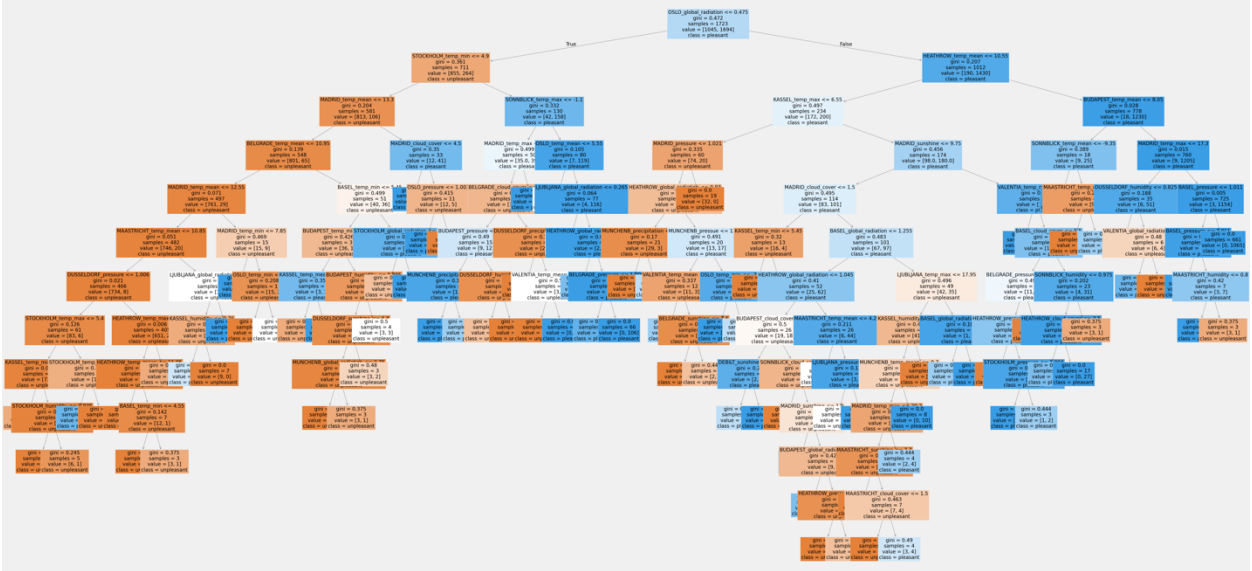
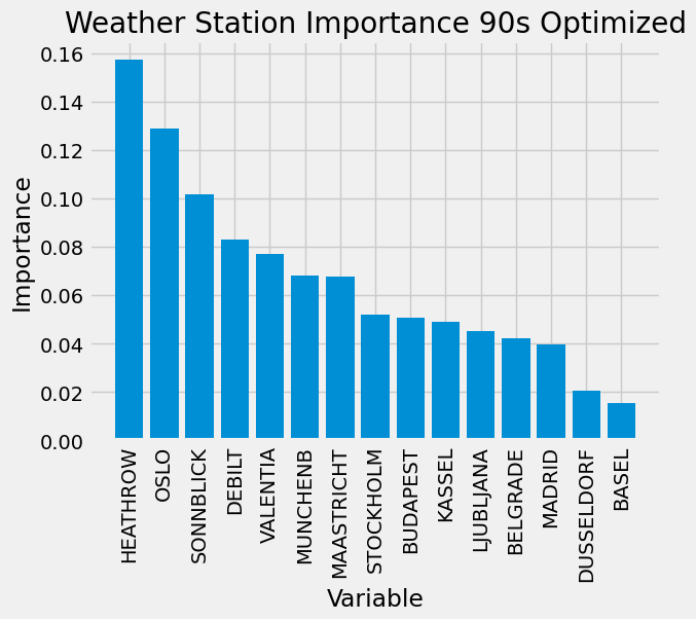
Shaquille Obomeghie
Exercise 2.4

Part 1: Random Forest

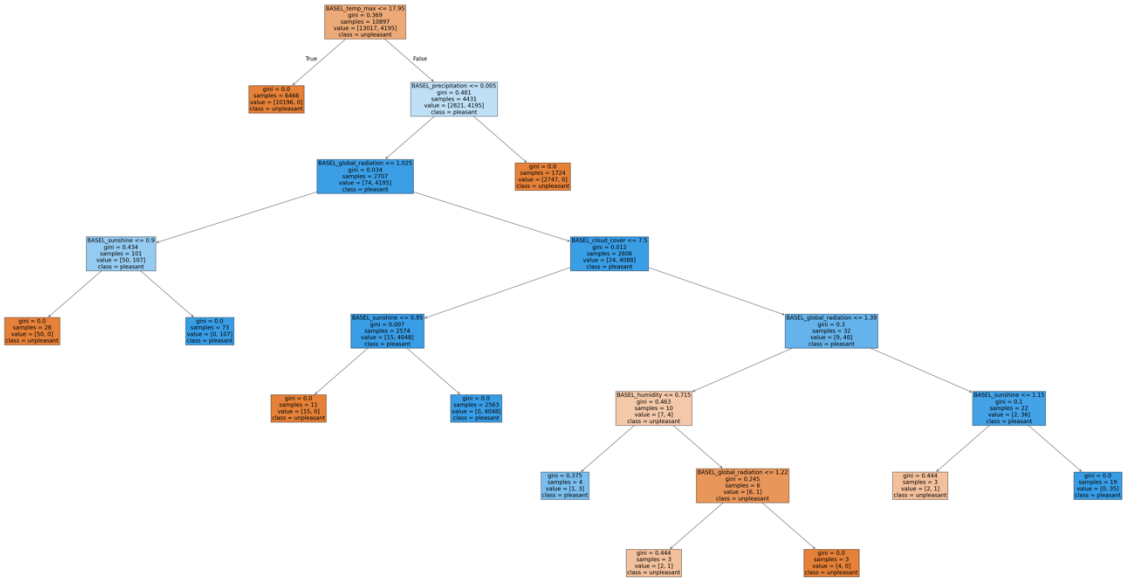
The optimized random forest accuracy is 95% compared to 96% of the previous random forest. The optimized tree has a more direct and understandable tree than that of the unoptimized

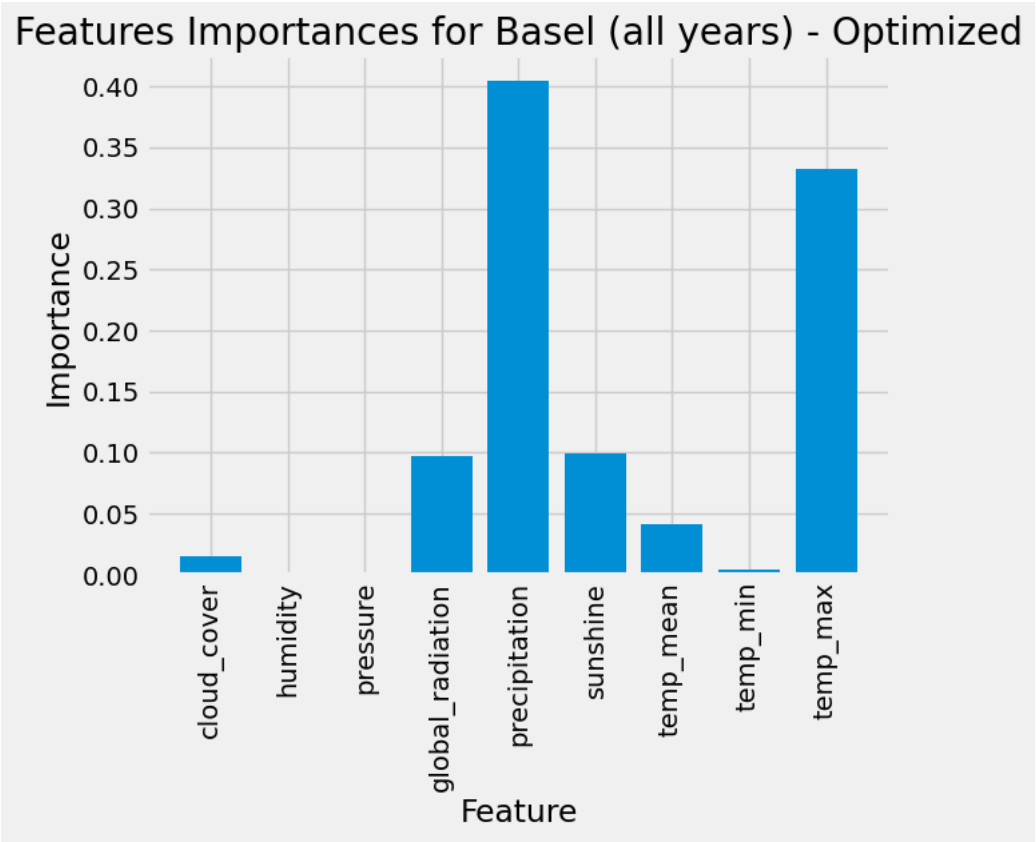




The top 3 importance changed to (Heathrow, Oslo and SOnnblick) in comparison to Madrid, Ljubljana and Budapest

I highlighted Basel and optimized the random tree. **The accuracy optimized to 100% from 99%.** This brings out a straightforward and understandable decision tree.





The feature importance changed to Precipitation, sunshine and temp_max from Precipitation, global_radiation and temp_max. The most notable change was the decrease of the global radiation from the previous unoptimized random tree

Deep Learning: I used the CNN to compared and optimized

```
# Create a Keras leyered model. Use initial hyperparameters: 30, 32, 124, softtmax

epochs = 30
batch_size = 32
n_hidden = 124

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax')) #softmax
```

This is the CNN when it isn't optimized with the accuracy of 13%

```
# Set the model with optimized hyperparameters

epochs = 47
batch_size = 460

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = 15

layers1 = 1
layers2 = 2
activation = 'softsign'
kernel = int(round(1.9444298503238986))  # Rounded kernel size for Conv1D
neurons = 61
normalization = 0.770967179954561
dropout = 0.7296061783380641
dropout_rate = 0.19126724140656393
optimizer = Adadelta(learning_rate=0.7631771981307285)  # Instantiate RMSprop with learning rate

model = Sequential()
model.add(Conv1D(neurons, kernel_size=kernel, activation=activation, input_shape=(timesteps, input_dim)))

if normalization > 0.5:
    model.add(BatchNormalization())

for i in range(layers1):
    model.add(Dense(neurons, activation=activation))

if dropout > 0.5:
    model.add(Dropout(dropout_rate))

for i in range(layers2):
    model.add(Dense(neurons, activation=activation))

model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

This is the CNN when it is optimized and the accuracy goes up to 92%.

Part 3: Iteration

I recommend breaking the data into smaller groups to test and iterate the dataset. This could be done by specific regions, periods such as yearly, and monthly and also by

grouping data based on weather variables such as temperature precipitation. Also, breaking down data into pleasant and unpleasant.

- Which model would you use for each iteration? Expand on your observations from the random forest and deep learning models.
    - I would pick the random forest because it's easier to understand when the tree is refined and when the deeper relationships between variables are emphasized. It also prevents overfitting when hyperparameters are tweaked. This makes the results more reliable. The only problem is it can get somewhat complicated with a larger dataset. Deep learning would be a better choice in that aspect.

- What variables would you recommend that Air Ambulance pay the most attention to while deciding whether it's safe to fly?
    - The air ambulance should prioritize precipitation because a change in precipitation, such as rain and snow, can affect the safety of the flight. Temperature is another important variable to consider. Cloud cover was not given much importance in our analysis but could be a crucial aspect of Air ambulance travel because this could easily affect the visibility of the ambulance.