Career Foundry Data Analytics Program

Machine Learning Specialization

Luis Gil

Task 2.3 - Complex Machine Learning Models and Keras Part 2

1. Data Preparation:

- The notebook starts by loading and preparing the dataset related to climate and weather data. Key features include temperature, humidity, wind speed, and other weather variables. These variables are used to predict whether the weather is "pleasant" or not for different weather stations.
- Data cleaning steps include handling missing values and ensuring proper data types for numerical and categorical columns.

2. Feature Selection:

- **Target Variable**: The goal is to predict "pleasant weather," which is encoded in the dataset as a binary variable (0 or 1).
- **Features**: A variety of weather-related features (e.g., temperature, humidity, wind speed) are selected for the model. Based on feature importance analysis later in the process, temperature-related metrics turn out to be highly predictive.

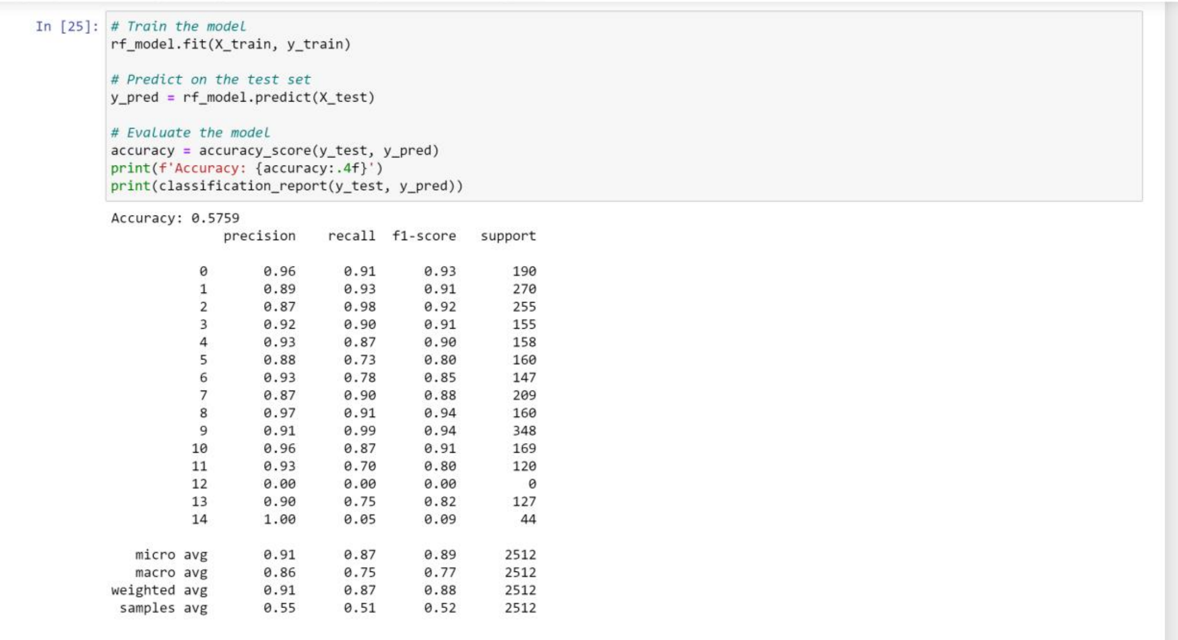
3. Modeling:

- **Model Used**: A **Random Forest Classifier** is applied to build the model. The random forest algorithm is chosen for its robustness and ability to handle a large number of features without significant overfitting.
- **Training and Testing Split**: The dataset is split into training and testing sets, ensuring that the model is evaluated on unseen data.

4. Model Evaluation:

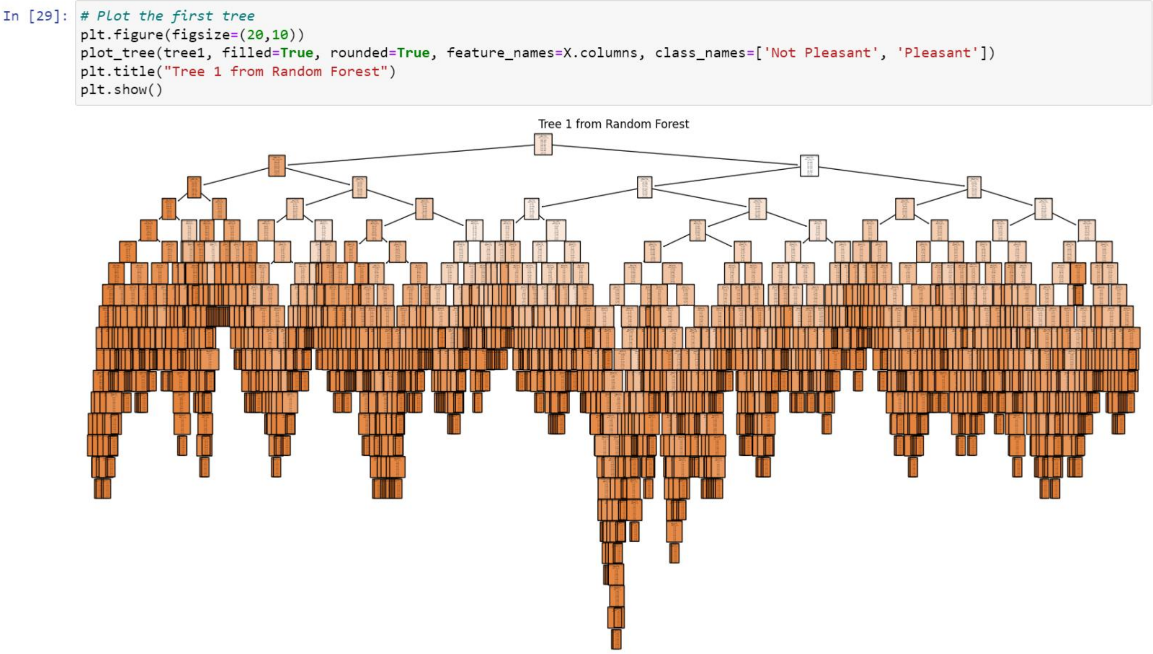
- **Accuracy**: The model achieves an accuracy score of **0.5759**, which suggests that while the model is somewhat successful at predicting "pleasant weather," there is room for improvement.
- **Classification Report**: Precision, recall, and F1-scores are provided for each weather station. The model performs well for some stations, like **BASEL** and **BUDAPEST**, but struggles with underrepresented stations such as **SONNBLICK** and **VALENTIA**.
    - **SONNBLICK**: No positive classifications were observed, which indicates that this station might not have enough data to make reliable predictions.

- **VALENTIA**: Low F1-scores are reported, likely due to insufficient representation in the dataset.

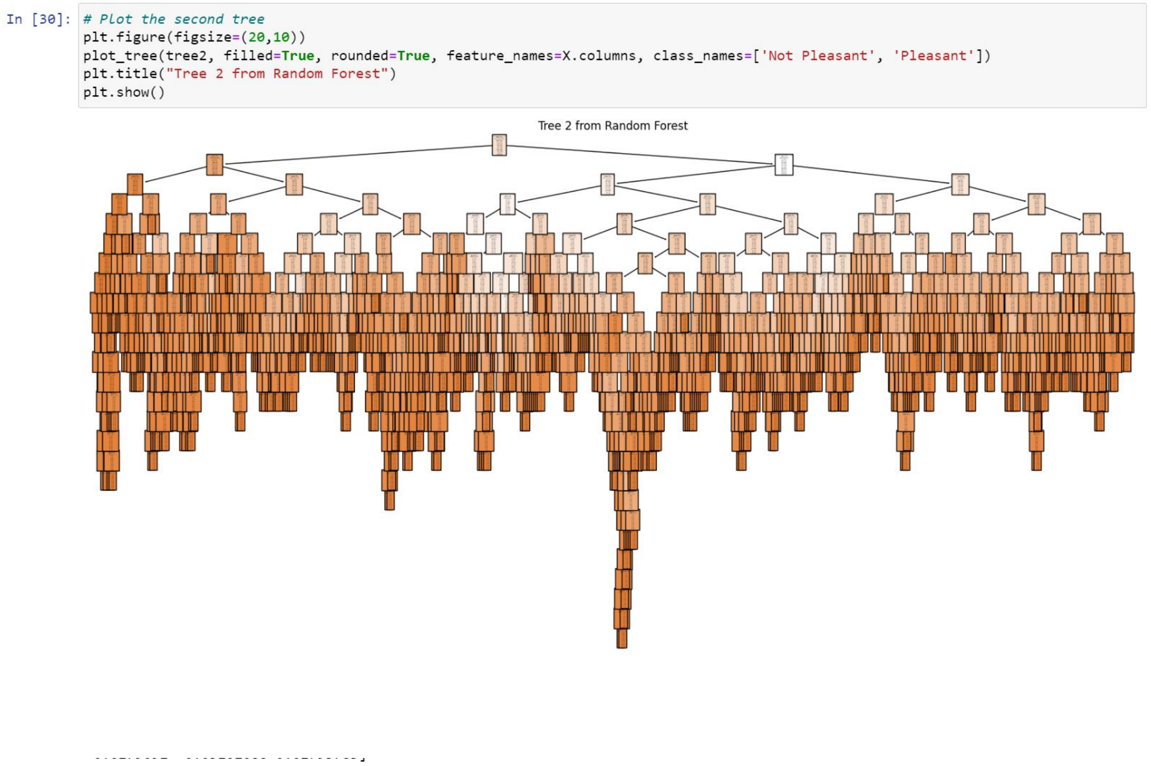


```
In [25]: # Train the model
         rf_model.fit(X_train, y_train)

         # Predict on the test set
         y_pred = rf_model.predict(X_test)

         # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print(f'Accuracy: {accuracy:.4f}')
         print(classification_report(y_test, y_pred))
```

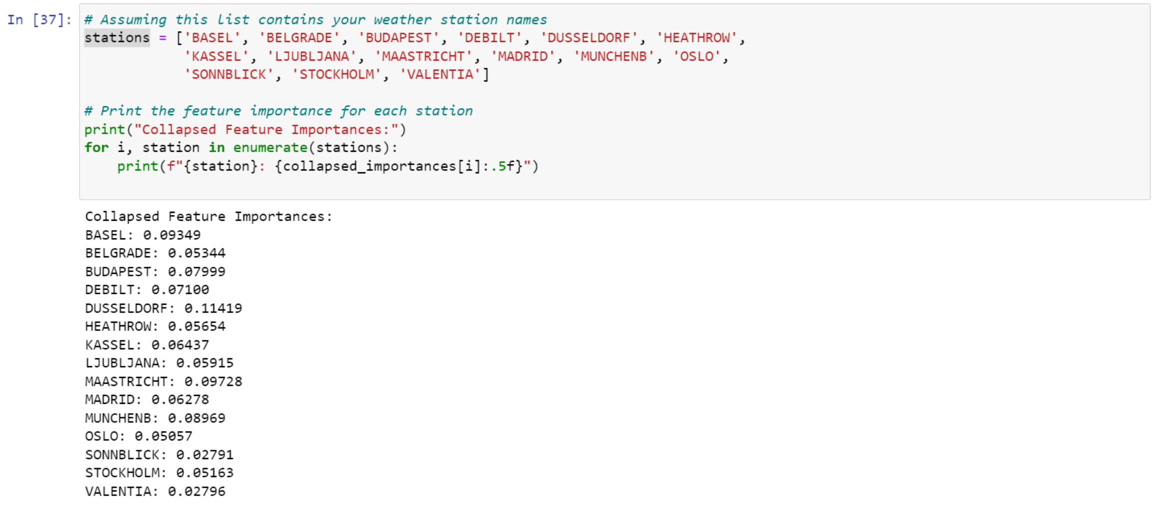
```
Accuracy: 0.5759
              precision    recall  f1-score   support

           0       0.96      0.91      0.93       190
           1       0.89      0.93      0.91       270
           2       0.87      0.98      0.92       255
           3       0.92      0.90      0.91       155
           4       0.93      0.87      0.90       158
           5       0.88      0.73      0.80       160
           6       0.93      0.78      0.85       147
           7       0.87      0.90      0.88       209
           8       0.97      0.91      0.94       160
           9       0.91      0.99      0.94       348
          10       0.96      0.87      0.91       169
          11       0.93      0.70      0.80       120
          12       0.00      0.00      0.00         0
          13       0.90      0.75      0.82       127
          14       1.00      0.05      0.09        44

   micro avg       0.91      0.87      0.89      2512
   macro avg       0.86      0.75      0.77      2512
weighted avg       0.91      0.87      0.88      2512
 samples avg       0.55      0.51      0.52      2512
```



```
In [29]: # Plot the first tree
         plt.figure(figsize=(20,10))
         plot_tree(tree1, filled=True, rounded=True, feature_names=X.columns, class_names=['Not Pleasant', 'Pleasant'])
         plt.title("Tree 1 from Random Forest")
         plt.show()
```



```
In [30]: # Plot the second tree
         plt.figure(figsize=(20,10))
         plot_tree(tree2, filled=True, rounded=True, feature_names=X.columns, class_names=['Not Pleasant', 'Pleasant'])
         plt.title("Tree 2 from Random Forest")
         plt.show()
```

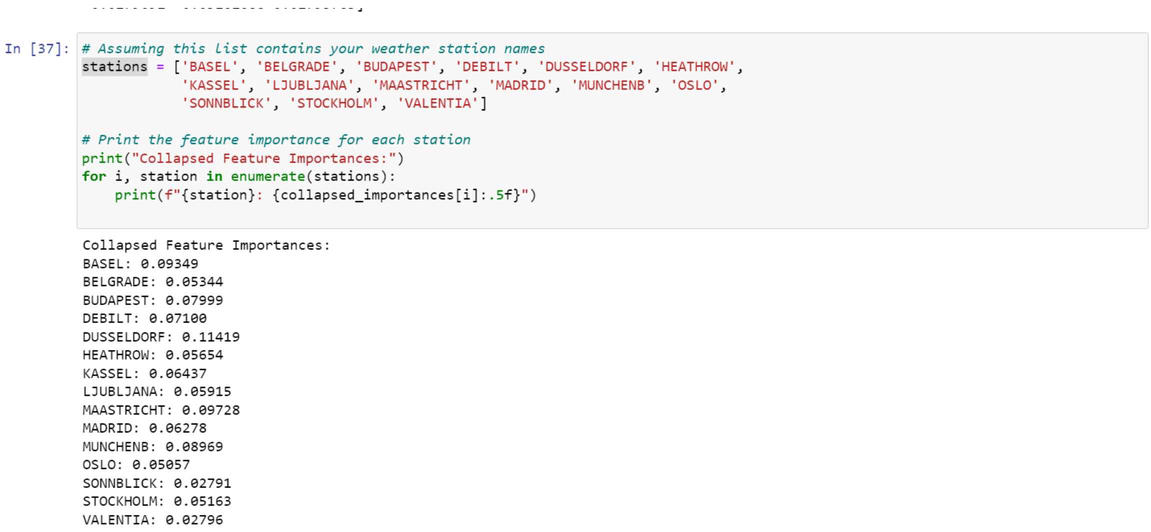


```
In [37]: # Assuming this list contains your weather station names
         stations = ['BASEL', 'BELGRADE', 'BUDAPEST', 'DEBILT', 'DUSSELDORF', 'HEATHROW',
                     'KASSEL', 'LJUBLJANA', 'MAASTRICHT', 'MADRID', 'MUNCHENB', 'OSLO',
                     'SONNBLICK', 'STOCKHOLM', 'VALENTIA']

         # Print the feature importance for each station
         print("Collapsed Feature Importances:")
         for i, station in enumerate(stations):
             print(f"{station}: {collapsed_importances[i]:.5f}")
```

```
Collapsed Feature Importances:
BASEL: 0.09349
BELGRADE: 0.05344
BUDAPEST: 0.07999
DEBILT: 0.07100
DUSSELDORF: 0.11419
HEATHROW: 0.05654
KASSEL: 0.06437
LJUBLJANA: 0.05915
MAASTRICHT: 0.09728
MADRID: 0.06278
MUNCHENB: 0.08969
OSLO: 0.05057
SONNBLICK: 0.02791
STOCKHOLM: 0.05163
VALENTIA: 0.02796
```

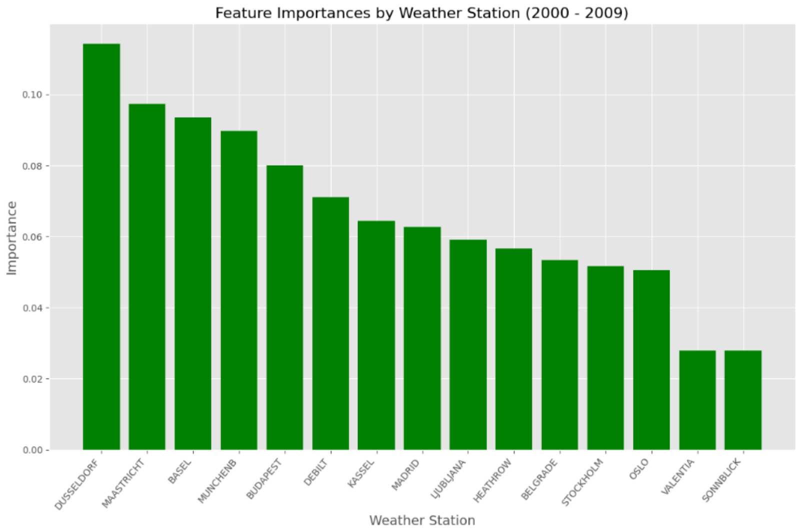
5. Feature Importance:

- A feature importance plot highlights which variables are most influential in predicting "pleasant weather."
- **Temperature-related features** emerge as the most significant predictors, which aligns with the intuitive understanding that temperature is a key factor in determining pleasant weather conditions.



```
In [37]: # Assuming this list contains your weather station names
         stations = ['BASEL', 'BELGRADE', 'BUDAPEST', 'DEBILT', 'DUSSELDORF', 'HEATHROW',
                     'KASSEL', 'LJUBLJANA', 'MAASTRICHT', 'MADRID', 'MUNCHENB', 'OSLO',
                     'SONNBLICK', 'STOCKHOLM', 'VALENTIA']

         # Print the feature importance for each station
         print("Collapsed Feature Importances:")
         for i, station in enumerate(stations):
             print(f"{station}: {collapsed_importances[i]:.5f}")
```

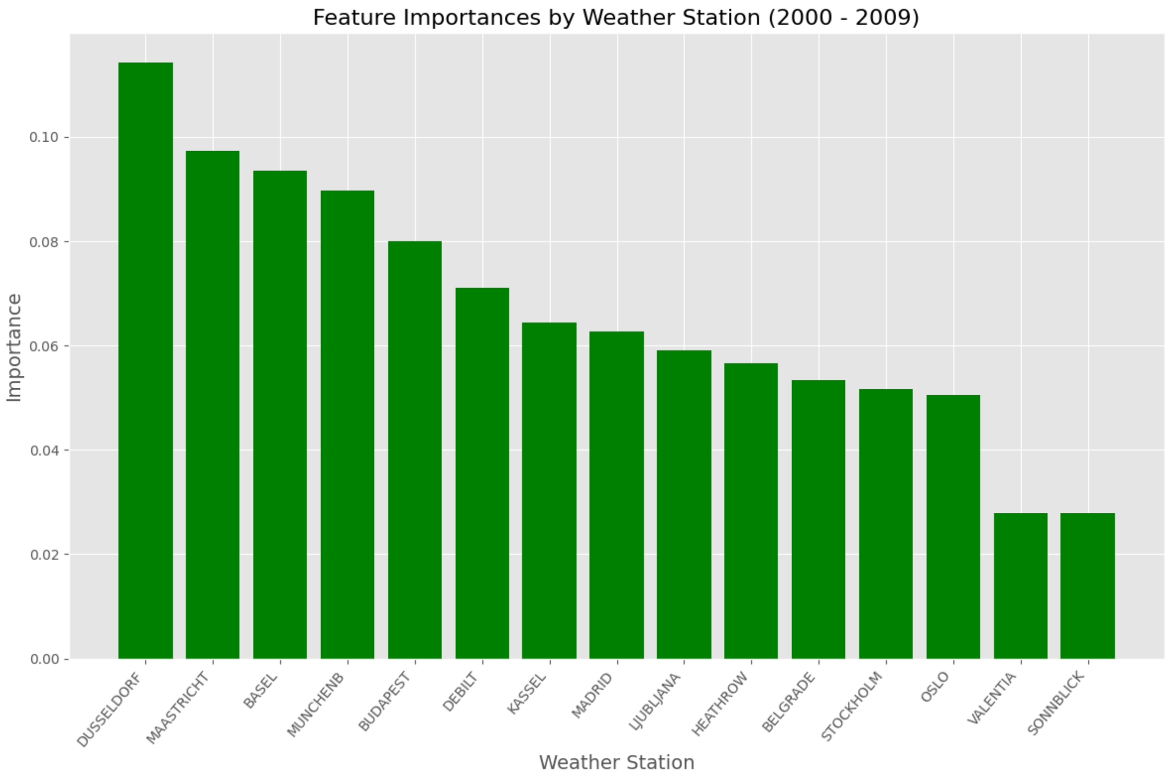
```
Collapsed Feature Importances:
BASEL: 0.09349
BELGRADE: 0.05344
BUDAPEST: 0.07999
DEBILT: 0.07100
DUSSELDORF: 0.11419
HEATHROW: 0.05654
KASSEL: 0.06437
LJUBLJANA: 0.05915
MAASTRICHT: 0.09728
MADRID: 0.06278
MUNCHENB: 0.08969
OSLO: 0.05057
SONNBLICK: 0.02791
STOCKHOLM: 0.05163
VALENTIA: 0.02796
```



6. Challenges & Issues:

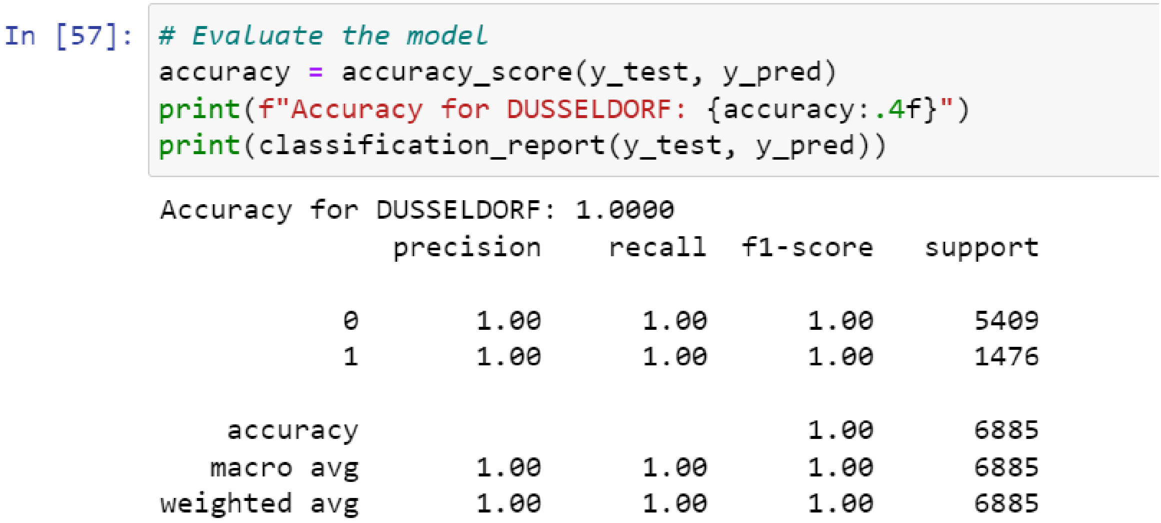
- **Class Imbalance**: Some stations, such as **SONNBLICK** and **VALENTIA**, have insufficient data to build reliable predictions. The class imbalance leads to poor performance in these areas.
- **Model Performance**: While the overall accuracy is moderate, the model could benefit from addressing class imbalance issues and tuning hyperparameters for improved performance across all stations.

7. Top 3 stations



DUSSELDORF

- **Accuracy**: 1.0000
- **Classification Report**:
  - Precision, Recall, F1-score are all **1.00** across both classes (pleasant and non-pleasant weather).
  - **Support**: 5,409 instances of non-pleasant weather and 1,476 instances of pleasant weather in the test set.
- **Key Features**:
  - **Precipitation** is the most significant factor, followed by **maximum temperature** and **mean temperature**.
  - Other important features include **global radiation**, **sunshine**, **cloud cover**, **humidity**, and **pressure**.



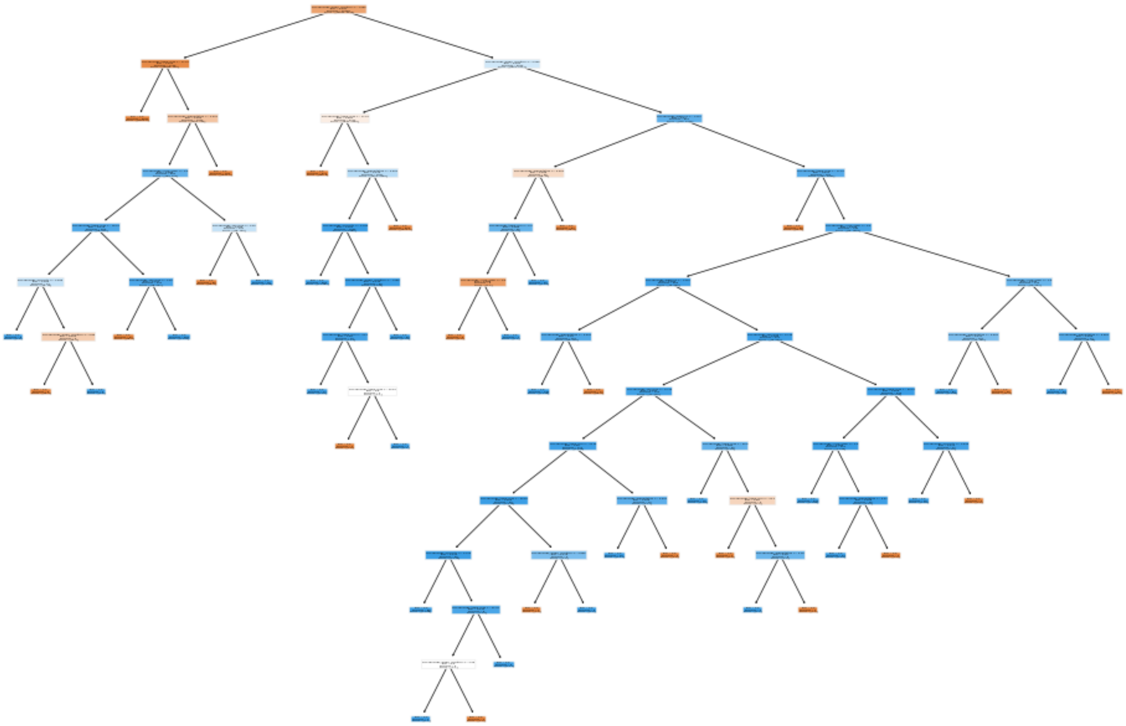
```
In [57]: # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy for DUSSELDORF: {accuracy:.4f}")
         print(classification_report(y_test, y_pred))
```

```
Accuracy for DUSSELDORF: 1.0000
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5409
           1       1.00      1.00      1.00      1476

    accuracy                           1.00      6885
   macro avg       1.00      1.00      1.00      6885
weighted avg       1.00      1.00      1.00      6885
```

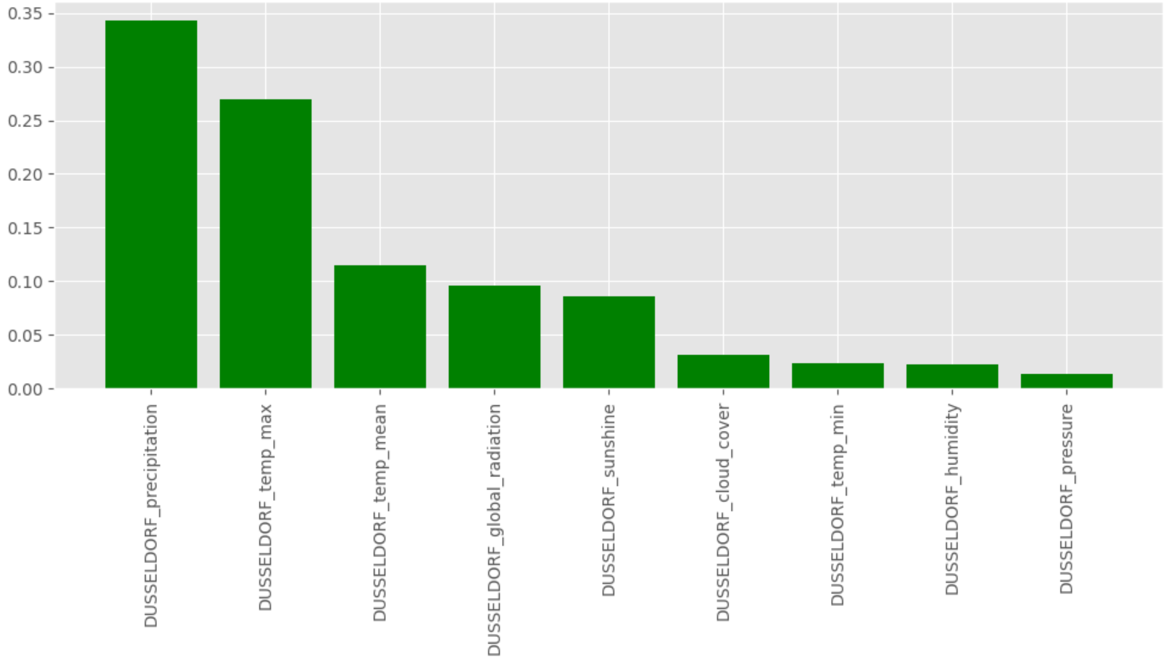
Career Foundry Data Analytics Program

Machine Learning Specialization

Luis Gil



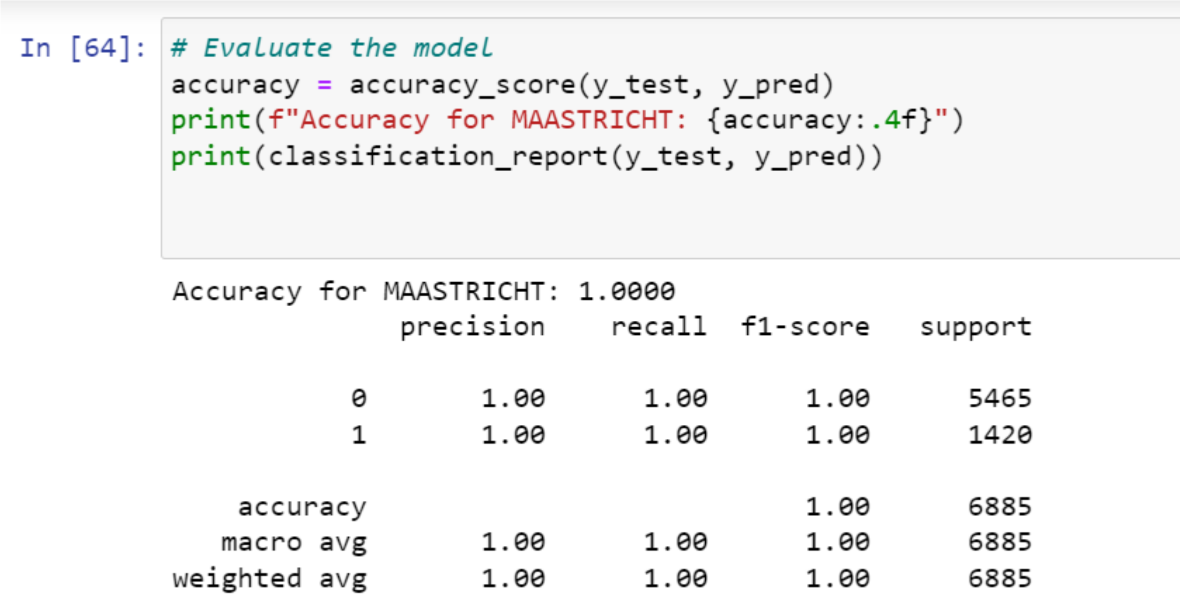
Decision Tree from Random Forest - DUSSELDORF



Feature Importances - DUSSELDORF

MAASTRICHT

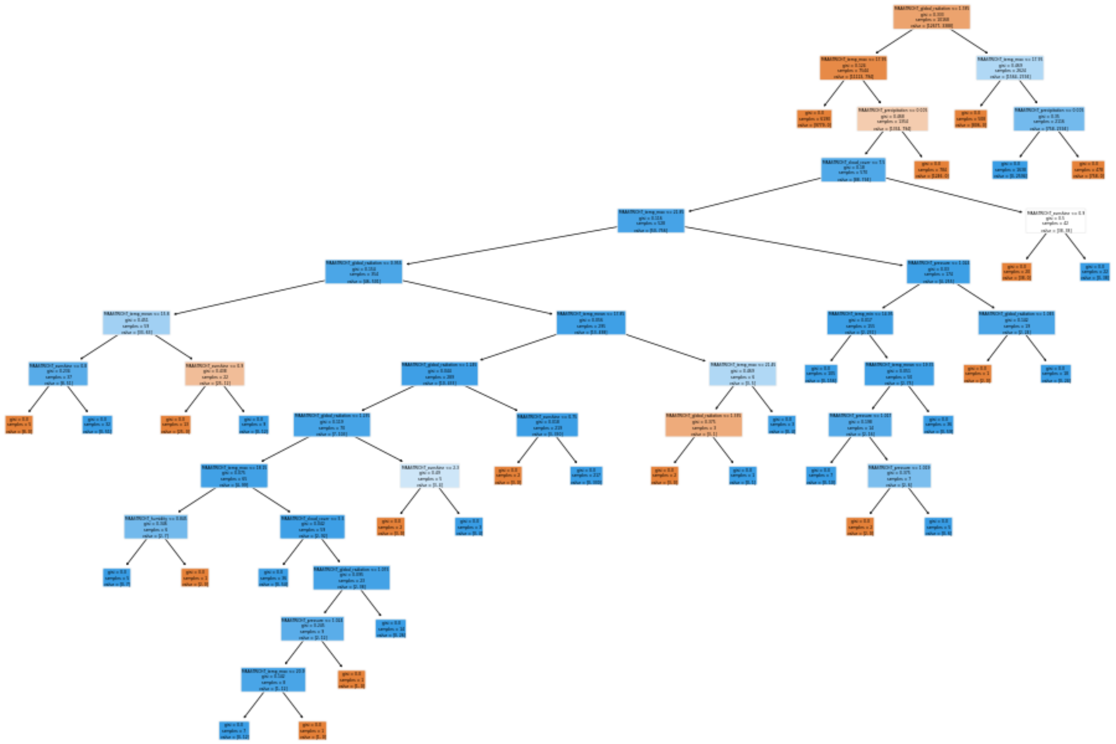
- **Accuracy**: 1.0000
- **Classification Report**:
  - Like DUSSELDORF, MAASTRICHT has perfect precision, recall, and F1-scores.
  - **Support**: 5,465 instances of non-pleasant weather and 1,420 instances of pleasant weather in the test set.
- **Key Features**:
  - The most important features include **precipitation**, **maximum temperature**, **mean temperature**, and **sunshine**.
  - Other features include **humidity**, **cloud cover**, and **pressure**.



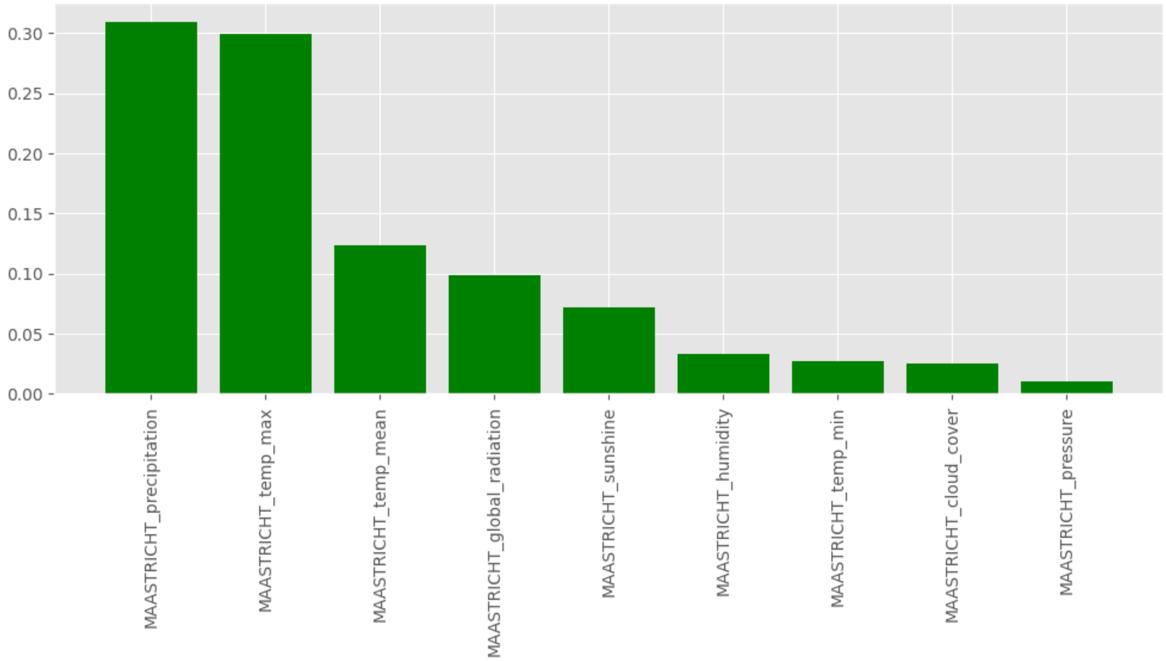
```
In [64]: # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy for MAASTRICHT: {accuracy:.4f}")
         print(classification_report(y_test, y_pred))
```

```
Accuracy for MAASTRICHT: 1.0000
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5465
           1       1.00      1.00      1.00      1420

    accuracy                           1.00      6885
   macro avg       1.00      1.00      1.00      6885
weighted avg       1.00      1.00      1.00      6885
```

Decision Tree from Random Forest - MAASTRICHT



Feature Importances - MAASTRICHT

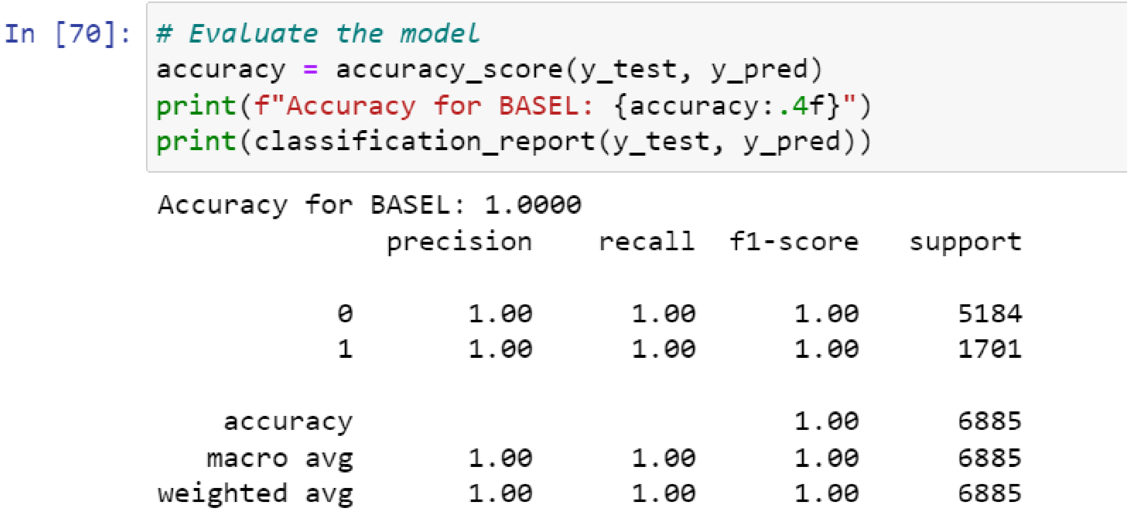


BASEL

- **Accuracy**: 1.0000
- **Classification Report**:
  - BASEL also has perfect classification metrics across all categories.
  - **Support**: 5,184 instances of non-pleasant weather and 1,701 instances of pleasant weather in the test set.
- **Key Features**:
  - Similar to the other stations, **precipitation** and **maximum temperature** are the most critical features.
  - Other important variables include **sunshine**, **global radiation**, **humidity**, **cloud cover**, and **pressure**.

Key Observations:

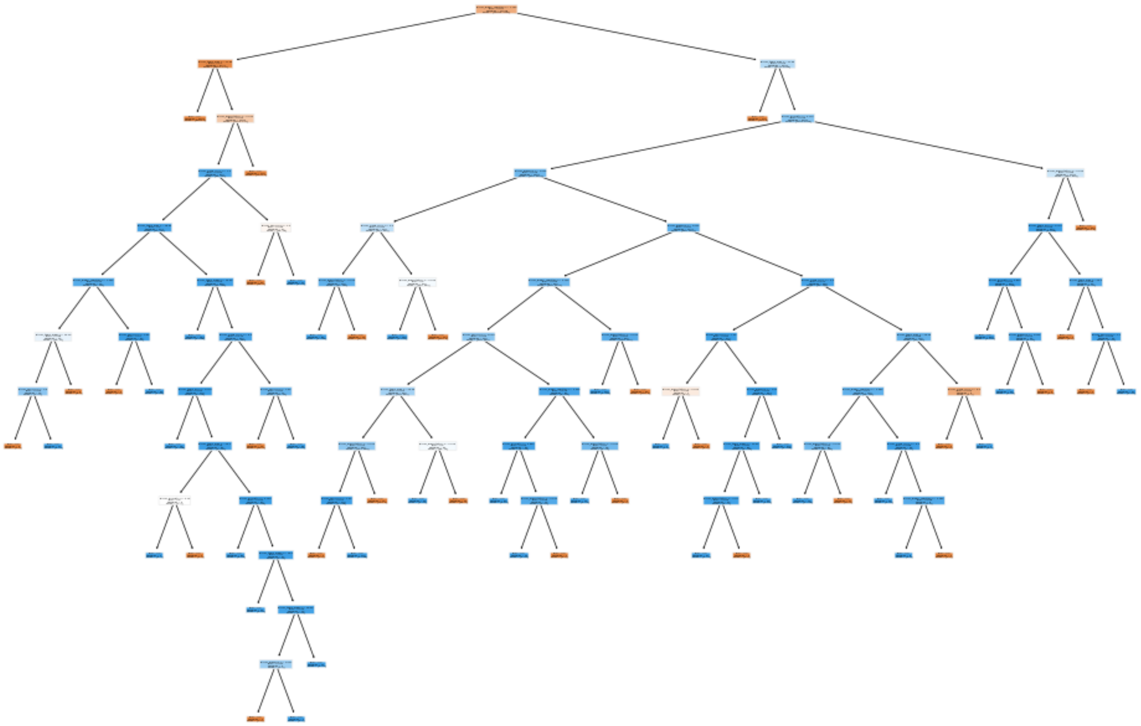
- For all three stations, **precipitation** and **temperature metrics (max and mean)** dominate the feature importance.
- The model has very high accuracy, suggesting it performs extremely well in predicting pleasant vs. non-pleasant weather for these stations.



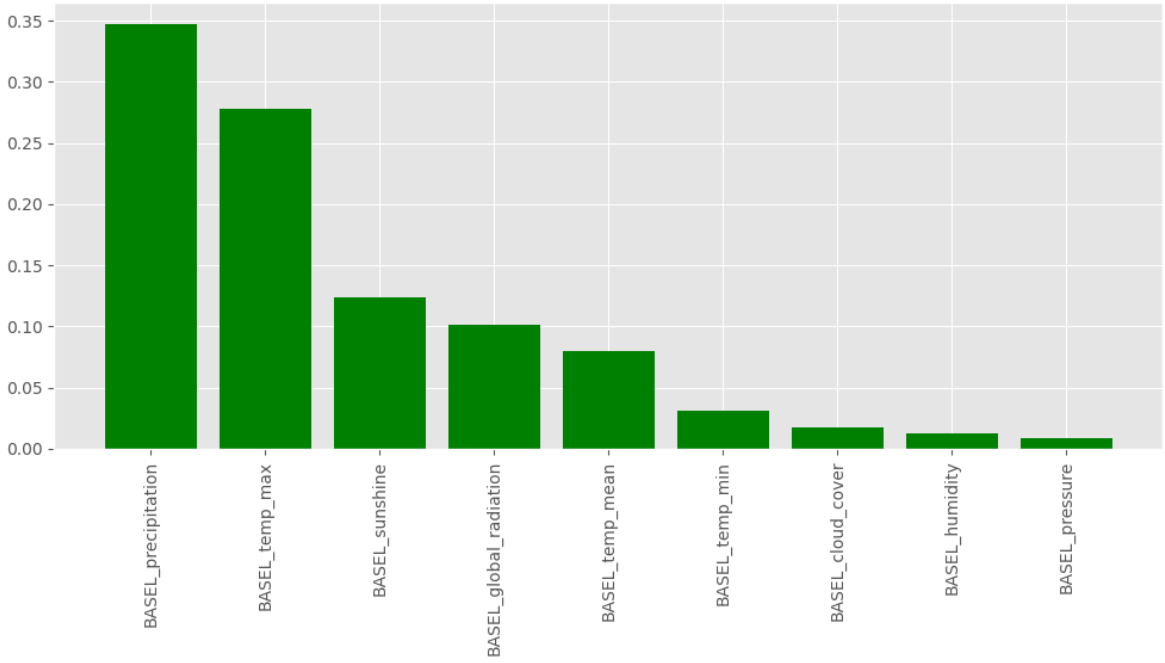
```
In [70]: # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy for BASEL: {accuracy:.4f}")
         print(classification_report(y_test, y_pred))
```

```
Accuracy for BASEL: 1.0000
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5184
           1       1.00      1.00      1.00      1701

    accuracy                           1.00      6885
   macro avg       1.00      1.00      1.00      6885
weighted avg       1.00      1.00      1.00      6885
```

Decision Tree from Random Forest - BASEL



Feature Importances - BASEL



8. Conclusion:

- The random forest model provides reasonable predictions for some stations but struggles with underrepresented locations.
- To further improve model accuracy, addressing data imbalance and optimizing hyperparameters are recommended next steps.
- The analysis of feature importance suggests that temperature metrics are crucial in determining pleasant weather, but other variables may also play a role and should be investigated further.