```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from tensorflow.keras.layers import Conv1D, Flatten, Reshape
```

```
# Load the datasets
weather_df = pd.read_csv('weather_prediction_dataset.csv')
bbq_labels_df = pd.read_csv('weather_prediction_bbq_labels.csv')
```

```
# Display the first few rows of the weather dataset
print("First few rows of the weather dataset:")
print(weather_df.head(6))
```

```
First few rows of the weather dataset:
       DATE  MONTH  BASEL_cloud_cover  BASEL_humidity  BASEL_pressure  \
0  20000101      1                  8            0.89          1.0286
1  20000102      1                  8            0.87          1.0318
2  20000103      1                  5            0.81          1.0314
3  20000104      1                  7            0.79          1.0262
4  20000105      1                  5            0.90          1.0246
5  20000106      1                  3            0.85          1.0244

   BASEL_global_radiation  BASEL_precipitation  BASEL_sunshine  \
0                    0.20                 0.03             0.0
1                    0.25                 0.00             0.0
2                    0.50                 0.00             3.7
3                    0.63                 0.35             6.9
4                    0.51                 0.07             3.7
5                    0.56                 0.00             5.7

   BASEL_temp_mean  BASEL_temp_min  ...  STOCKHOLM_temp_min  \
0              2.9             1.6  ...                -9.3
1              3.6             2.7  ...                 0.5
2              2.2             0.1  ...                -1.0
3              3.9             0.5  ...                 2.5
4              6.0             3.8  ...                -1.8
5              4.2             1.9  ...                -0.6

   STOCKHOLM_temp_max  TOURS_wind_speed  TOURS_humidity  TOURS_pressure  \
0                 0.7               1.6            0.97          1.0275
1                 2.0               2.0            0.99          1.0293
2                 2.8               3.4            0.91          1.0267
3                 4.6               4.9            0.95          1.0222
4                 2.9               3.6            0.95          1.0209
5                 4.0               3.4            0.92          1.0209

   TOURS_global_radiation  TOURS_precipitation  TOURS_temp_mean  \
0                    0.25                 0.04              8.5
1                    0.17                 0.16              7.9
2                    0.27                 0.00              8.1
3                    0.11                 0.44              8.6
4                    0.39                 0.04              8.0
5                    0.55                 0.20              7.1

   TOURS_temp_min  TOURS_temp_max
0             7.2             9.8
1             6.6             9.2
2             6.6             9.6
3             6.4            10.8
4             6.4             9.5
5             3.5            10.7

[6 rows x 165 columns]
```

```
# Display the first few rows of the BBQ labels dataset
print("\nFirst few rows of the BBQ labels dataset:")
print(bbq_labels_df.head(6))
```

```
First few rows of the BBQ labels dataset:
       DATE  BASEL_BBQ_weather  BUDAPEST_BBQ_weather  DE_BBQ_weather  \
0  20000101              False                 False           False
```

```
1  20000102              False                False           False
2  20000103              False                False           False
3  20000104              False                False           False
4  20000105              False                False           False
5  20000106              False                False           False

   DRESDEN_BBQ_weather  DUSSELDORF_BBQ_weather  HEATHROW_BBQ_weather  \
0                False                   False                 False
1                False                   False                 False
2                False                   False                 False
3                False                   False                 False
4                False                   False                 False
5                False                   False                 False

   KASSEL_BBQ_weather  LJUBLJANA_BBQ_weather  MAASTRICHT_BBQ_weather  \
0               False                  False                   False
1               False                  False                   False
2               False                  False                   False
3               False                  False                   False
4               False                  False                   False
5               False                  False                   False

   MALMO_BBQ_weather  MONTELIMAR_BBQ_weather  MUENCHEN_BBQ_weather  \
0              False                  False                 False
1              False                  False                 False
2              False                  False                 False
3              False                  False                 False
4              False                  False                 False
5              False                  False                 False

   OSLO_BBQ_weather  PERPIGNAN_BBQ_weather  SONNBLICK_BBQ_weather  \
0             False                  False                  False
1             False                  False                  False
2             False                  False                  False
3             False                  False                  False
4             False                  False                  False
5             False                  False                  False

   STOCKHOLM_BBQ_weather  TOURS_BBQ_weather
0                  False              False
1                  False              False
2                  False              False
3                  False              False
4                  False              False
5                  False              False
```

```
# Get the column names and number of columns in the BBQ labels dataset
bbq_columns_len, bbq_columns = len(bbq_labels_df.columns), bbq_labels_df.columns
print("\nNumber of columns in the BBQ labels dataset:", bbq_columns_len)
print("Column names in the BBQ labels dataset:", bbq_columns)
```

```
Number of columns in the BBQ labels dataset: 18
Column names in the BBQ labels dataset: Index(['DATE', 'BASEL_BBQ_weather', 'BUDAPEST_BBQ_weather', 'DE_BBQ_weather',
       'DRESDEN_BBQ_weather', 'DUSSELDORF_BBQ_weather', 'HEATHROW_BBQ_weather',
       'KASSEL_BBQ_weather', 'LJUBLJANA_BBQ_weather', 'MAASTRICHT_BBQ_weather',
       'MALMO_BBQ_weather', 'MONTELIMAR_BBQ_weather', 'MUENCHEN_BBQ_weather',
       'OSLO_BBQ_weather', 'PERPIGNAN_BBQ_weather', 'SONNBLICK_BBQ_weather',
       'STOCKHOLM_BBQ_weather', 'TOURS_BBQ_weather'],
      dtype='object')
```

```
# Get the column names in the weather dataset
weather_columns = weather_df.columns
print("\nColumn names in the weather dataset:", weather_columns)
```

```
Column names in the weather dataset: Index(['DATE', 'MONTH', 'BASEL_cloud_cover', 'BASEL_humidity',
       'BASEL_pressure', 'BASEL_global_radiation', 'BASEL_precipitation',
       'BASEL_sunshine', 'BASEL_temp_mean', 'BASEL_temp_min',
       ...
       'STOCKHOLM_temp_min', 'STOCKHOLM_temp_max', 'TOURS_wind_speed',
       'TOURS_humidity', 'TOURS_pressure', 'TOURS_global_radiation',
       'TOURS_precipitation', 'TOURS_temp_mean', 'TOURS_temp_min',
       'TOURS_temp_max'],
      dtype='object', length=165)
```

```
# Selecting Dresden's weather data for BBQ prediction
dresden_weather_columns = weather_columns[30:41]
dresden_weather_df = weather_df[['DATE'] + list(dresden_weather_columns)]
dresden_weather_df['BBQ'] = bbq_labels_df['DRESDEN_BBQ_weather']
```

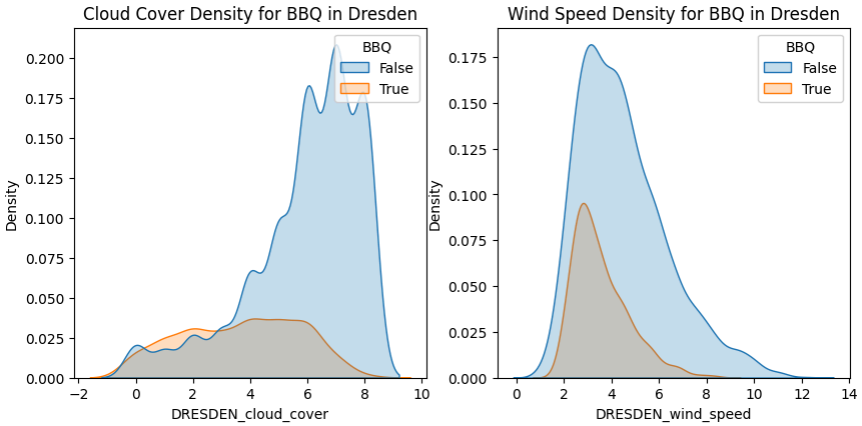
```
<ipython-input-30-bdfde4e4bfa9>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
  dresden_weather_df['BBQ'] = bbq_labels_df['DRESDEN_BBQ_weather']
```

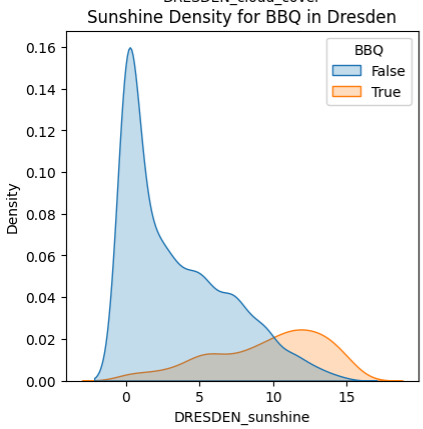
```
# Display the first few rows of Dresden's weather data
print("\nDresden's weather data for BBQ prediction:")
print(dresden_weather_df.head())
```

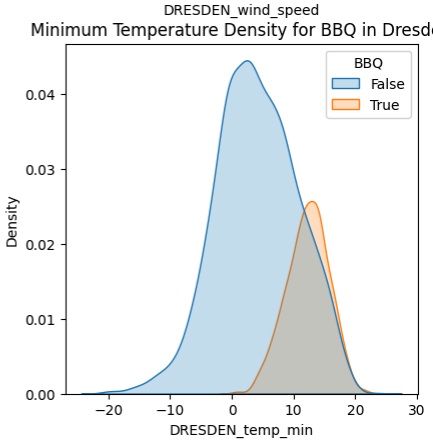
```
Dresden's weather data for BBQ prediction:
       DATE  DRESDEN_cloud_cover  DRESDEN_wind_speed  DRESDEN_wind_gust  \
0  20000101                    8                 3.2                7.2
1  20000102                    7                 4.0                8.8
2  20000103                    7                 5.4               12.1
3  20000104                    8                 6.0               14.4
4  20000105                    2                 5.6               15.8

   DRESDEN_humidity  DRESDEN_global_radiation  DRESDEN_precipitation  \
0              0.89                      0.09                   0.32
1              0.89                      0.23                   0.00
2              0.79                      0.18                   0.00
3              0.88                      0.11                   0.22
4              0.76                      0.49                   0.00

   DRESDEN_sunshine  DRESDEN_temp_mean  DRESDEN_temp_min  DRESDEN_temp_max  \
0               0.0                1.0              -1.8               2.0
1               0.4                2.5               1.4               4.0
2               0.0                4.2               1.3               5.1
3               0.0                4.4               3.4               5.2
4               5.7                1.8              -0.5               6.9

   DUSSELDORF_cloud_cover    BBQ
0                       8  False
1                       6  False
2                       7  False
3                       7  False
4                       4  False
```

Data Visualization

```
# Subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_cloud_cover', hue='BBQ', fill=True, ax=axs[0, 0])
axs[0, 0].set_title('Cloud Cover Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_wind_speed', hue='BBQ', fill=True, ax=axs[0, 1])
axs[0, 1].set_title('Wind Speed Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_sunshine', hue='BBQ', fill=True, ax=axs[1, 0])
axs[1, 0].set_title('Sunshine Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_temp_min', hue='BBQ', fill=True, ax=axs[1, 1])
axs[1, 1].set_title('Minimum Temperature Density for BBQ in Dresden')
plt.show()
```

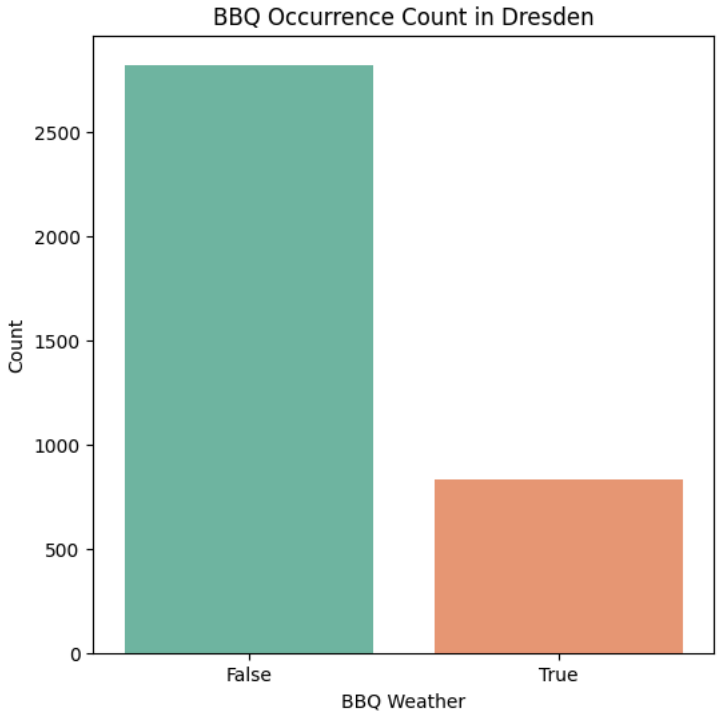




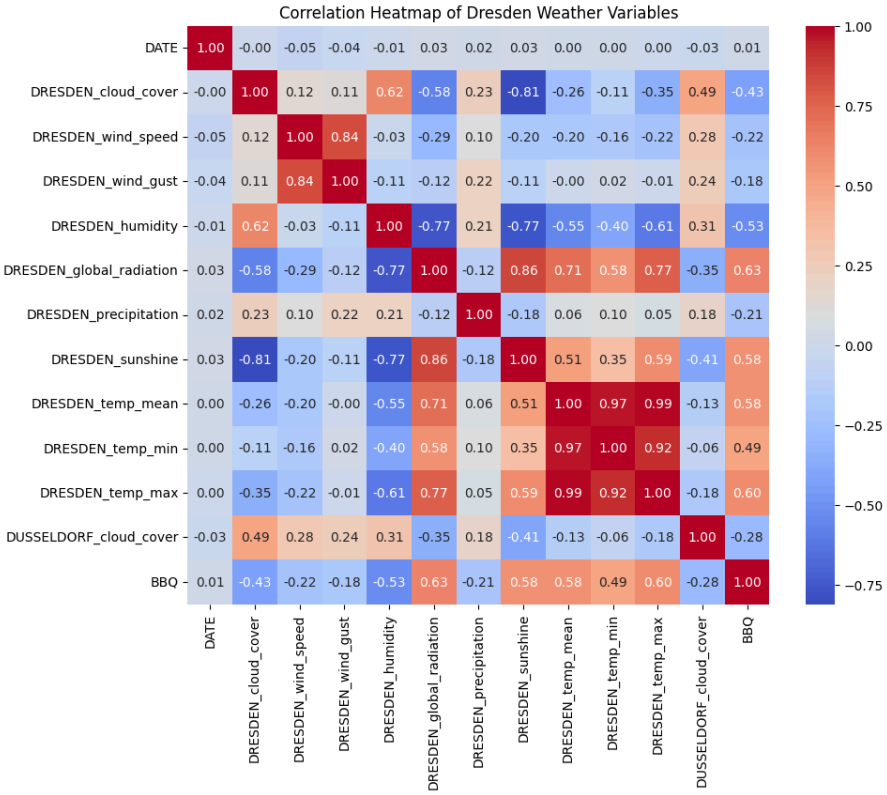



```
# BBQ Occurance Countplot
plt.figure(figsize=(6, 6))
sns.countplot(data=dresden_weather_df, x='BBQ', palette='Set2')
plt.title('BBQ Occurrence Count in Dresden')
plt.xlabel('BBQ Weather')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-33-68a9544d8eed>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(data=dresden_weather_df, x='BBQ', palette='Set2')
```

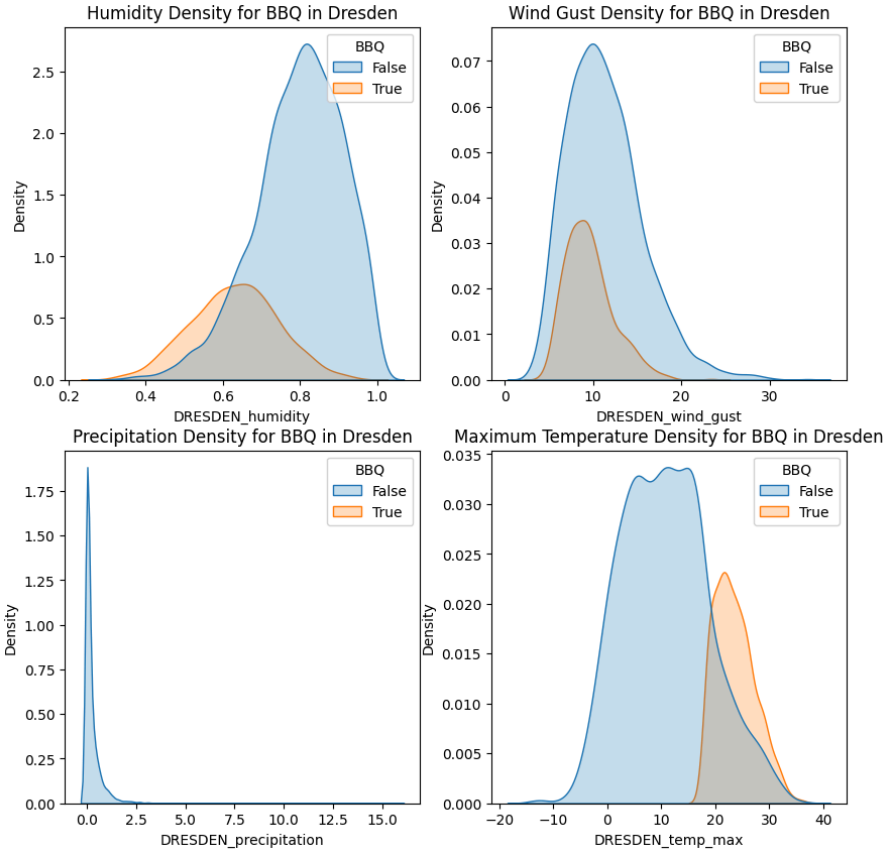



```
# Correlation Heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(dresden_weather_df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap of Dresden Weather Variables')
plt.show()
```

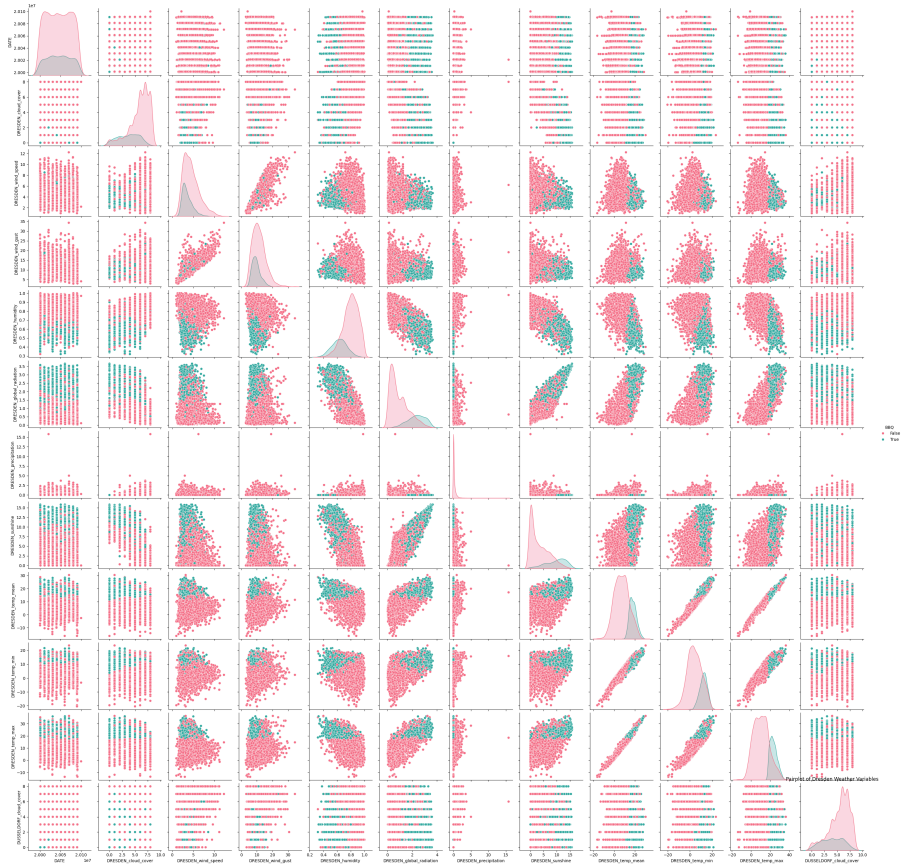



```
# Subplots
fig, axs = plt.subplots(2, 2, figsize=(10, 10))
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_humidity', hue='BBQ', fill=True, ax=axs[0, 0])
axs[0, 0].set_title('Humidity Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_wind_gust', hue='BBQ', fill=True, ax=axs[0, 1])
axs[0, 1].set_title('Wind Gust Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_precipitation', hue='BBQ', fill=True, ax=axs[1, 0])
axs[1, 0].set_title('Precipitation Density for BBQ in Dresden')
sns.kdeplot(data=dresden_weather_df, x='DRESDEN_temp_max', hue='BBQ', fill=True, ax=axs[1, 1])
axs[1, 1].set_title('Maximum Temperature Density for BBQ in Dresden')
plt.show()
```

```
<ipython-input-35-d2f6ac38d4cd>:7: UserWarning: Dataset has 0 variance; skipping density
  sns.kdeplot(data=dresden_weather_df, x='DRESDEN_precipitation', hue='BBQ', fill=True,
```

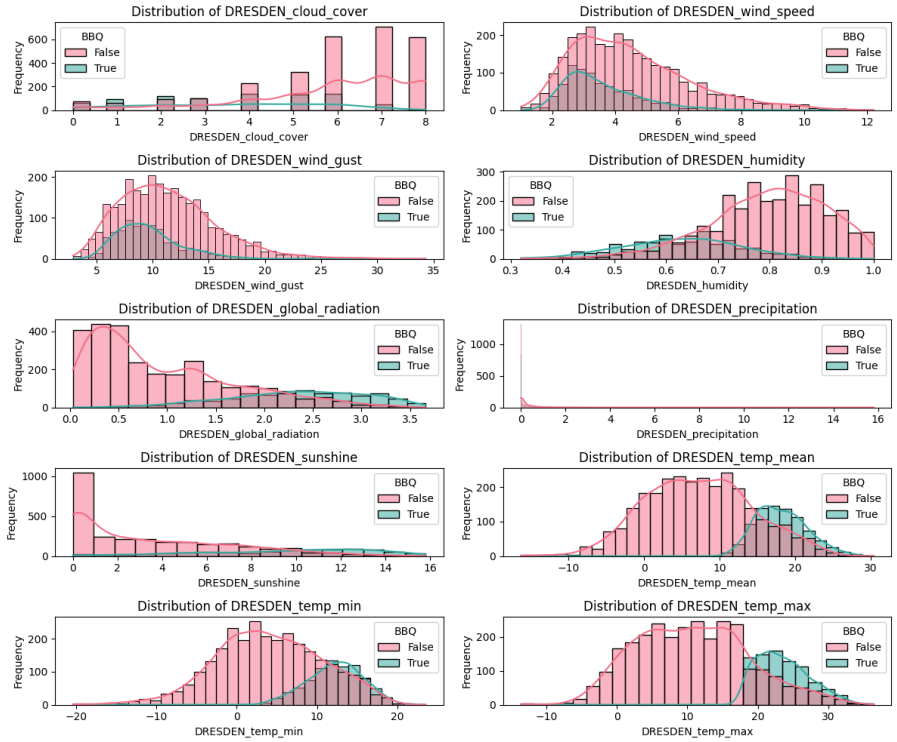



```
# Pairplot
sns.pairplot(dresden_weather_df, hue='BBQ', palette='husl')
plt.title('Pairplot of Dresden Weather Variables')
plt.show()
```

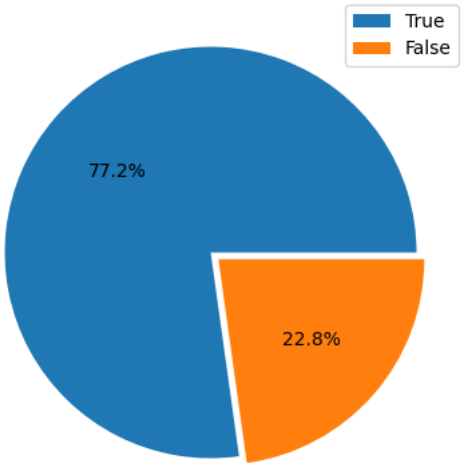



```
# Distribution plots for weather variables
plt.figure(figsize=(12, 10))
plt.subplots_adjust(hspace=0.5)
variables = ['DRESDEN_cloud_cover', 'DRESDEN_wind_speed', 'DRESDEN_wind_gust',
             'DRESDEN_humidity', 'DRESDEN_global_radiation', 'DRESDEN_precipitation',
             'DRESDEN_sunshine', 'DRESDEN_temp_mean', 'DRESDEN_temp_min', 'DRESDEN_temp_max']

for i, var in enumerate(variables, 1):
    plt.subplot(5, 2, i)
    sns.histplot(data=dresden_weather_df, x=var, kde=True, hue='BBQ', palette='husl')
    plt.title(f'Distribution of {var}')
    plt.xlabel(var)
    plt.ylabel('Frequency')
```

```
plt.tight_layout()
plt.show()
```



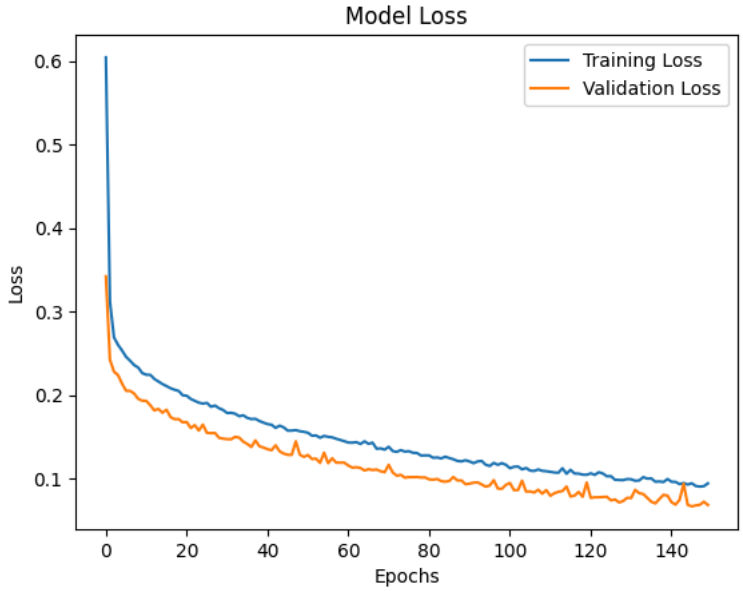

```
# Piechart for BBQ weather
plt.figure(figsize=(5, 5))
explode = [0, 0.05]
plt.pie(dresden_weather_df['BBQ'].value_counts(), explode=explode, autopct='%.1f%%')
plt.legend(['True', 'False'])
plt.show()
```




```
# Data Preparation
X = dresden_weather_df.drop(['DATE','BBQ'], axis=1)
y = dresden_weather_df['BBQ']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=28)
```

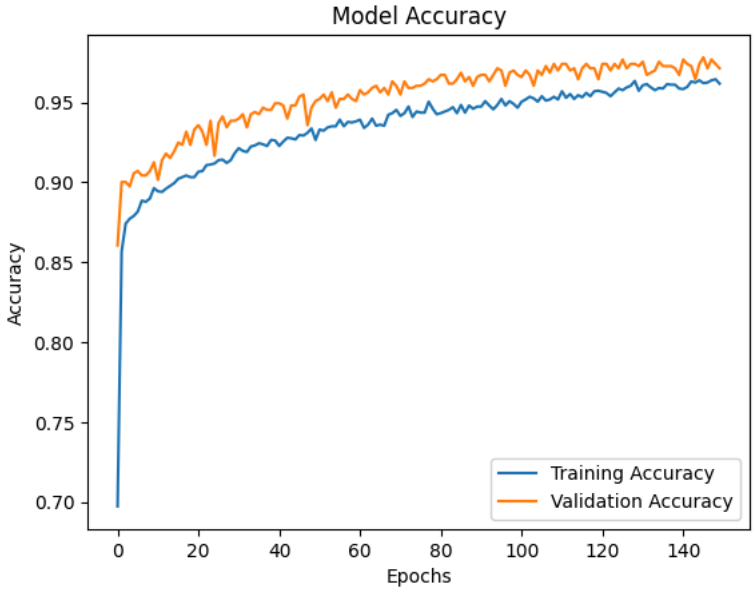
```
# ANN Model Building
model = Sequential([
    Dense(X_train.shape[1], activation="relu"),
    Dense(X_train.shape[1] // 2, activation="relu"),
    Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
history = model.fit(x=X_train, y=y_train, epochs=150, validation_data=(X_test, y_test), verbose=0)
```

```
# Plotting Loss and Accuracy
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
```

```
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

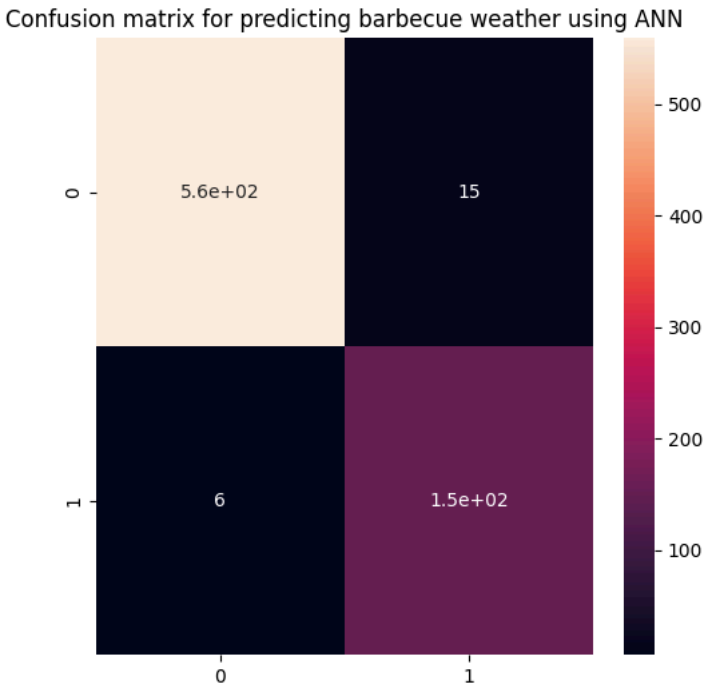



```
# Model Evaluation
predictions = np.round(model.predict(X_test))
print(classification_report(y_test, predictions))
```

```
23/23 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

       False       0.99      0.97      0.98       574
        True       0.91      0.96      0.93       157

    accuracy                           0.97       731
   macro avg       0.95      0.97      0.96       731
weighted avg       0.97      0.97      0.97       731
```

```
# Calculate accuracy for ANN predictions
ann_acc = accuracy_score(y_test, predictions)
```

```
# Visualize confusion matrix for ANN predictions
plt.figure(figsize=(6, 6))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True)
plt.title('Confusion matrix for predicting barbecue weather using ANN')
plt.show()
```



CNN Model Building

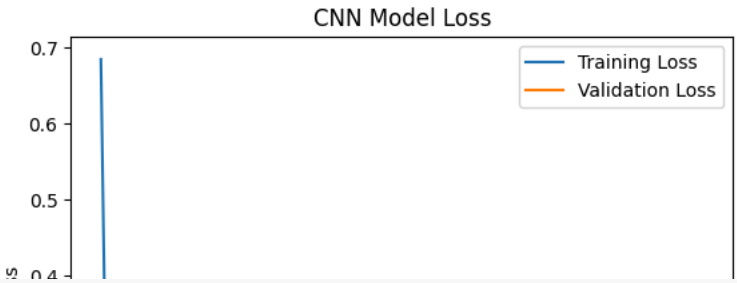
```
# Reshape the input data for CNN
X_train_reshaped = X_train.values.reshape((X_train.shape[0], 1, X_train.shape[1], 1))
X_test_reshaped = X_test.values.reshape((X_test.shape[0], 1, X_test.shape[1], 1))
```

```
# Define the CNN model
model_cnn = Sequential([
    Reshape((1, X_train.shape[1], 1)),
    Conv1D(filters=10, kernel_size=2, activation='relu', input_shape=(1, X_train.shape[1], 1)),
    Flatten(),
    Dense(1, activation='sigmoid')
])
```

```
# Compile the CNN model
model_cnn.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
# Train the CNN model
history_cnn = model_cnn.fit(x=X_train_reshaped, y=y_train, epochs=150, validation_data=(X_test_reshaped, y_test), verbose=0)
```

```
# Plot Loss and Accuracy for CNN model
plt.plot(history_cnn.history['loss'], label='Training Loss')
plt.plot(history_cnn.history['val_loss'], label='Validation Loss')
plt.title('CNN Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
plt.plot(history_cnn.history['accuracy'], label='Training Accuracy')
```