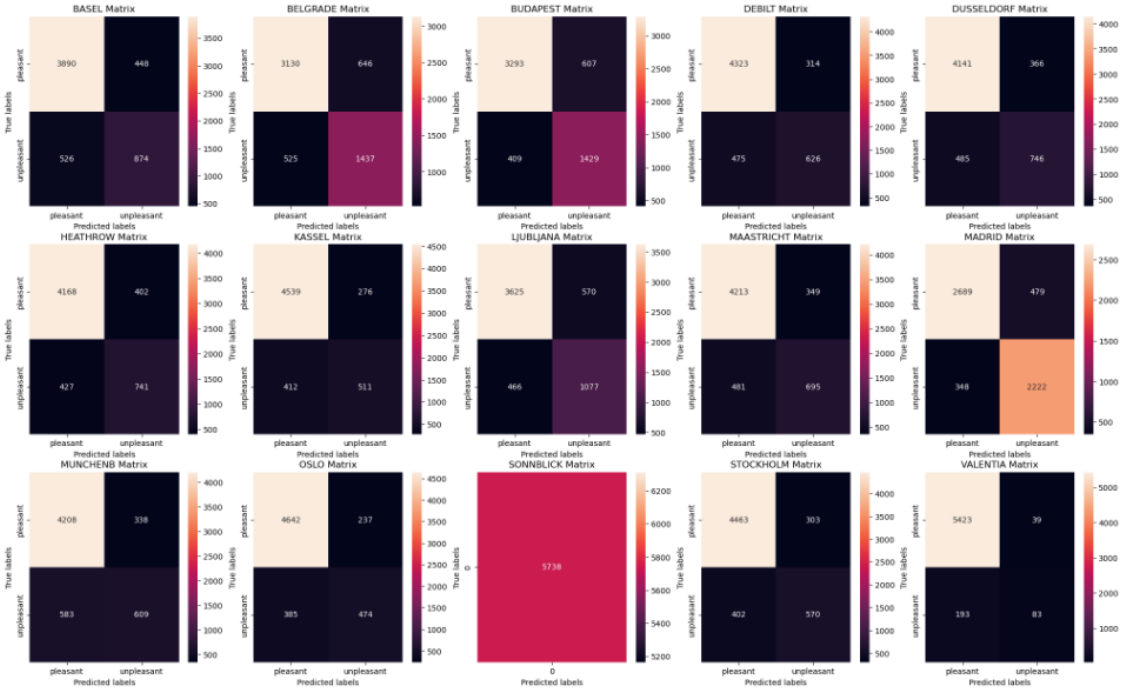
Iteration #2: 3 neighbor KNN

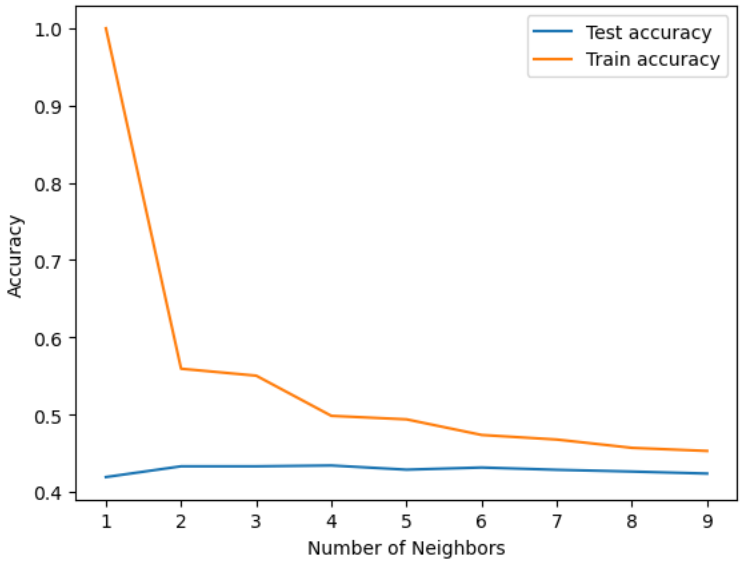




As with four neighbors, this one showed similar results, with the same weather stations behaving similarly. Once again, **Sonnblick** and **Valentia** are highly imbalanced, which negatively affects model performance as it fails to capture much variation in pleasant weather. **Basel, Budapest, Belgrade, Kassel, Heathrow, and Maastricht** show balanced predictions with higher accuracy, meaning they help improve the model's overall performance. **Madrid, Stockholm, and Oslo** show less accurate predictions, specifically when trying to predict pleasant weather days. Also, the graph of the train vs test data shows similar patterns to the first iteration, with the test accuracy stabilizing around **42%**, and the train accuracy plummeting to **55%.**

Iteration #3: 10 neighbor KNN





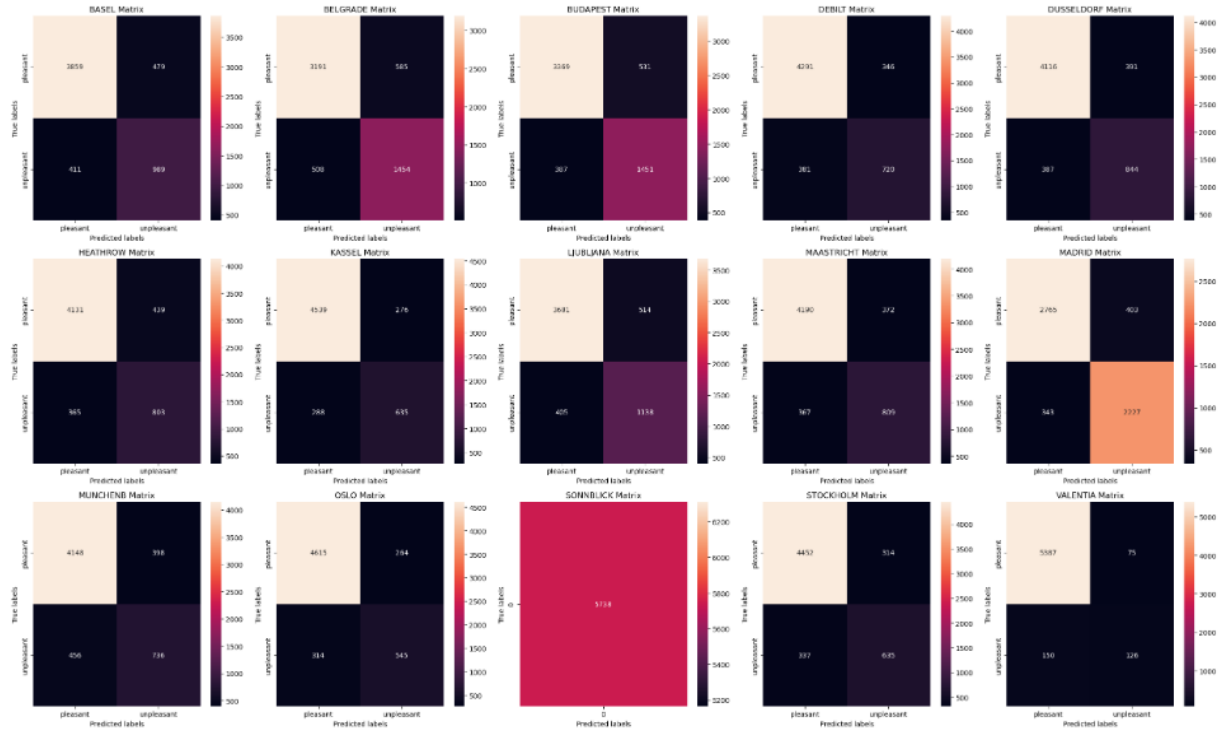
Lastly, I bumped the KNN to 10 neighbors, just to see what happens when increasing the number of neighbors. As it turns out, this performed slightly worse than the other two; while the test accuracy remained about the same, around **42%,** the training accuracy continued to trend downwards as the number of neighbors increased, eventually stabilizing around **47%**. This tells us that as the number of neighbors increases, underfitting increases, which worsens the model's performance. However, accuracy is just one metric to evaluate a model's overall performance. To improve granularity and possibly garner more

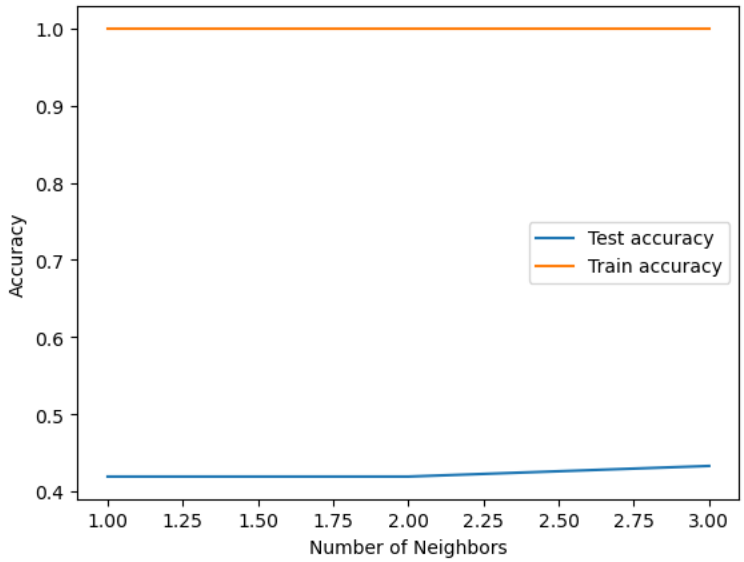
insights, I will dive into the other available metrics, namely recall, precision, and F1 score, to further understand and build a higher-resolution picture of what's going on.

Why is there a big variation in how well the stations predict pleasant weather?

Overall, there's no significant improvement with 3, 4, or 10 neighbors; the same stations are consistently underperforming, and the same stations are performing well. This could be due to the inherent imbalance of the data; with so many unpleasant days at certain stations, the model simply predicts those days more often. We saw this with **Sonnblick** and **Valentia**. On the other hand, in the stations with a better data balance, we see more accurate predictions (**Basel, Belgrade, Budapest, Kassel**). With that being said, under-sampling or oversampling techniques could be applied to minority or majority classes to see if the model's performance improves. Another option is trying a weighted KNN, which in theory could help mitigate the imbalance and improve the model's performance.

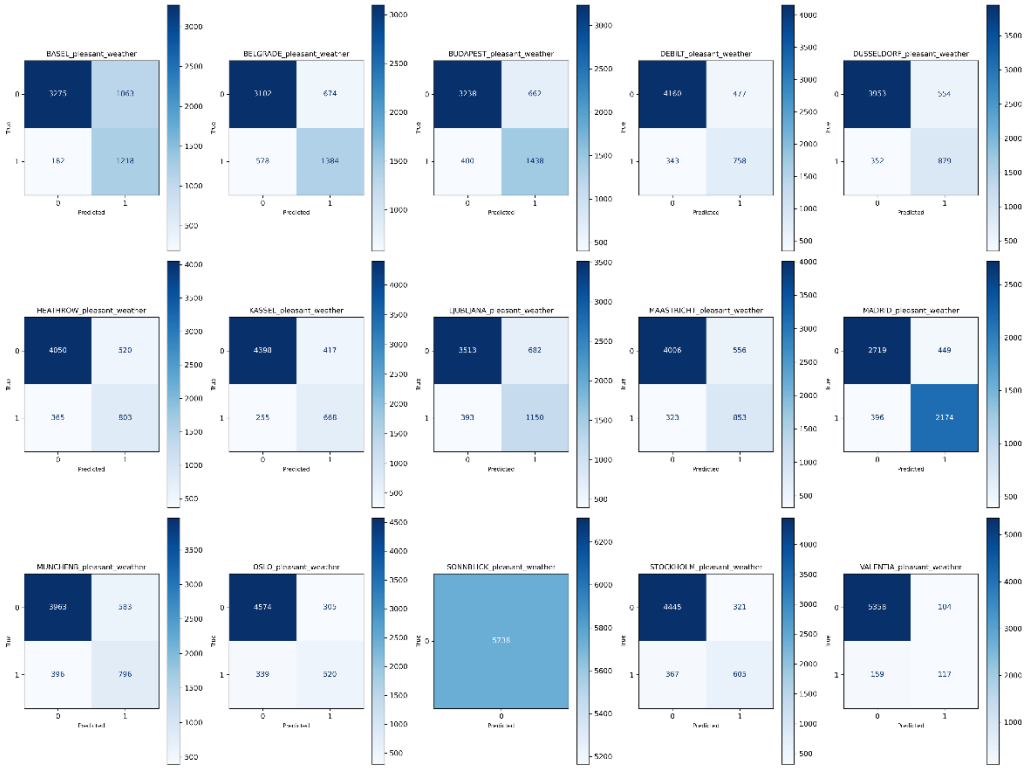
Bonus: Weighted KNN with 4 neighbors

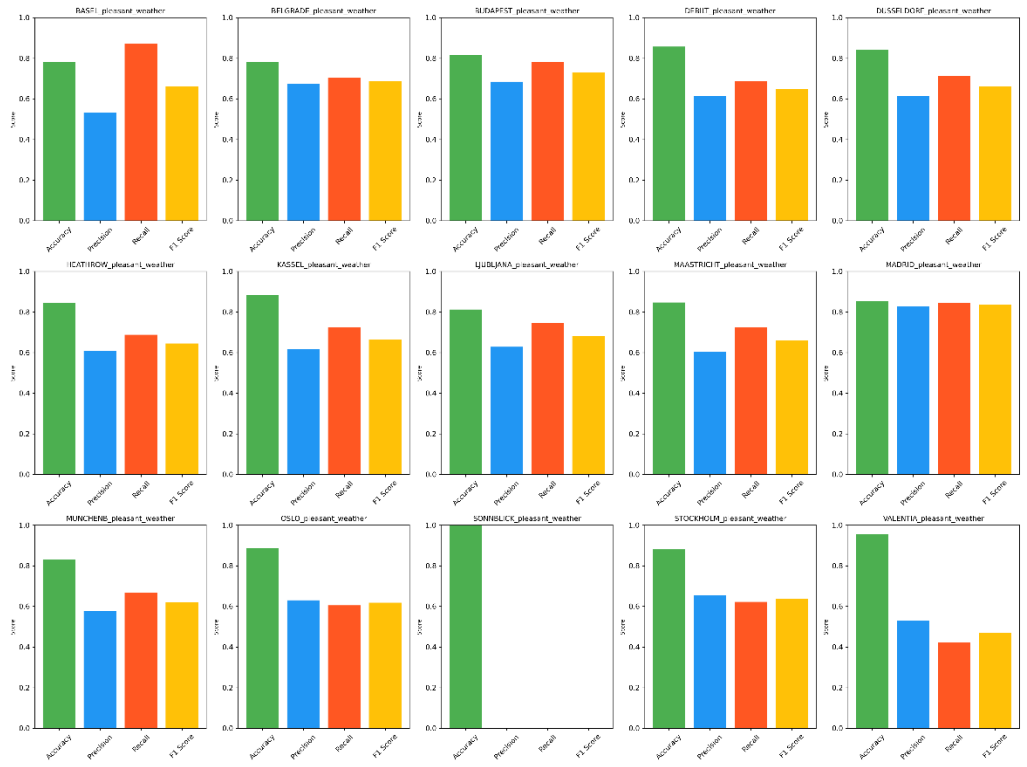




Looking at the test vs train accuracy, the only difference the weighted KNN has given us is that the train accuracy remains at 100% throughout, while the test accuracy remains quite low. Once again, this is reflective of overfitting: the model knows the train data too well and struggles to generalize to the test data, resulting in poor performance. While there is a slight uptick in accuracy, it is not enough for me to conclude that trying more neighbors in a weighted KNN will result in a model that is highly accurate in both the training and test data. Also, the same weather stations appear to be performing well (**Basel, Belgrade, Budapest, Kassel**), while the same ones continue to struggle with accuracy and imbalance (**Sonnblick, Valentia, Madrid, Stockholm and Oslo**). After having tried three different KNN iterations and then a weighted KNN, I think it is time to move on to other model types.

Bonus #2: Undersampling technique with 4 neighbors

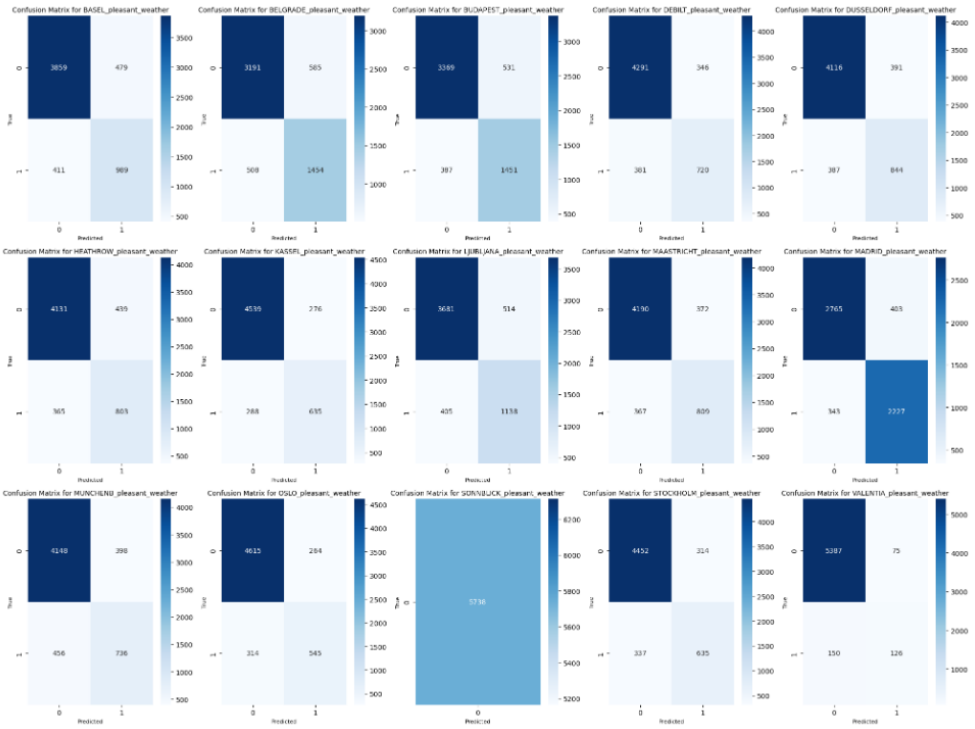




The above images show the results for the undersampling technique, with each station having **accuracy, precision, recall, and F1 scores** charted for comparison. Precision tells us how accurate the pleasant weather day predictions are; if it's low, many **false positives** occur. If the recall is low, then a lot of **false negatives** occur; conversely, if high, it tells us that the model is correctly identifying most of the pleasant weather days. The F1 score is a composite metric of precision and recall; in other words, it takes both false positives and false negatives into account. It is clear that the **Madrid** and **Budapest** stations are performing well overall, with high precision and recall noted. Madrid, specifically, is the best-performing station out of all of them. **Sonnblick** is still imbalanced- it has perfect accuracy, but that's because every day is an unpleasant one. **Valentia** is still imbalanced, and has both low precision and recall, meaning it has a lot of false negatives and false positives. Many of the other stations are either better in recall or precision, but not both. They all show an F1 score between **0.6** and **0.7**.

Performance comparison to previous models: The undersampling technique has shown mild to moderate improvements in model performance. Both precision and recall have improved for most stations, showing that the model is better at avoiding false positives and false negatives as well. Because of that, F1 scores are better as well. Accuracy remains high but because of the undersampling technique applied, overfitting has been reduced slightly, since the model was forced to train on a more balanced dataset. However, extreme cases like Sonnblick and Valentia still give the model issues.

Bonus #3: Adjusting class weights; 4 neighbors



Using a distance-based weighting technique, this is the resultant confusion matrix for the stations. Some improvements were noted, with **Madrid, Budapest, and Belgrade** showing the best performance across the key metrics (they had a balanced precision and recall). **Valentia** and **MunchenB** had lower F1 scores due to imbalanced recall and precision, and **Sonnblick** continues to be an outlier with it having no pleasant days to predict. Next, I will tune the number of neighbors used and the distance metrics as well.

Bonus #4: Tuning the # of neighbors and adjusting the distance weight

I applied a code to find the best distance metric combined with the ideal number of neighbors based on overall model performance. It came back with the **Manhattan** distance metric and **six neighbors**. However, these are the performance metrics:

Accuracy: 0.4456

Precision: 0.7360

Recall: 0.7236

F1 Score: 0.7270

This is a slight improvement over previous attempts, however, the accuracy is still quite low.

Bonus #5: Further tuning of hyperparameters

I ran a hyperparameter tuning code to see if I could further improve the performance of the KNN. It found that the **Manhattan metric, 6 neighbors, and distance weighting** came up with the best performance. However, as you can see, the test accuracy was still quite low:

Accuracy: 0.446

Precision: 0.736

Recall: 0.724

F1 Score: 0.727

These numbers are essentially the same as the previous iteration. After a few iterations of finding the best number of neighbors, adjusting the distance weighting, and trying to find the best distance metric, the performance of the model still is underwhelming when it comes to overall accuracy.