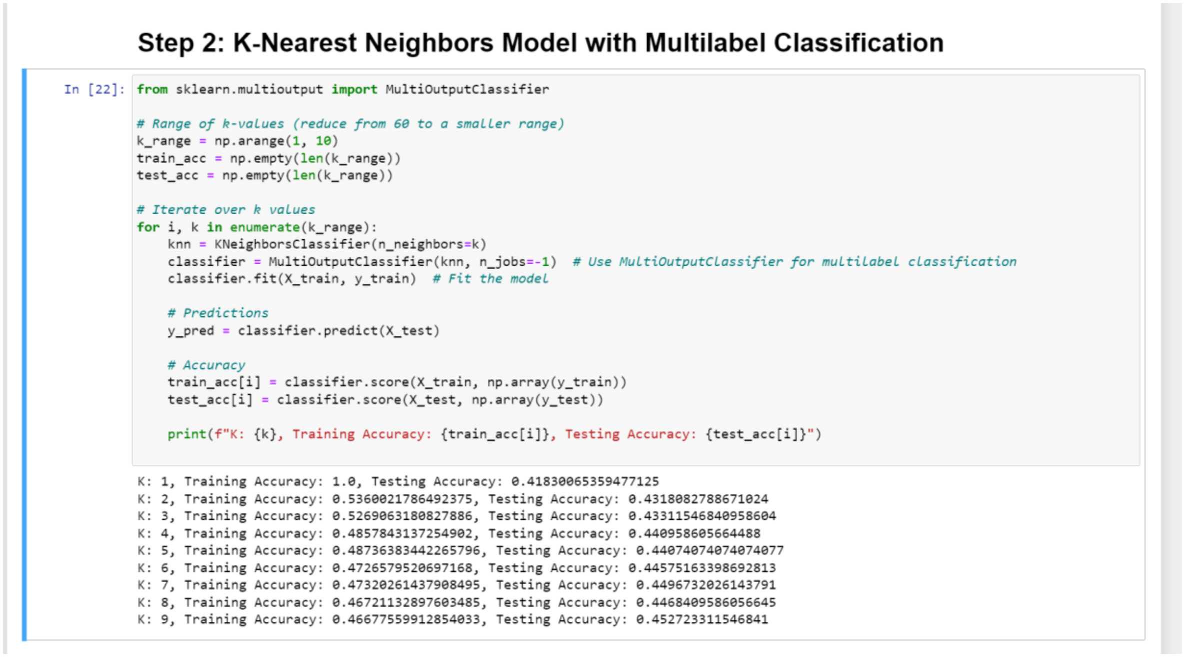
Career Foundry Data Analytics Program

Machine Learning Specialization

Luis Gil

Snippet A & B:



Step 2: K-Nearest Neighbors Model with Multilabel Classification

In [22]:

```python
from sklearn.multioutput import MultiOutputClassifier

# Range of k-values (reduce from 60 to a smaller range)
k_range = np.arange(1, 10)
train_acc = np.empty(len(k_range))
test_acc = np.empty(len(k_range))

# Iterate over k values
for i, k in enumerate(k_range):
    knn = KNeighborsClassifier(n_neighbors=k)
    classifier = MultiOutputClassifier(knn, n_jobs=-1)  # Use MultiOutputClassifier for multilabel classification
    classifier.fit(X_train, y_train)  # Fit the model

    # Predictions
    y_pred = classifier.predict(X_test)

    # Accuracy
    train_acc[i] = classifier.score(X_train, np.array(y_train))
    test_acc[i] = classifier.score(X_test, np.array(y_test))

    print(f"K: {k}, Training Accuracy: {train_acc[i]}, Testing Accuracy: {test_acc[i]}")
```

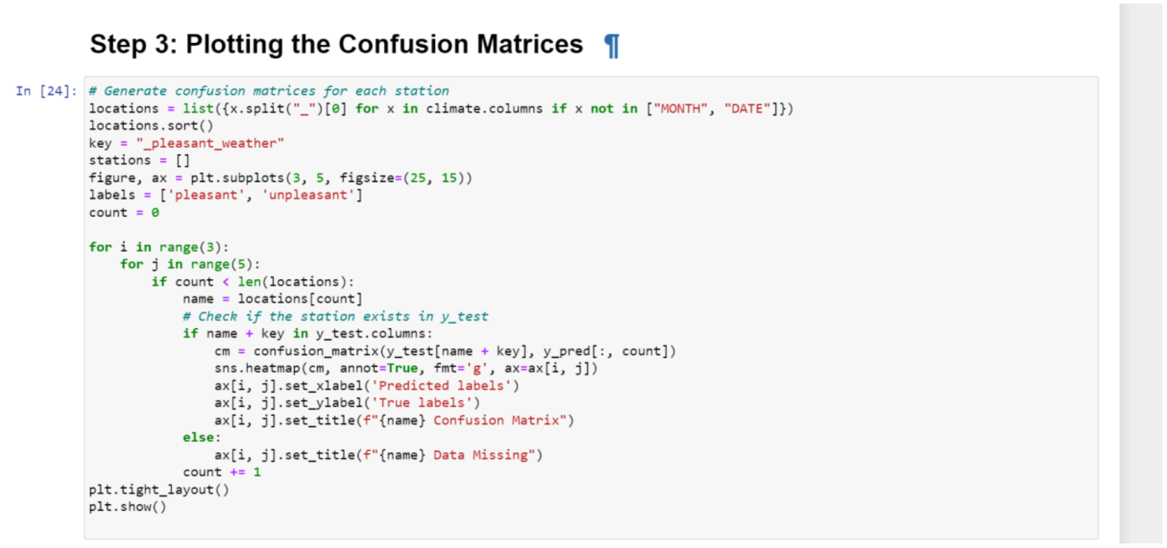
```
K: 1, Training Accuracy: 1.0, Testing Accuracy: 0.41830065359477125
K: 2, Training Accuracy: 0.5360021786492375, Testing Accuracy: 0.4318082788671024
K: 3, Training Accuracy: 0.5269063180827886, Testing Accuracy: 0.43311546840958604
K: 4, Training Accuracy: 0.4857843137254902, Testing Accuracy: 0.440958605664488
K: 5, Training Accuracy: 0.48736383442265796, Testing Accuracy: 0.44074074074074077
K: 6, Training Accuracy: 0.4726579520697168, Testing Accuracy: 0.44575163398692813
K: 7, Training Accuracy: 0.47320261437908495, Testing Accuracy: 0.4496732026143791
K: 8, Training Accuracy: 0.46721132897603485, Testing Accuracy: 0.44684095860566645
K: 9, Training Accuracy: 0.46677559912854033, Testing Accuracy: 0.452723311546841
```

Career Foundry Data Analytics Program

Machine Learning Specialization

Luis Gil

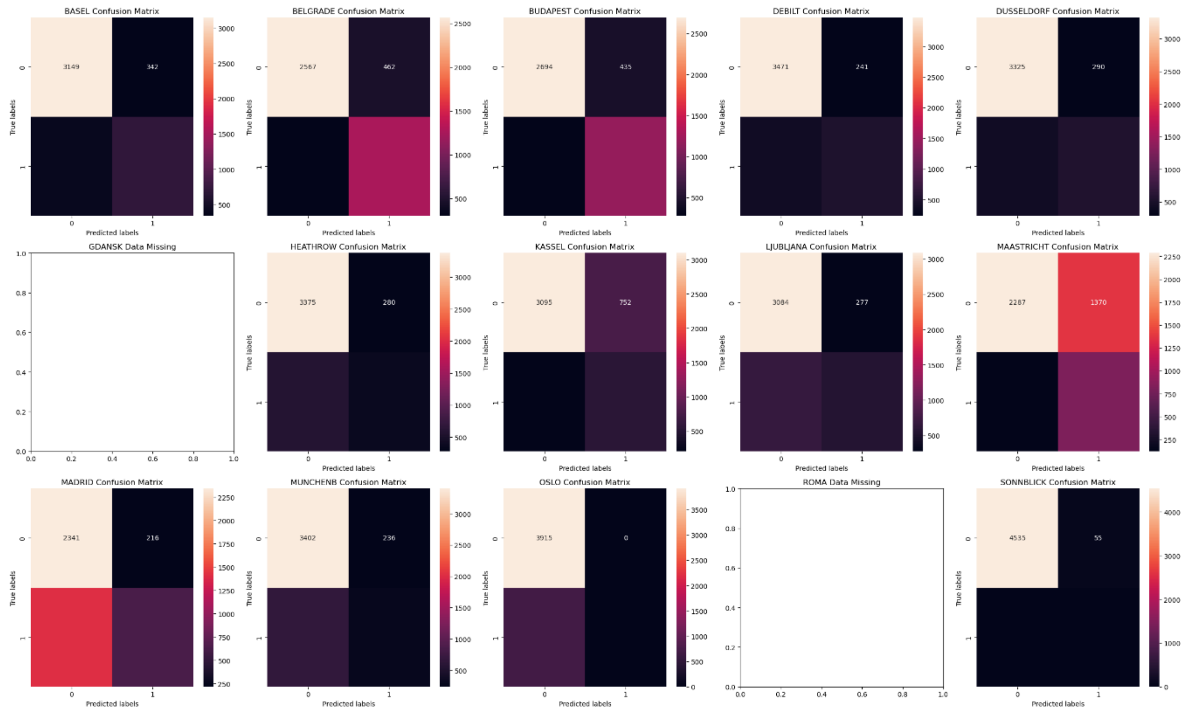
Snippet C



Step 3: Plotting the Confusion Matrices ¶

```
In [24]: # Generate confusion matrices for each station
         locations = list({x.split("_")[0] for x in climate.columns if x not in ["MONTH", "DATE"]})
         locations.sort()
         key = "_pleasant_weather"
         stations = []
         figure, ax = plt.subplots(3, 5, figsize=(25, 15))
         labels = ['pleasant', 'unpleasant']
         count = 0

         for i in range(3):
             for j in range(5):
                 if count < len(locations):
                     name = locations[count]
                     # Check if the station exists in y_test
                     if name + key in y_test.columns:
                         cm = confusion_matrix(y_test[name + key], y_pred[:, count])
                         sns.heatmap(cm, annot=True, fmt='g', ax=ax[i, j])
                         ax[i, j].set_xlabel('Predicted labels')
                         ax[i, j].set_ylabel('True labels')
                         ax[i, j].set_title(f"{name} Confusion Matrix")
                     else:
                         ax[i, j].set_title(f"{name} Data Missing")
                     count += 1
         plt.tight_layout()
         plt.show()
```



Luis Gil

Observation:

The K-Nearest Neighbors (KNN) algorithm shows mixed results when predicting the current weather data. While the training accuracy starts at 1.0 for K=1, indicating overfitting, the testing accuracy remains relatively low, peaking at around 0.4527 for K=9. This suggests that while the model performs well on the training data, it struggles to generalize effectively to unseen data. Some weather stations, such as BASEL, DEBILT, and OSLO, demonstrate relatively higher accuracy with minimal misclassifications, particularly OSLO, which shows no misclassification for unpleasant weather. However, stations like MAASTRICHT exhibit significant misclassification, indicating that the model has difficulty predicting weather for these regions. The variation in accuracy across stations may be due to differences in climate patterns or data characteristics from each station. Additionally, the absence of sufficient data for stations like GDANSK and ROMA highlights gaps in the dataset, potentially impacting overall model accuracy. While increasing K helps to reduce overfitting, the gains in testing accuracy are modest, suggesting that other model improvements may be necessary for better performance.