Christian Campbell

Predicting Rainfall

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [2]:

```
weather_df = pd.read_csv(r"C:\Users\chris\Documents\Bellevue University\10 - Applied Data Science\Project
```

In [3]:

```
weather_df.head()
```

Out[3]:

	DATE	MONTH	BASEL_cloud_cover	BASEL_humidity	BASEL_pressure	BASEL_global_radiation	BASEL_precipitation	BASI
0	20000101	1	8	0.89	1.0286	0.20	0.03	
1	20000102	1	8	0.87	1.0318	0.25	0.00	
2	20000103	1	5	0.81	1.0314	0.50	0.00	
3	20000104	1	7	0.79	1.0262	0.63	0.35	
4	20000105	1	5	0.90	1.0246	0.51	0.07	

5 rows × 165 columns

In [4]:

```
# List of columns to keep
columns_to_keep = [
    'MONTH', 'HEATHROW_cloud_cover', 'HEATHROW_humidity', 'HEATHROW_pressure', 'HEATHROW_global_radiatior
    'HEATHROW_precipitation', 'HEATHROW_sunshine', 'HEATHROW_temp_mean', 'HEATHROW_temp_min', 'HEATHROW_t

# Keep only the columns specified in columns_to_keep
weather_df1 = weather_df[columns_to_keep]

weather_df1.head()
```

Out[4]:

	MONTH	HEATHROW_cloud_cover	HEATHROW_humidity	HEATHROW_pressure	HEATHROW_global_radiation	HEATHROW_preci
0	1	7	0.94	1.0245	0.18	
1	1	7	0.89	1.0253	0.20	
2	1	8	0.91	1.0186	0.13	
3	1	5	0.89	1.0148	0.34	
4	1	5	0.85	1.0142	0.25	

In [5]:

```python
# Applies condition
weather_df1['HEATHROW_rain'] = weather_df1['HEATHROW_precipitation'].apply(lambda x: 1 if x > 0 else 0)

# Creates the new dataframe weather_df2
weather_df2 = weather_df1.copy()

weather_df2.head(10)
```

```
C:\Users\chris\AppData\Local\Temp\ipykernel_24100\1930984631.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexi
ng.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/inde
xing.html#returning-a-view-versus-a-copy)
  weather_df1['HEATHROW_rain'] = weather_df1['HEATHROW_precipitation'].apply(lambda x: 1 if x > 0 el
se 0)
```

Out[5]:

	MONTH	HEATHROW_cloud_cover	HEATHROW_humidity	HEATHROW_pressure	HEATHROW_global_radiation	HEATHROW_p
0	1	7	0.94	1.0245	0.18	
1	1	7	0.89	1.0253	0.20	
2	1	8	0.91	1.0186	0.13	
3	1	5	0.89	1.0148	0.34	

In [6]:

```
# Removes the precipitation column
weather_df3 = weather_df2.drop(columns=['HEATHROW_precipitation'])

# Displays the new dataframe to verify the result
weather_df3.head(10)
```

Out[6]:

	MONTH	HEATHROW_cloud_cover	HEATHROW_humidity	HEATHROW_pressure	HEATHROW_global_radiation	HEATHROW_sunsl
0	1	7	0.94	1.0245	0.18	
1	1	7	0.89	1.0253	0.20	
2	1	8	0.91	1.0186	0.13	
3	1	5	0.89	1.0148	0.34	
4	1	5	0.85	1.0142	0.25	
5	1	6	0.84	1.0127	0.20	
6	1	6	0.82	1.0172	0.31	
7	1	4	0.81	1.0165	0.52	
8	1	0	0.84	1.0276	0.55	
9	1	5	0.86	1.0347	0.40	

Split data

In [7]:

```
# Splits the data into features (X) and target (y)
X = weather_df3.drop(columns=["HEATHROW_rain"])
y = weather_df3["HEATHROW_rain"]

# Splits the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training, testing and evaluating the logistic regression model

In [8]:

```
# Initializes the logistic regression model
log_reg = LogisticRegression()

# Train the model on the training set
log_reg.fit(X_train, y_train)
```

```
C:\Users\chris\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: l
bfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/module
s/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-lear
n.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[8]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [9]:

```
# Predicts on the test set
y_pred = log_reg.predict(X_test)
```

In [10]:

```
# Evaluates the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [11]:

```
# Displays the evaluation metrics
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.7250341997264022
Confusion Matrix:
[[242 124]
 [ 77 288]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.66      0.71       366
           1       0.70      0.79      0.74       365

    accuracy                           0.73       731
   macro avg       0.73      0.73      0.72       731
weighted avg       0.73      0.73      0.72       731
```

Training, testing and evaluating the random forest model

In [12]:

```
# Initializes the Random Forest classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Trains the model on the training set
rf_classifier.fit(X_train, y_train)
```

Out[12]:

```
RandomForestClassifier(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:

```
# Predicts on the test set
y_pred = rf_classifier.predict(X_test)
```

In [14]:

```
# Evaluates the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [15]:

```
# Display the evaluation metrics
print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
```

```
Accuracy: 0.7811217510259918
Confusion Matrix:
[[286  80]
 [ 80 285]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.78      0.78       366
           1       0.78      0.78      0.78       365

    accuracy                           0.78       731
   macro avg       0.78      0.78      0.78       731
weighted avg       0.78      0.78      0.78       731
```

Feature Importance

In [18]:

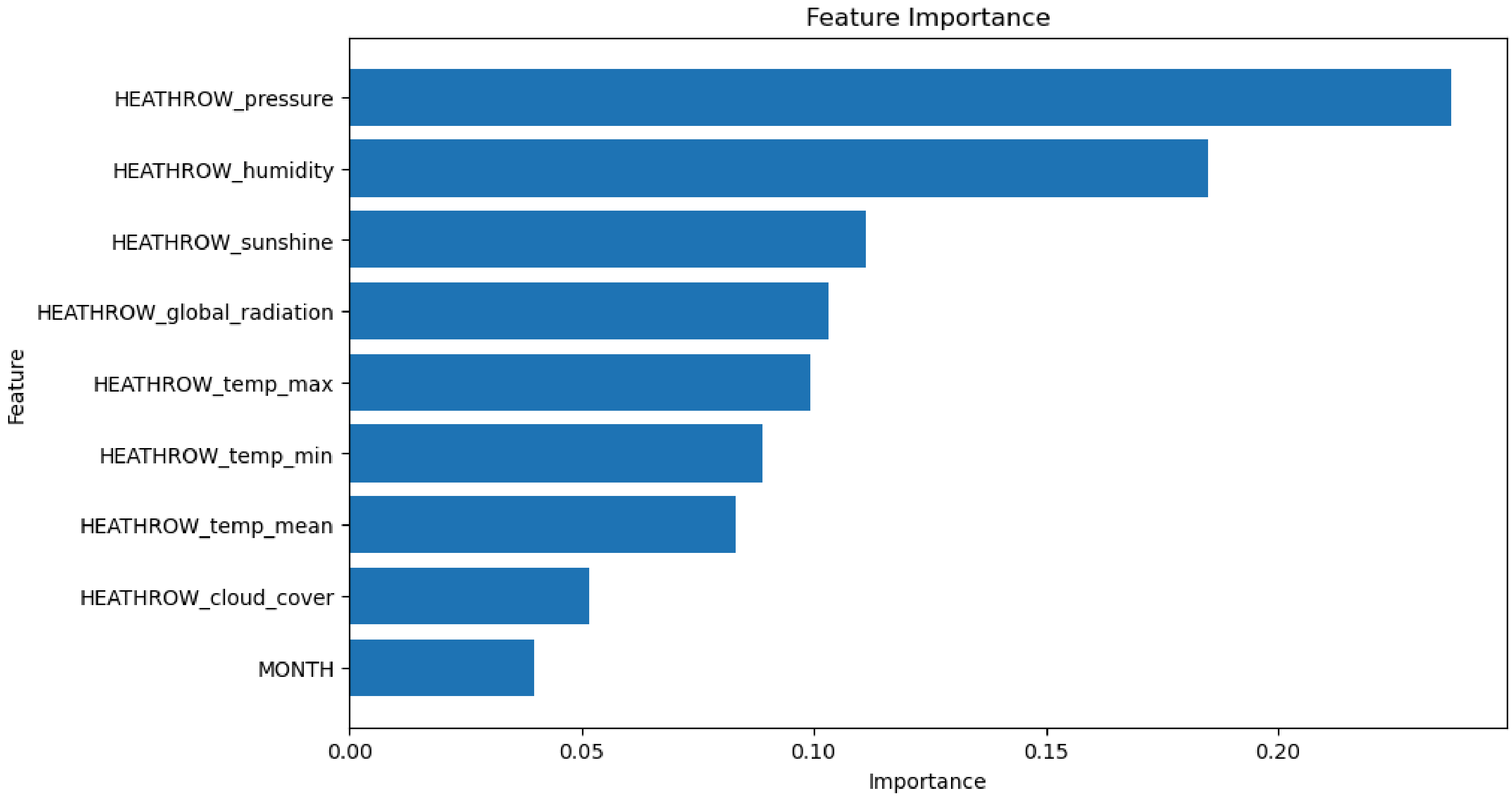
```
# Gets feature importance
feature_importance = rf_classifier.feature_importances_

# Creates a DataFrame for feature importance
feature_importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': feature_importance
}).sort_values(by='Importance', ascending=False)

# Adds a rank column to the DataFrame
feature_importance_df['Rank'] = range(1, len(feature_importance_df) + 1)
```

In [19]:

```
# Plots the feature importance as a bar graph
plt.figure(figsize=(10, 6))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'])
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance')
plt.gca().invert_yaxis()
plt.show()
```



In [20]:

```
# Displays the feature importance table ranked by importance
print("Feature Importance Ranked by Importance:")
print(feature_importance_df[['Rank', 'Feature', 'Importance']].reset_index(drop=True))
```

```
Feature Importance Ranked by Importance:
   Rank                    Feature  Importance
0     1          HEATHROW_pressure    0.237404
1     2          HEATHROW_humidity    0.184932
2     3          HEATHROW_sunshine    0.111176
3     4  HEATHROW_global_radiation    0.103171
4     5          HEATHROW_temp_max    0.099372
5     6          HEATHROW_temp_min    0.089204
6     7         HEATHROW_temp_mean    0.083304
7     8        HEATHROW_cloud_cover    0.051720
8     9                      MONTH    0.039718
```

In [ ]: