Refactored Capstone Notebook

What Changed

The core logic, features, and models remain the same, but the following improvements were made:

Key Improvements

- All preprocessing moved into **ColumnTransformer + Pipeline**
- **SMOTE** applied *inside* pipelines (no data leakage)
- Consistent **Stratified Cross-Validation**
- Unified evaluation across models
- Entire pipelines saved for deployment

1. Imports

In [1]:
```
pip install optuna
```

```
Collecting optuna
  Downloading optuna-4.7.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: alembic>=1.5.0 in /usr/local/lib/python3.12/dist-p
ackages (from optuna) (1.18.3)
Collecting colorlog (from optuna)
  Downloading colorlog-6.10.1-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
(from optuna) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-
packages (from optuna) (26.0)
Requirement already satisfied: sqlalchemy>=1.4.2 in /usr/local/lib/python3.12/dis
t-packages (from optuna) (2.0.46)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (f
rom optuna) (4.67.3)
Requirement already satisfied: PyYAML in /usr/local/lib/python3.12/dist-packages
(from optuna) (6.0.3)
Requirement already satisfied: Mako in /usr/local/lib/python3.12/dist-packages (f
rom alembic>=1.5.0->optuna) (1.3.10)
Requirement already satisfied: typing-extensions>=4.12 in /usr/local/lib/python3.
12/dist-packages (from alembic>=1.5.0->optuna) (4.15.0)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.12/dist-pack
ages (from sqlalchemy>=1.4.2->optuna) (3.3.1)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.12/dis
t-packages (from Mako->alembic>=1.5.0->optuna) (3.0.3)
Downloading optuna-4.7.0-py3-none-any.whl (413 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 413.9/413.9 kB 10.7 MB/s eta 0:00:00
Downloading colorlog-6.10.1-py3-none-any.whl (11 kB)
Installing collected packages: colorlog, optuna
Successfully installed colorlog-6.10.1 optuna-4.7.0
```

In [2]:
```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import optuna
import xgboost as xgb
import lightgbm as lgb
import joblib
import warnings
warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val
from sklearn.preprocessing import MinMaxScaler, OneHotEncoder, StandardScaler, F
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_sco
from sklearn.feature_selection import mutual_info_classif, SelectKBest
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier,
from sklearn.svm import SVC
from scipy.stats import f_oneway,ttest_ind
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE
```

2. Data Loading

In [3]:
```
# Mount Google Drive to access files stored in my google Drive

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:
```
def load_data(train_path, test_path):
    """Load train and test datasets from given file paths."""
    train_df = pd.read_csv(train_path)
    test_df = pd.read_csv(test_path)
    return train_df, test_df

df_train, df_test = load_data('/content/drive/MyDrive/Capstone Project Data/trai
                              '/content/drive/MyDrive/Capstone Project Data/test

# Display samples
print("Train Sample:")
display(df_train.head())
print("Test Sample:")
display(df_test.head())

# Display shapes
print(f"Train shape: {df_train.shape}")
print(f"Test shape: {df_test.shape}")
```

```
Train Sample:
```

	id	day	pressure	maxtemp	temparature	mintemp	dewpoint	humidity	cloud	sun
0	0	1	1017.4	21.2	20.6	19.9	19.4	87.0	88.0	
1	1	2	1019.5	16.2	16.9	15.8	15.4	95.0	91.0	
2	2	3	1024.1	19.4	16.1	14.6	9.3	75.0	47.0	
3	3	4	1013.4	18.1	17.8	16.9	16.8	95.0	95.0	
4	4	5	1021.8	21.3	18.4	15.2	9.6	52.0	45.0	

```
Test Sample:
```

	id	day	pressure	maxtemp	temparature	mintemp	dewpoint	humidity	cloud	s
0	2190	1	1019.5	17.5	15.8	12.7	14.9	96.0	99.0	
1	2191	2	1016.5	17.5	16.5	15.8	15.1	97.0	99.0	
2	2192	3	1023.9	11.2	10.4	9.4	8.9	86.0	96.0	
3	2193	4	1022.9	20.6	17.3	15.2	9.5	75.0	45.0	
4	2194	5	1022.2	16.1	13.8	6.4	4.3	68.0	49.0	

```
Train shape: (2190, 13)
Test shape: (730, 12)
```

3. Basic Info & Summary

In [5]:

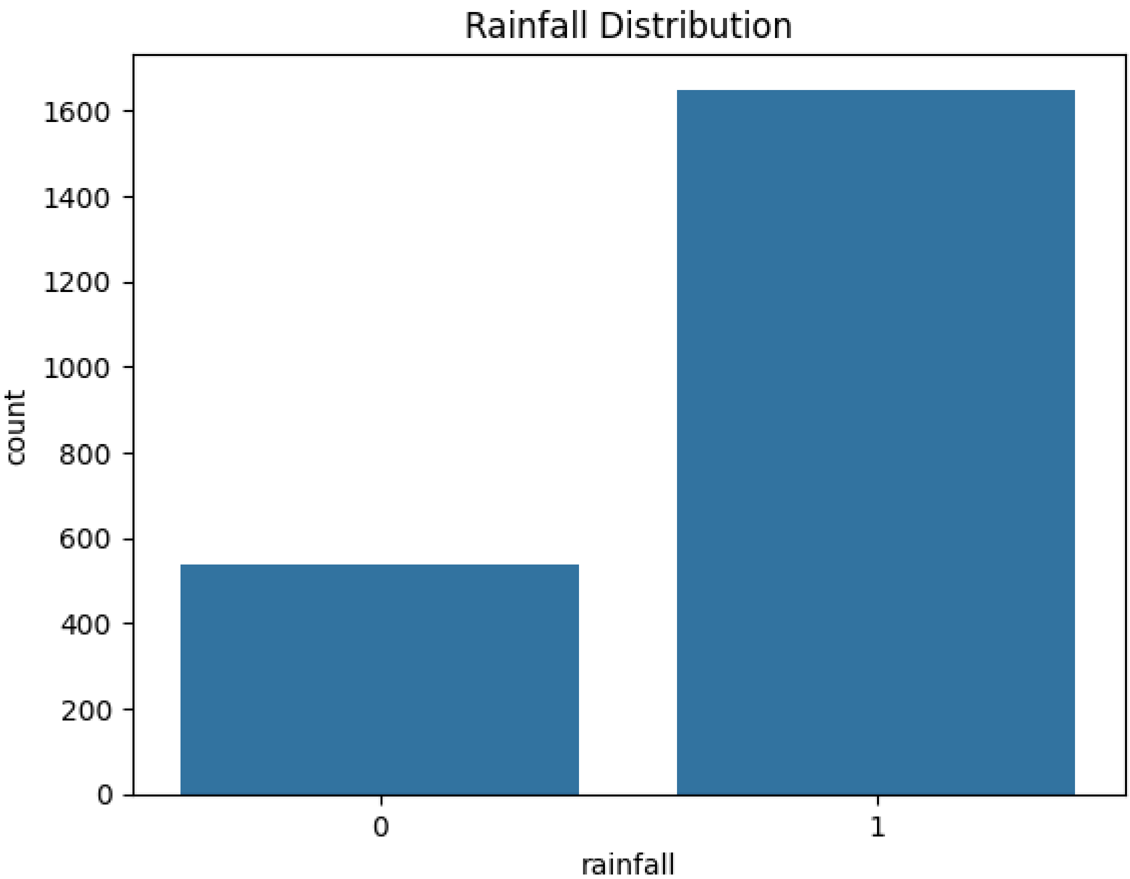
```
# Check data types
print(df_train.info())

# Summary statistics
print(df_train.describe())

# Check for missing values
print(df_train.isnull().sum())
print(df_test.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2190 entries, 0 to 2189
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             2190 non-null   int64
 1   day            2190 non-null   int64
 2   pressure       2190 non-null   float64
 3   maxtemp        2190 non-null   float64
 4   temparature    2190 non-null   float64
 5   mintemp        2190 non-null   float64
 6   dewpoint       2190 non-null   float64
 7   humidity       2190 non-null   float64
 8   cloud          2190 non-null   float64
 9   sunshine       2190 non-null   float64
 10  winddirection  2190 non-null   float64
 11  windspeed      2190 non-null   float64
 12  rainfall       2190 non-null   int64
dtypes: float64(10), int64(3)
memory usage: 222.6 KB
None
                id          day     pressure      maxtemp  temparature  \
count  2190.000000  2190.000000  2190.000000  2190.000000  2190.000000
mean   1094.500000   179.948402  1013.602146    26.365799    23.953059
std     632.342866   105.203592     5.655366     5.654330     5.222410
min       0.000000     1.000000   999.000000    10.400000     7.400000
25%     547.250000    89.000000  1008.600000    21.300000    19.300000
50%    1094.500000   178.500000  1013.000000    27.800000    25.500000
75%    1641.750000   270.000000  1017.775000    31.200000    28.400000
max    2189.000000   365.000000  1034.600000    36.000000    31.500000

           mintemp     dewpoint     humidity        cloud     sunshine  \
count  2190.000000  2190.000000  2190.000000  2190.000000  2190.000000
mean     22.170091    20.454566    82.036530    75.721918     3.744429
std       5.059120     5.288406     7.800654    18.026498     3.626327
min       4.000000    -0.300000    39.000000     2.000000     0.000000
25%      17.700000    16.800000    77.000000    69.000000     0.400000
50%      23.850000    22.150000    82.000000    83.000000     2.400000
75%      26.400000    25.000000    88.000000    88.000000     6.800000
max      29.800000    26.700000    98.000000   100.000000    12.100000

       winddirection    windspeed     rainfall
count    2190.000000  2190.000000  2190.000000
mean      104.863151    21.804703     0.753425
std        80.002416     9.898659     0.431116
min        10.000000     4.400000     0.000000
25%        40.000000    14.125000     1.000000
50%        70.000000    20.500000     1.000000
75%       200.000000    27.900000     1.000000
max       300.000000    59.500000     1.000000
id               0
day              0
pressure         0
maxtemp          0
temparature      0
mintemp          0
dewpoint         0
humidity         0
cloud            0
sunshine         0
```

```
winddirection    0
windspeed        0
rainfall         0
dtype: int64
id               0
day              0
pressure         0
maxtemp          0
temparature      0
mintemp          0
dewpoint         0
humidity         0
cloud            0
sunshine         0
winddirection    1
windspeed        0
dtype: int64
```

4.Target Variable Analysis

In [6]:
```
# Visualize the distribution of the target variable 'rainfall'

sns.countplot(x='rainfall', data=df_train)
plt.title("Rainfall Distribution")
plt.show()

# Percentage of each class
print(df_train['rainfall'].value_counts(normalize=True))
```



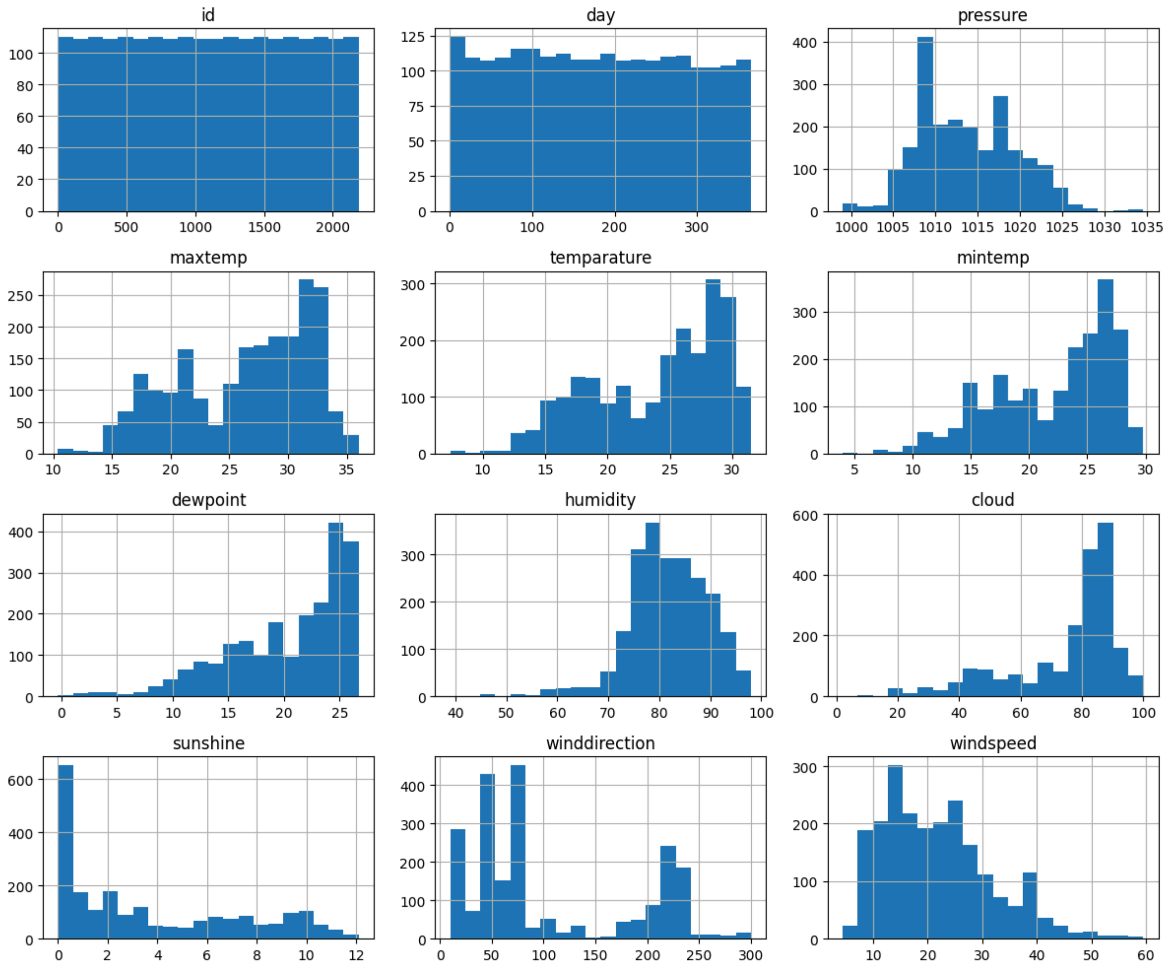
```
rainfall
1    0.753425
0    0.246575
Name: proportion, dtype: float64
```

In [7]:

```python
# Identify all numeric features for analysis

num_cols = df_train.select_dtypes(include='number').columns.tolist()
num_cols.remove('rainfall')  # exclude target
```

In [8]:

```python
# Histograms
df_train[num_cols].hist(figsize=(12, 10), bins=20)
plt.tight_layout()
plt.show()
```



In [9]:

```python
# Check skewness of numeric features
# Helps decide if transformations are needed (for linear/logistic models)
# Strongly skewed features: cloud, dewpoint (consider transforming if necessary)

df_train[num_cols].skew()
```

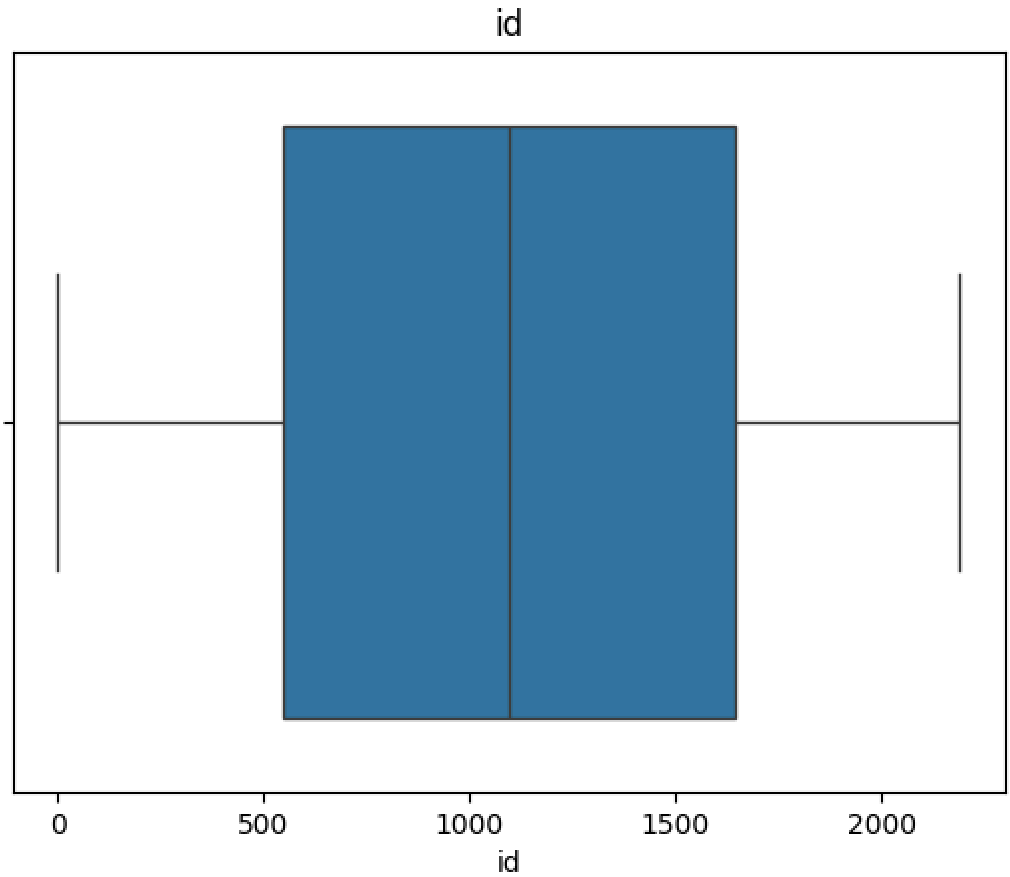
Out[9]:

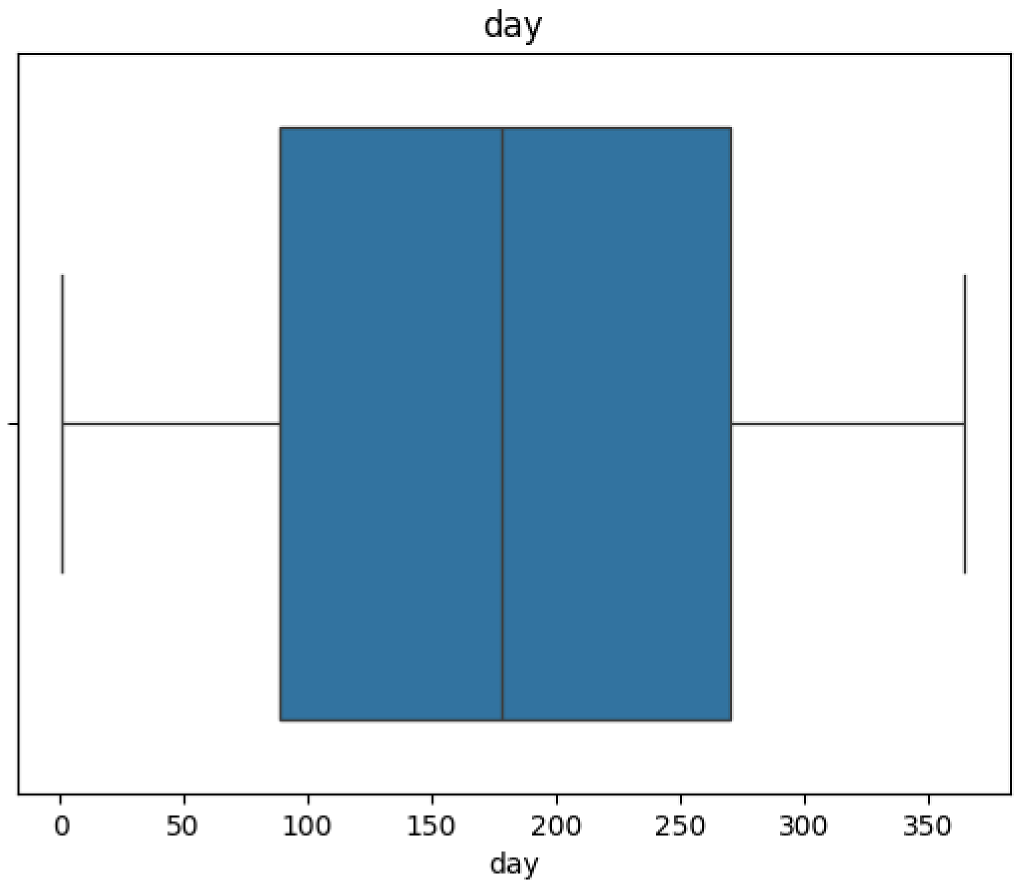
	0
id	0.000000
day	0.030615
pressure	0.284062
maxtemp	-0.490890
temparature	-0.557471
mintemp	-0.649179
dewpoint	-0.997889
humidity	-0.561541
cloud	-1.339274
sunshine	0.639871
winddirection	0.708063
windspeed	0.769390

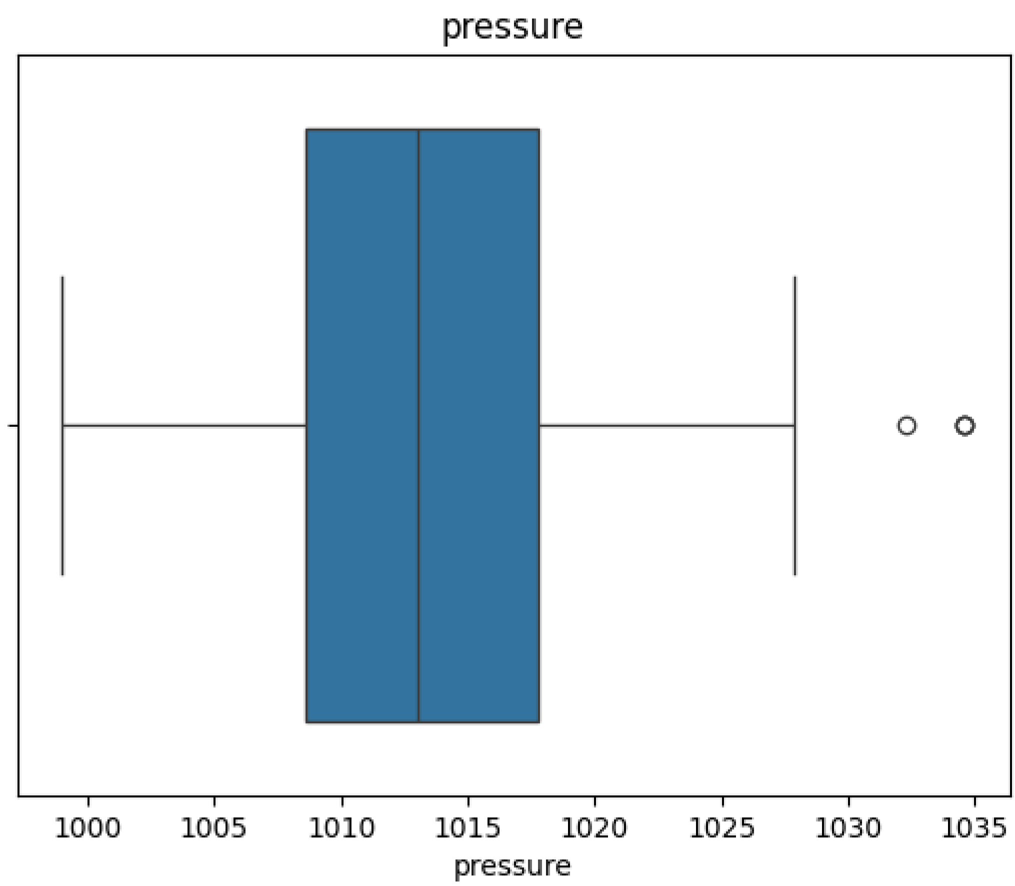
dtype: float64

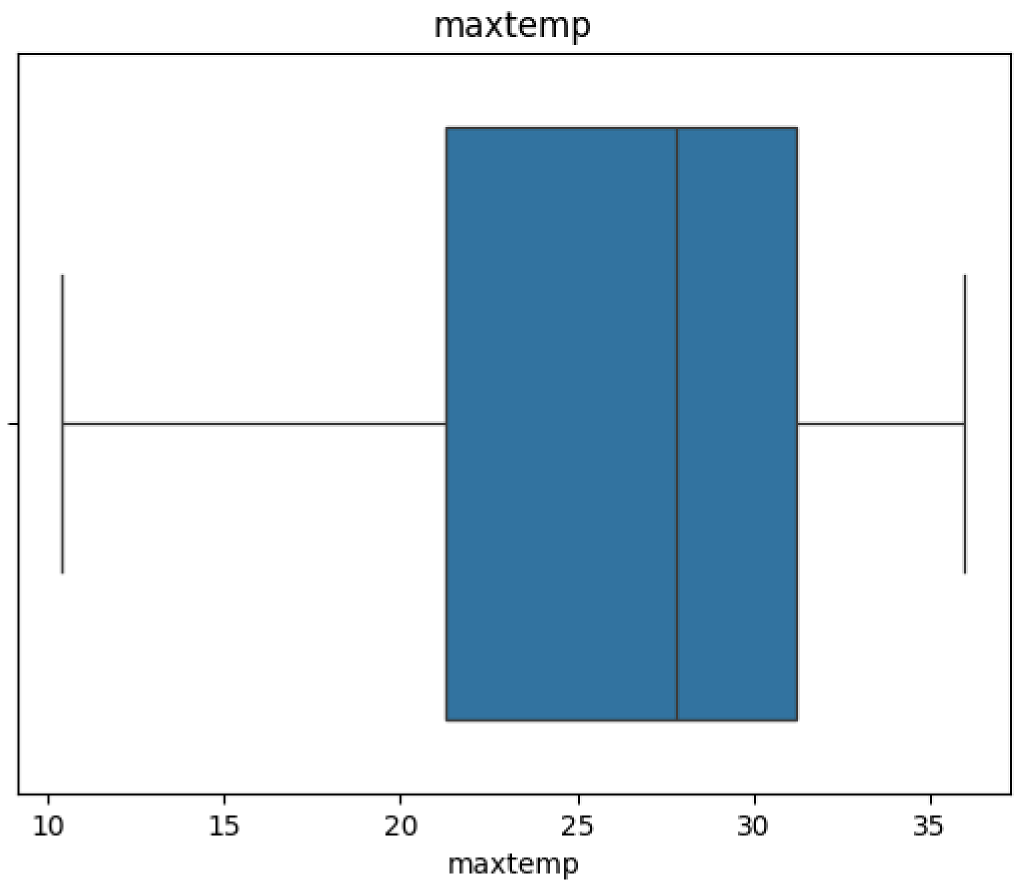
In [10]:

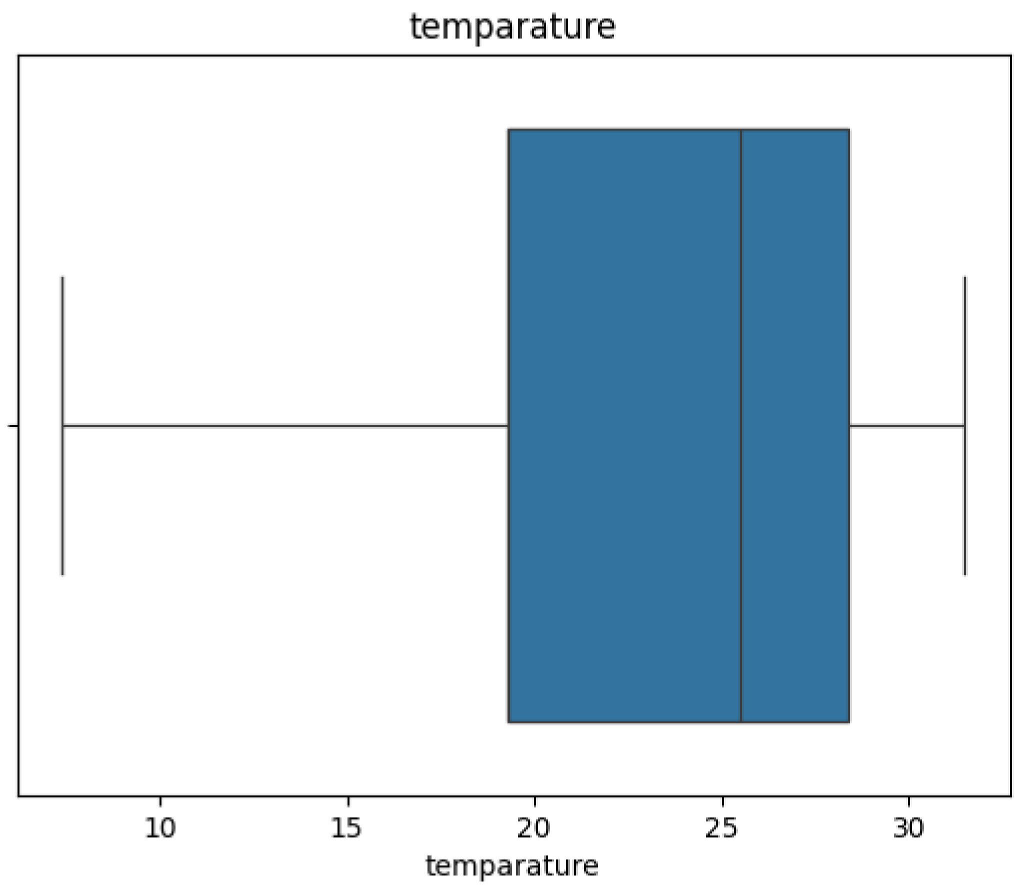
```
# Boxplots for outliers
for col in num_cols:
    sns.boxplot(x=df_train[col])
    plt.title(col)
    plt.show()
```

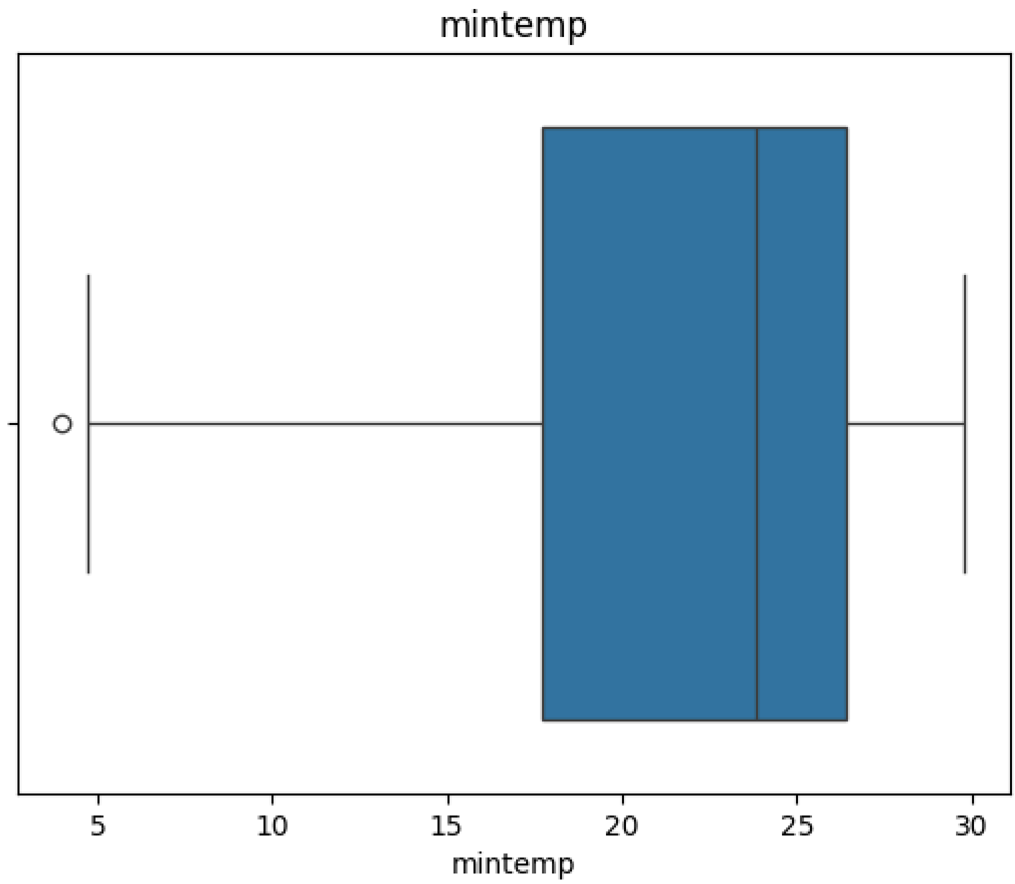


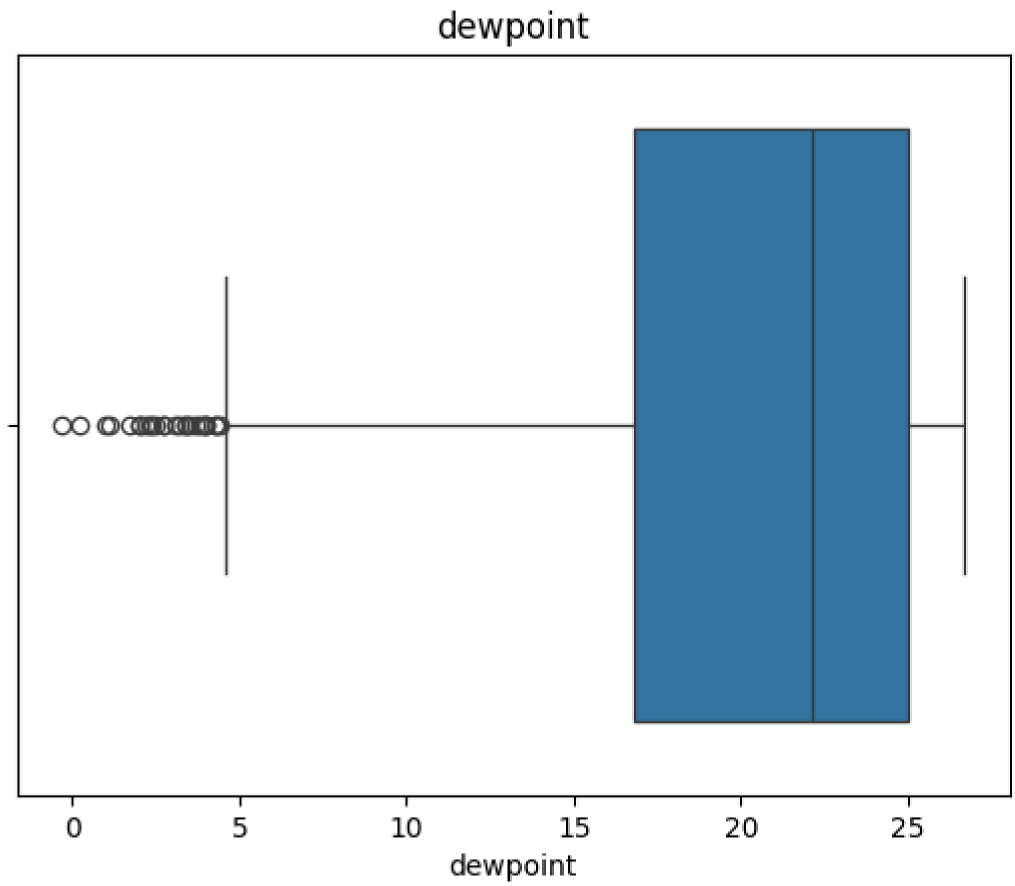

day
0
50
100
150
200
250
300
350
day

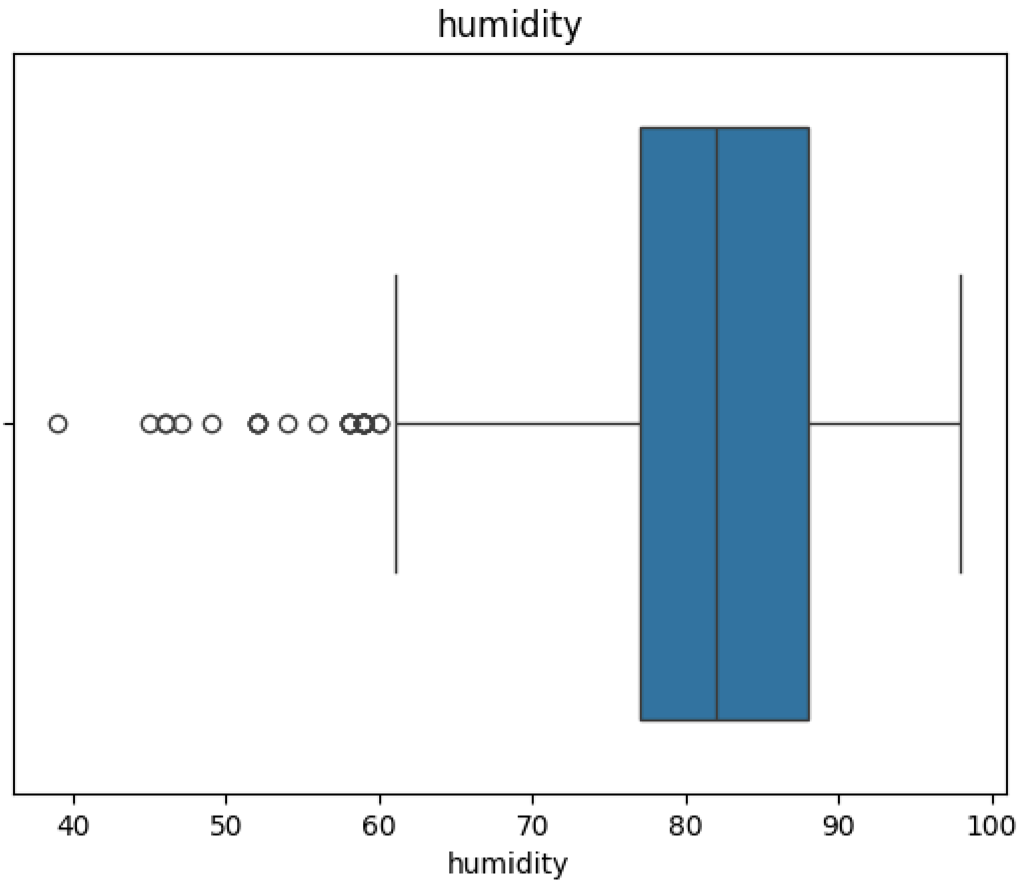

pressure
1000
1005
1010
1015
1020
1025
1030
1035
pressure

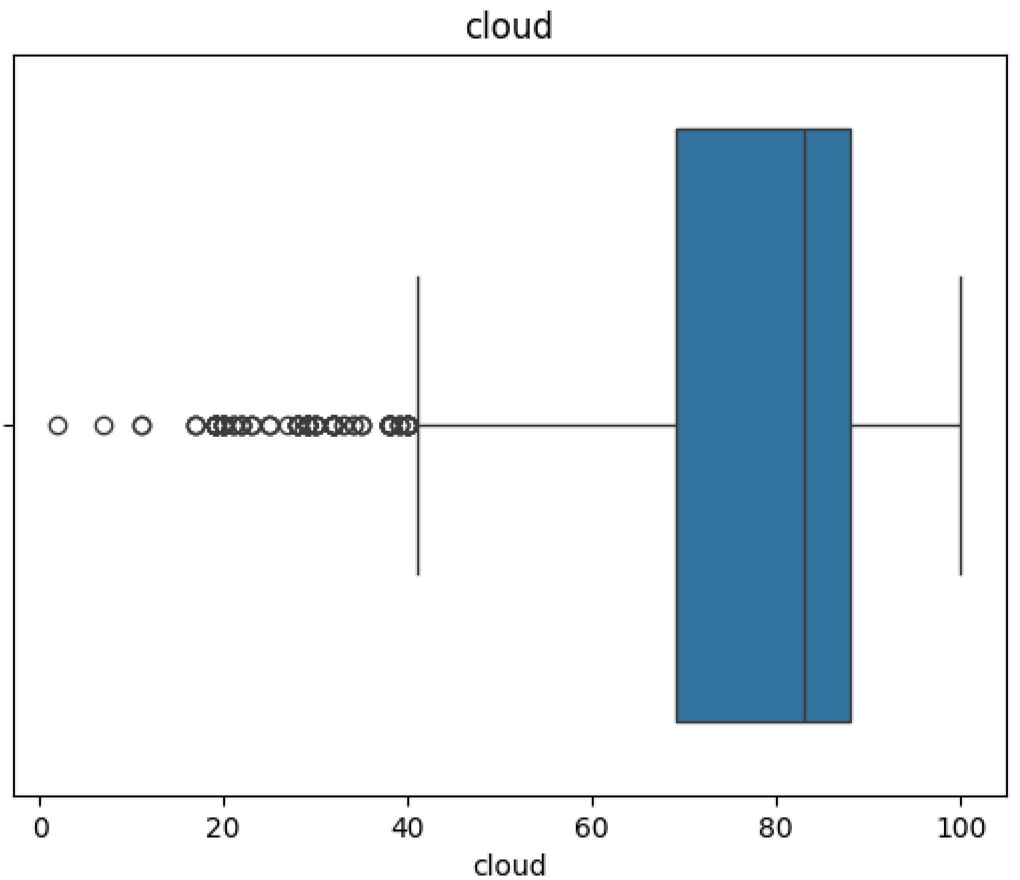

maxtemp
10
15
20
25
30
35
maxtemp

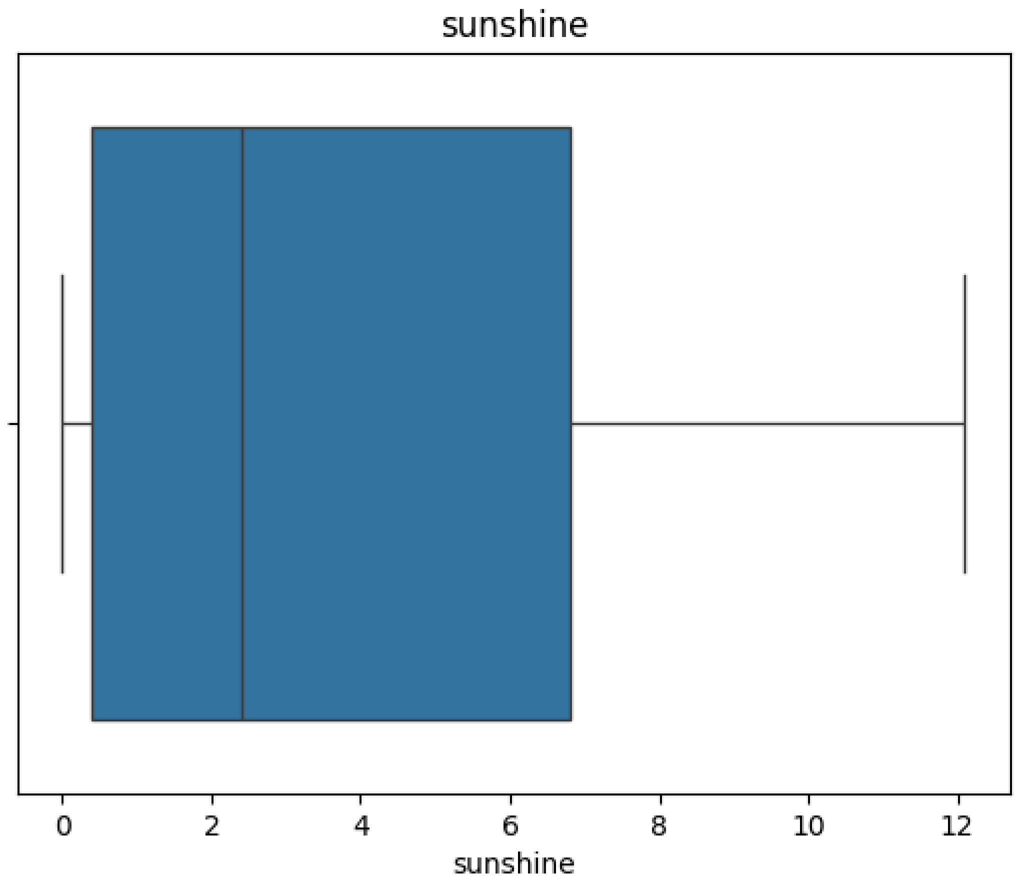

temparature
10
15
20
25
30
temparature

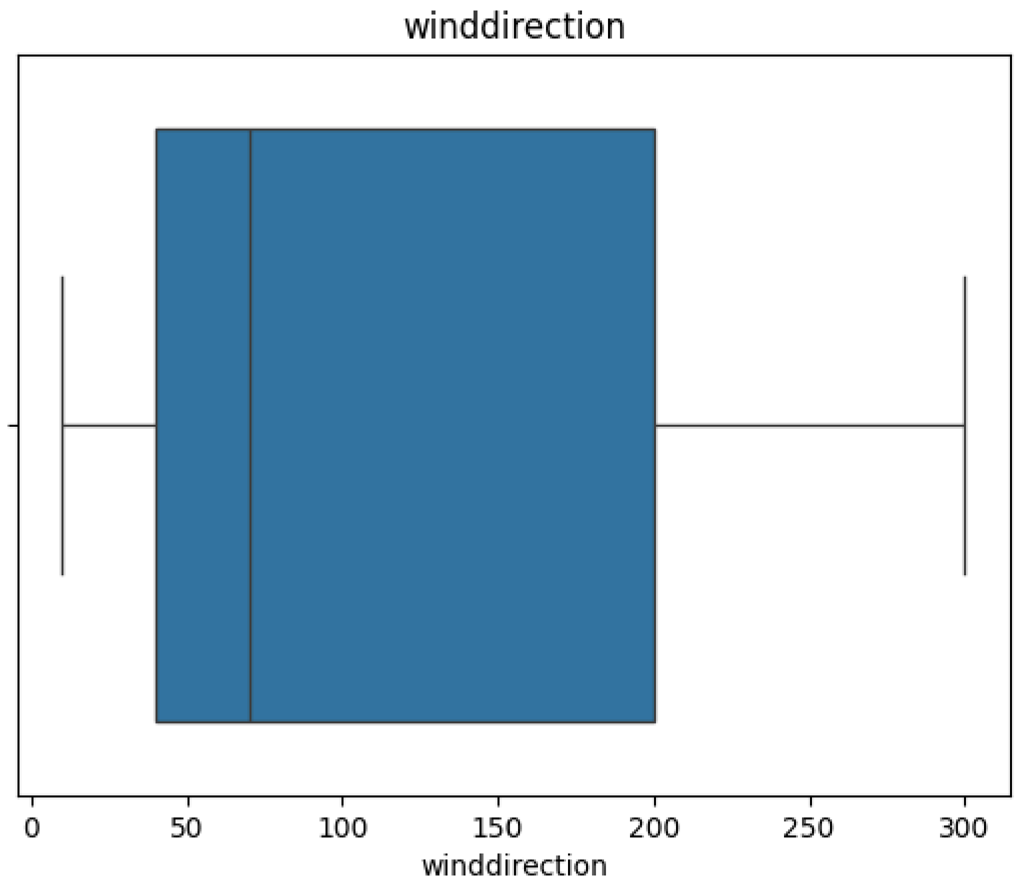

mintemp
5
10
15
20
25
30
mintemp

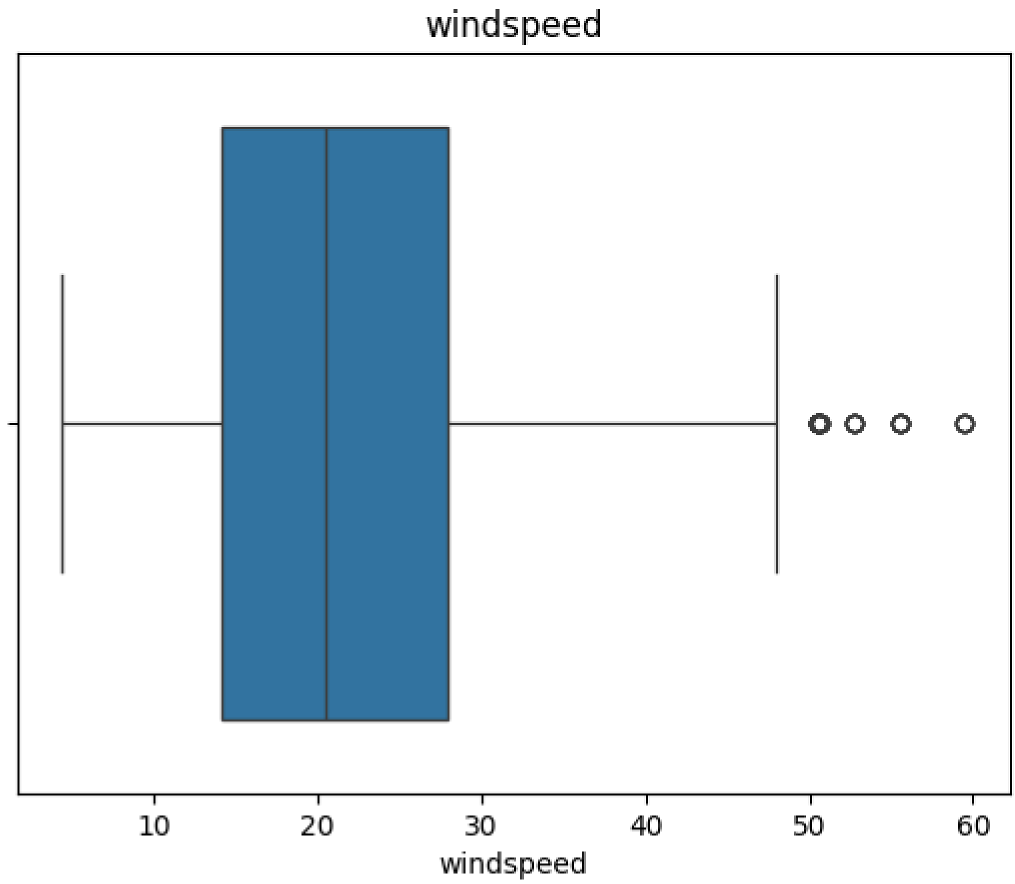

dewpoint
0
5
10
15
20
25
dewpoint


humidity
40
50
60
70
80
90
100
humidity


cloud
0
20
40
60
80
100
cloud


sunshine
0
2
4
6
8
10
12
sunshine

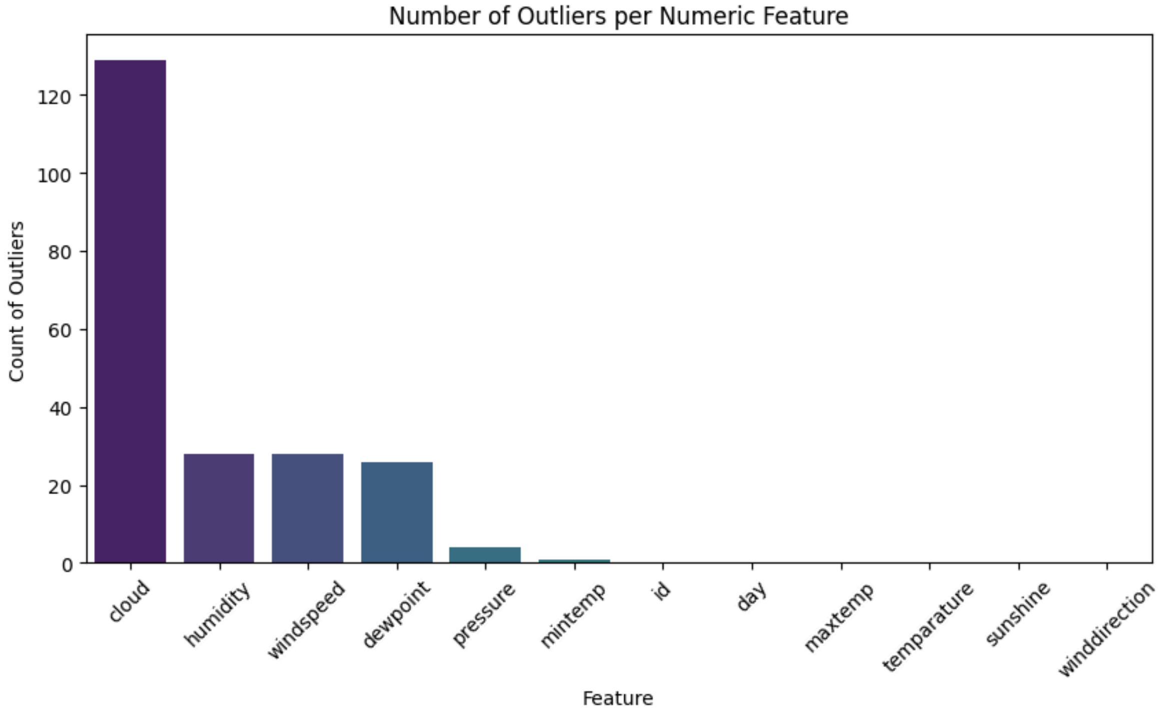

winddirection
0
50
100
150
200
250
300
winddirection



In [11]:
```
outlier_summary = []

for col in num_cols:
    Q1 = df_train[col].quantile(0.25)
    Q3 = df_train[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR
    outliers = df_train[(df_train[col] < lower) | (df_train[col] > upper)][col]
    outlier_summary.append({
        'Feature': col,
        'Num_Outliers': len(outliers),
        'Min_Outlier': outliers.min() if len(outliers) > 0 else None,
        'Max_Outlier': outliers.max() if len(outliers) > 0 else None,
    })

outlier_df = pd.DataFrame(outlier_summary).sort_values(by='Num_Outliers', ascend
display(outlier_df)
```

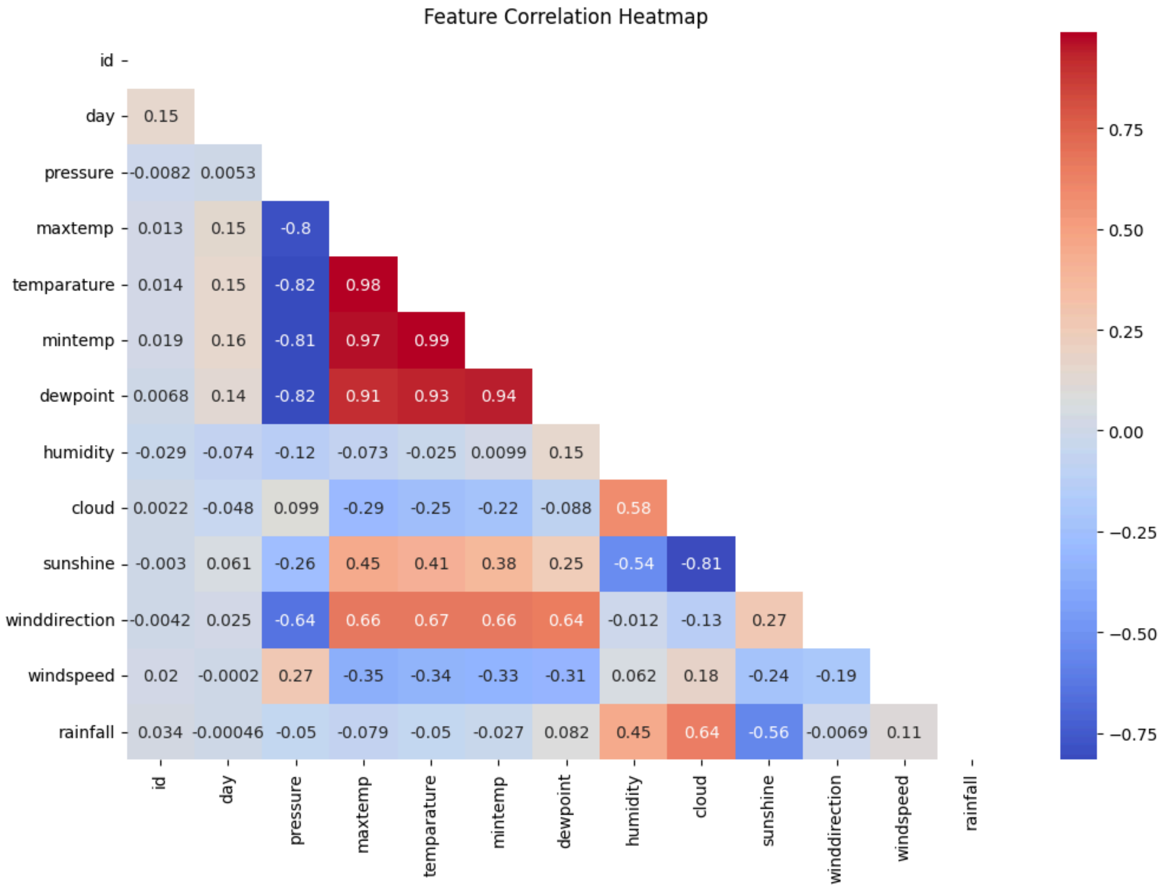
	Feature	Num_Outliers	Min_Outlier	Max_Outlier
8	cloud	129	2.0	40.0
7	humidity	28	39.0	60.0
11	windspeed	28	50.6	59.5
6	dewpoint	26	-0.3	4.4
2	pressure	4	1032.3	1034.6
5	mintemp	1	4.0	4.0
0	id	0	NaN	NaN
1	day	0	NaN	NaN
3	maxtemp	0	NaN	NaN
4	temparature	0	NaN	NaN
9	sunshine	0	NaN	NaN
10	winddirection	0	NaN	NaN

In [12]:
```
# Visualizing columns with the most outliers

plt.figure(figsize=(10,5))
sns.barplot(data=outlier_df, x='Feature', y='Num_Outliers', palette='viridis')
plt.title("Number of Outliers per Numeric Feature")
plt.xticks(rotation=45)
plt.ylabel("Count of Outliers")
plt.show()
```

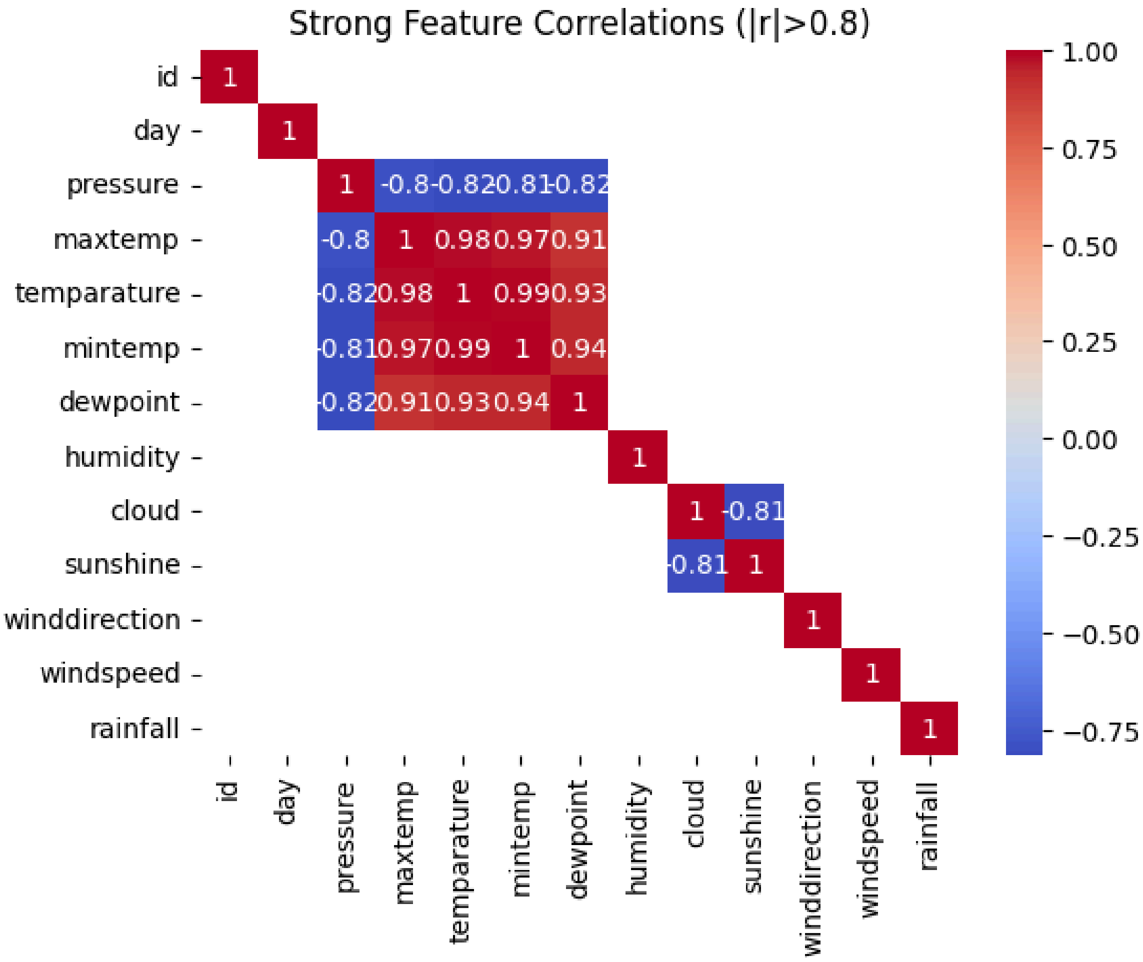


In [13]:
```
# Correlation heatmap

plt.figure(figsize=(12,8))
```

```python
corr = df_train[num_cols + ['rainfall']].corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, annot=True, cmap='coolwarm', mask=mask)
plt.title("Feature Correlation Heatmap")
plt.show()
```



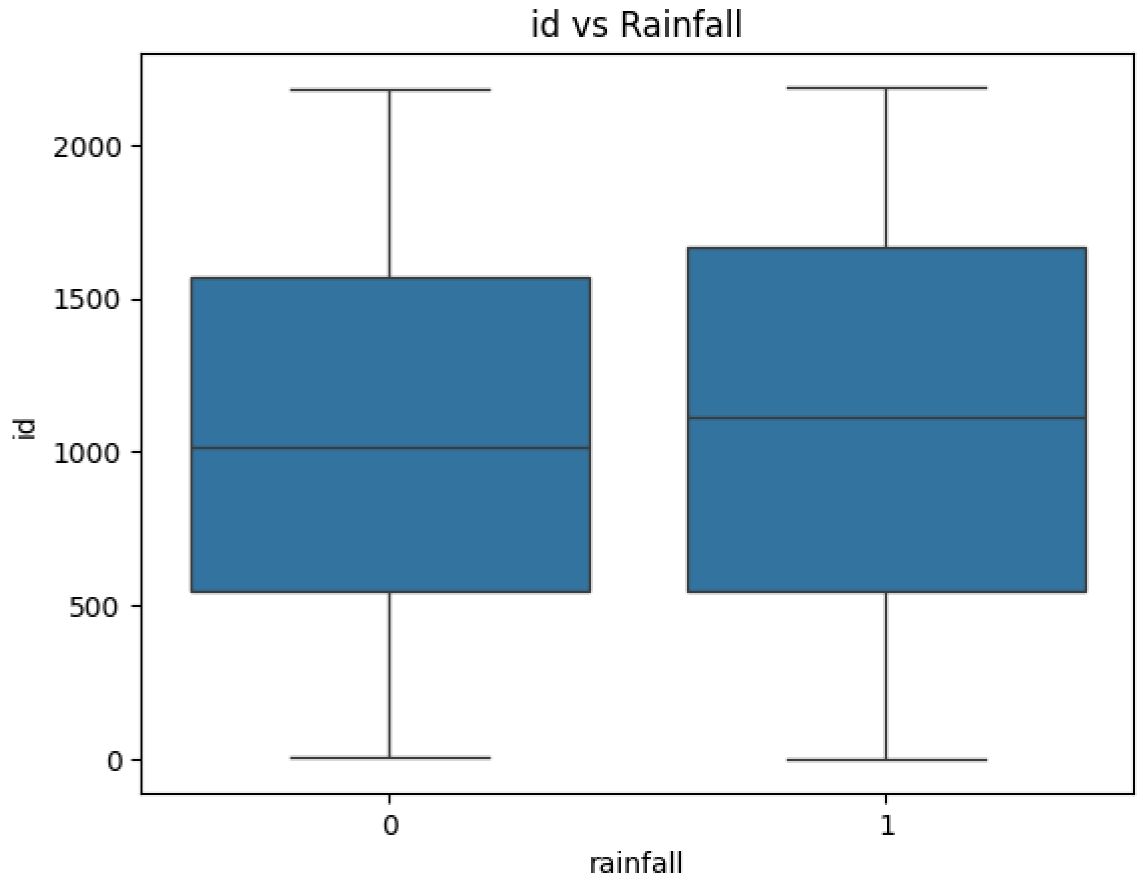
In [14]:
```python
# Highlighting Strong correlations

strong_corr = corr.abs() > 0.8
sns.heatmap(corr, annot=True, cmap='coolwarm', mask=~strong_corr)
plt.title("Strong Feature Correlations (|r|>0.8)")
plt.show()
```

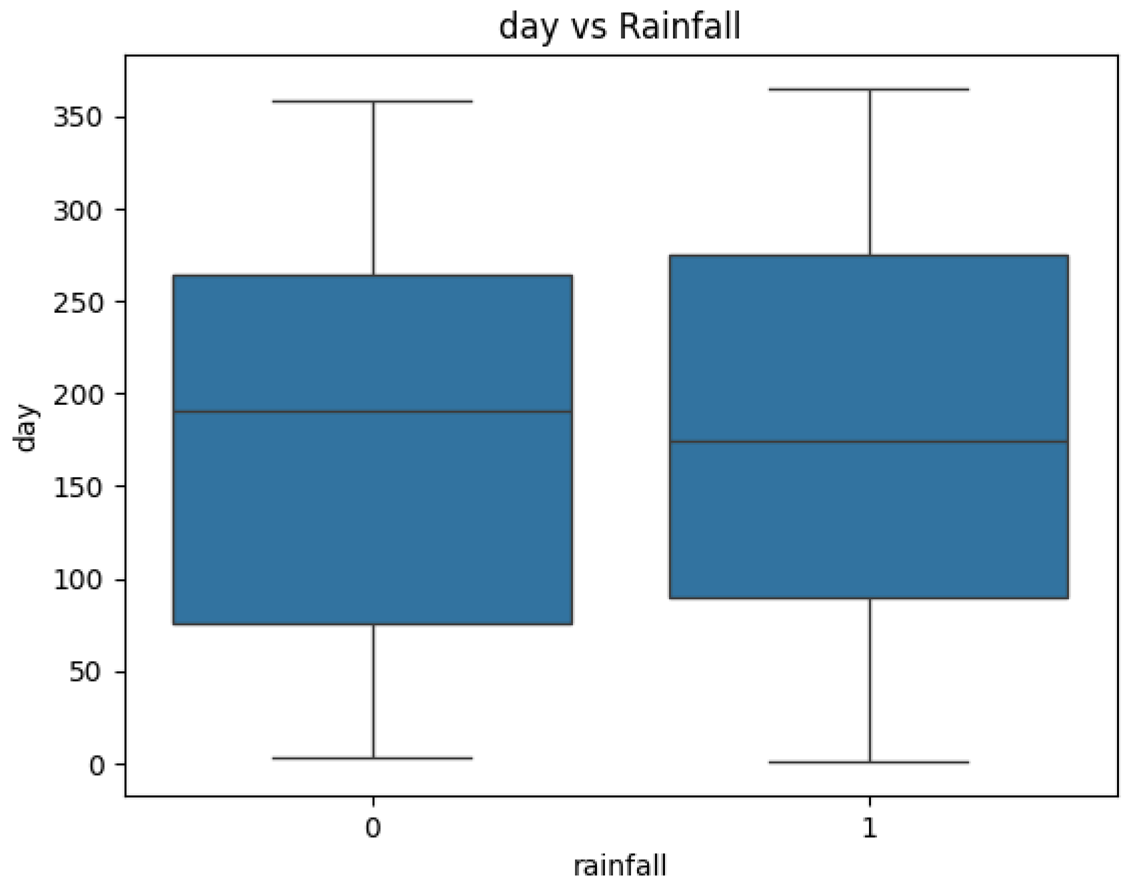


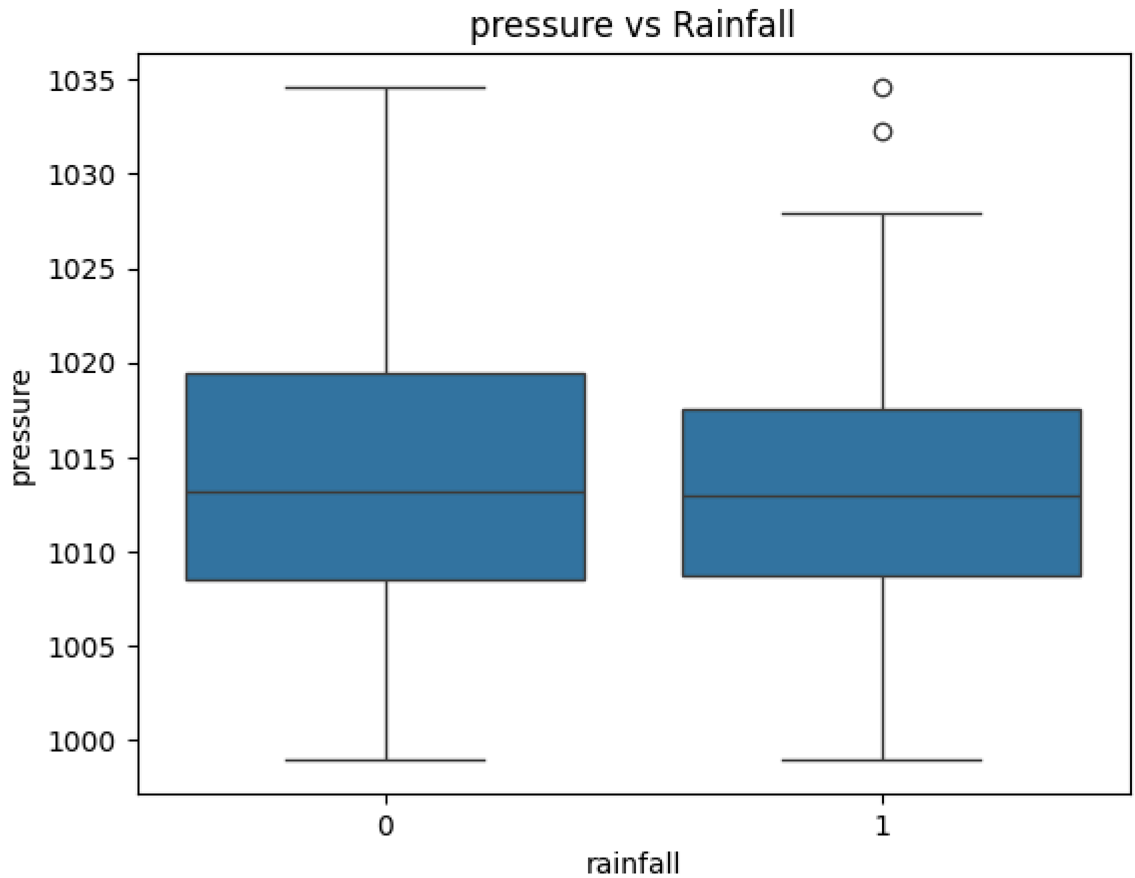
In [15]:
```
cat_cols = df_train.select_dtypes(include='object').columns.tolist()

for col in cat_cols:
    sns.countplot(y=col, data=df_train, order=df_train[col].value_counts().index
    plt.title(f"{col} Distribution")
    plt.show()
```

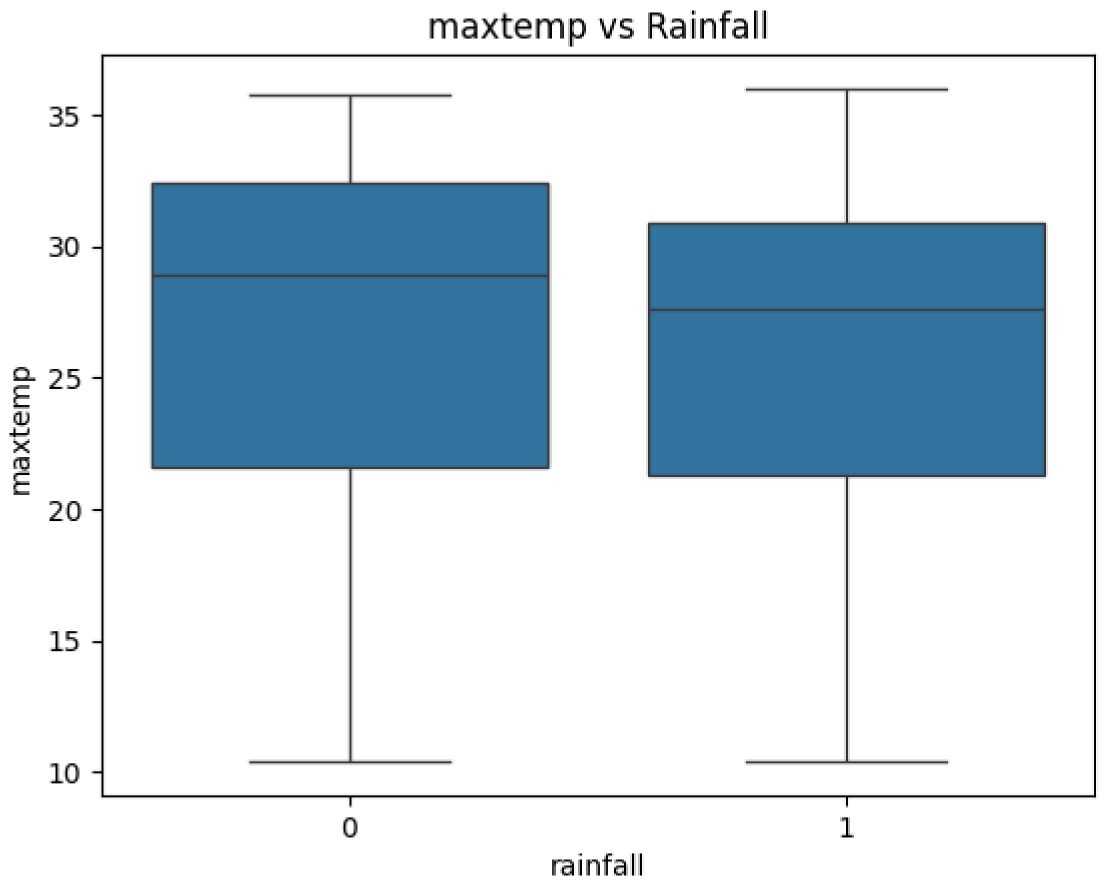
5.Feature Relationships with Target

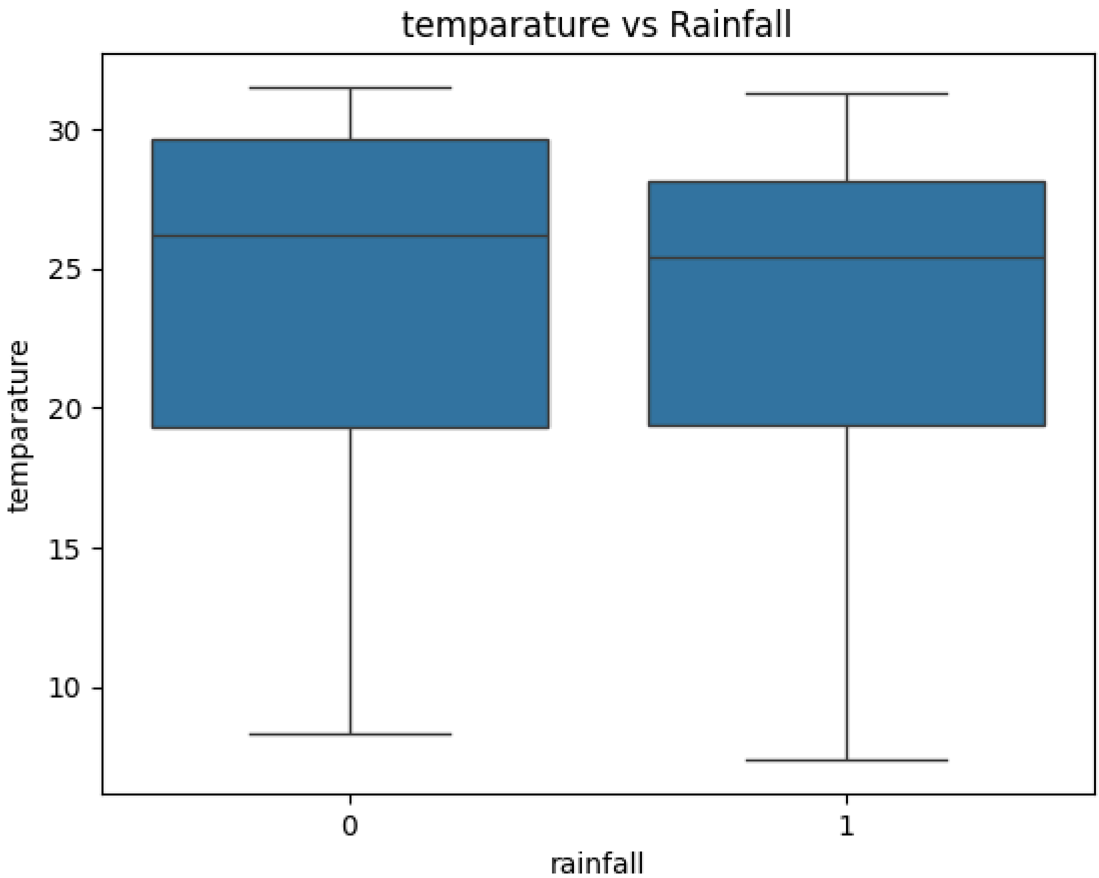
In [16]:
```
# Numerical vs Target
for col in num_cols:
    sns.boxplot(x='rainfall', y=col, data=df_train)
    plt.title(f"{col} vs Rainfall")
    plt.show()
```

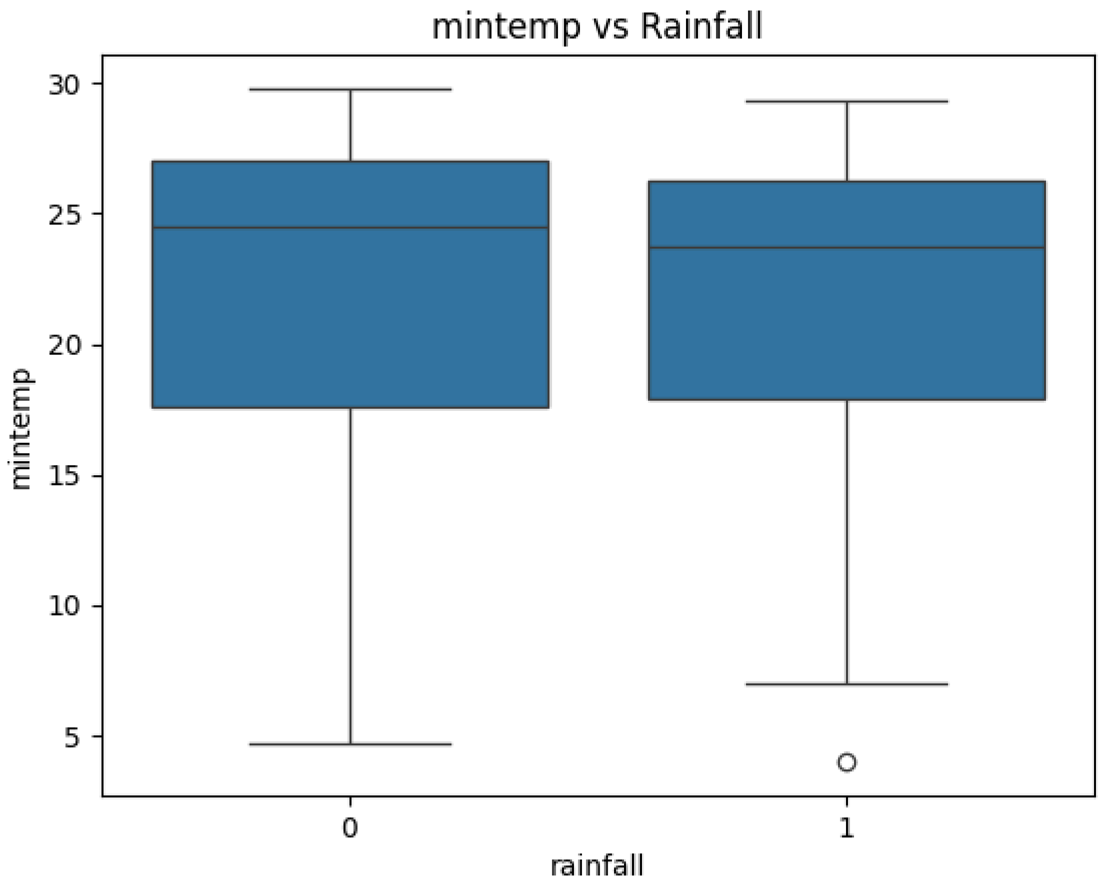

id vs Rainfall
id
2000
1500
1000
500
0
0
1
rainfall

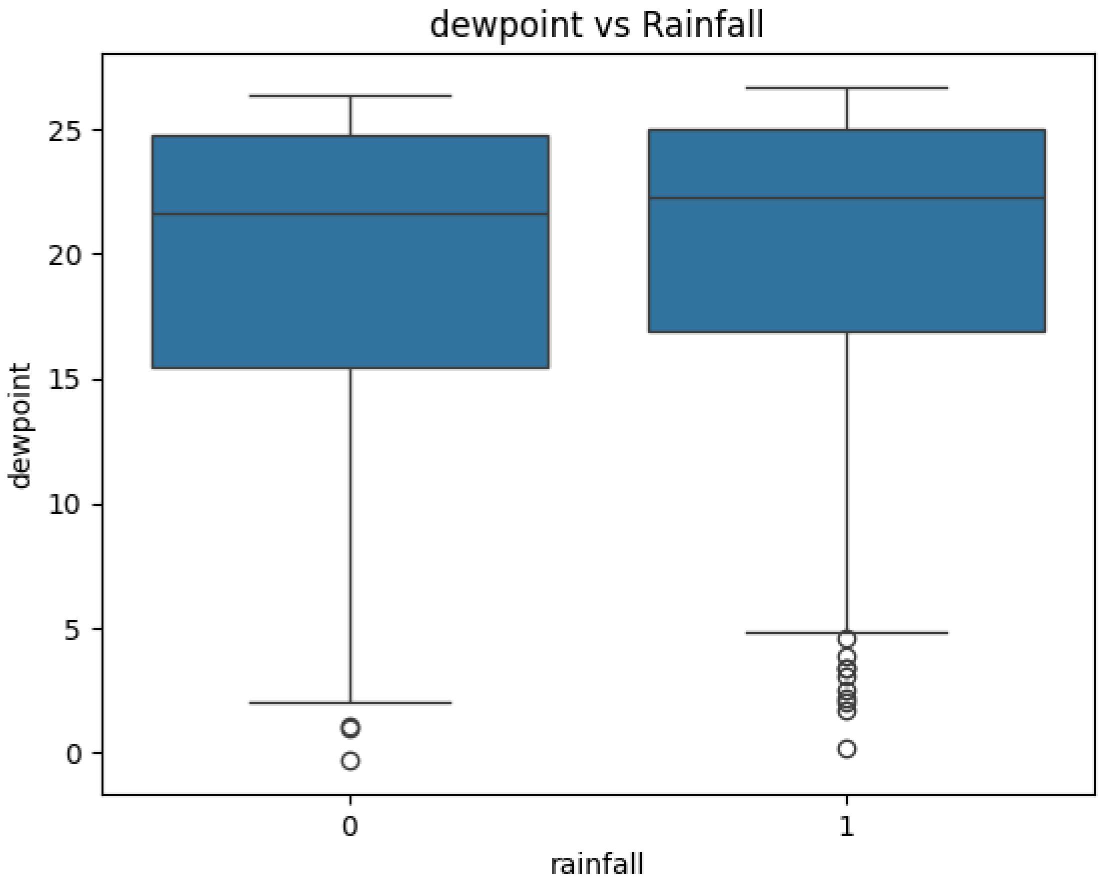

day vs Rainfall
day
350
300
250
200
150
100
50
0
0
1
rainfall

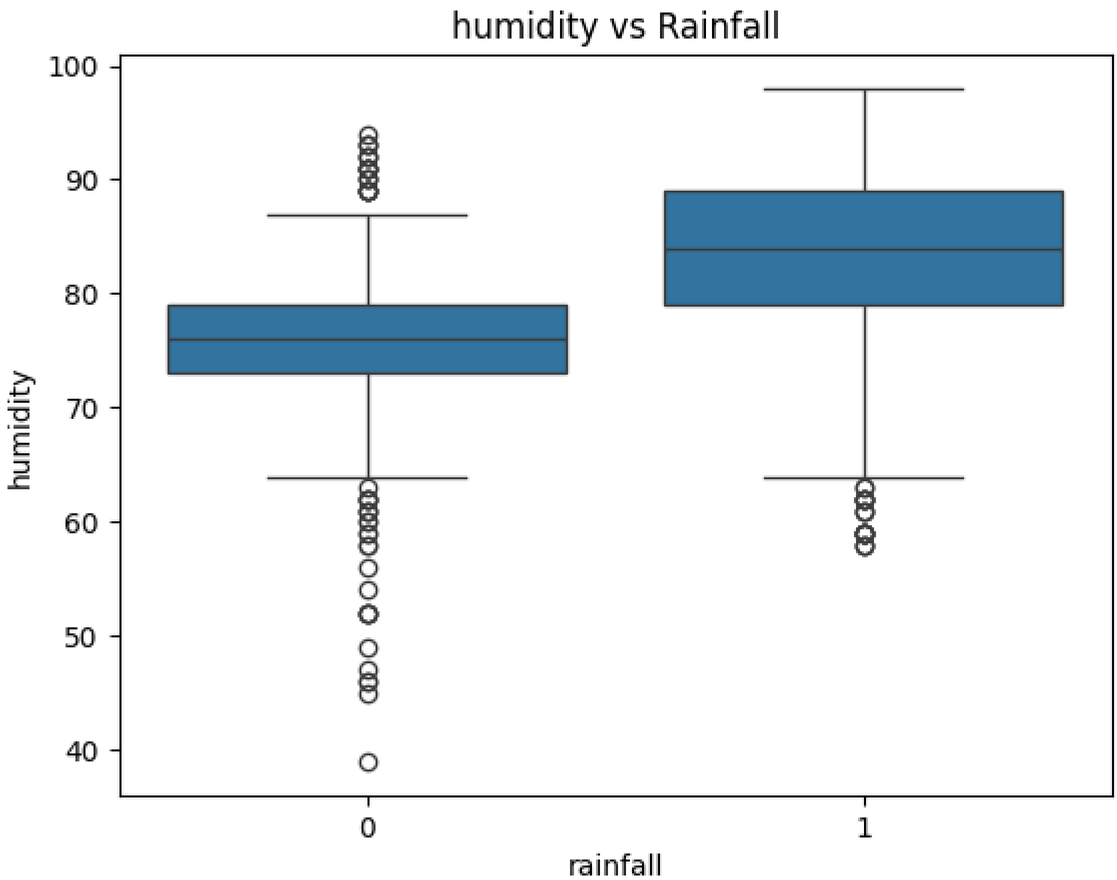

pressure vs Rainfall
1035
1030
1025
1020
1015
1010
1005
1000
pressure
0
1
rainfall

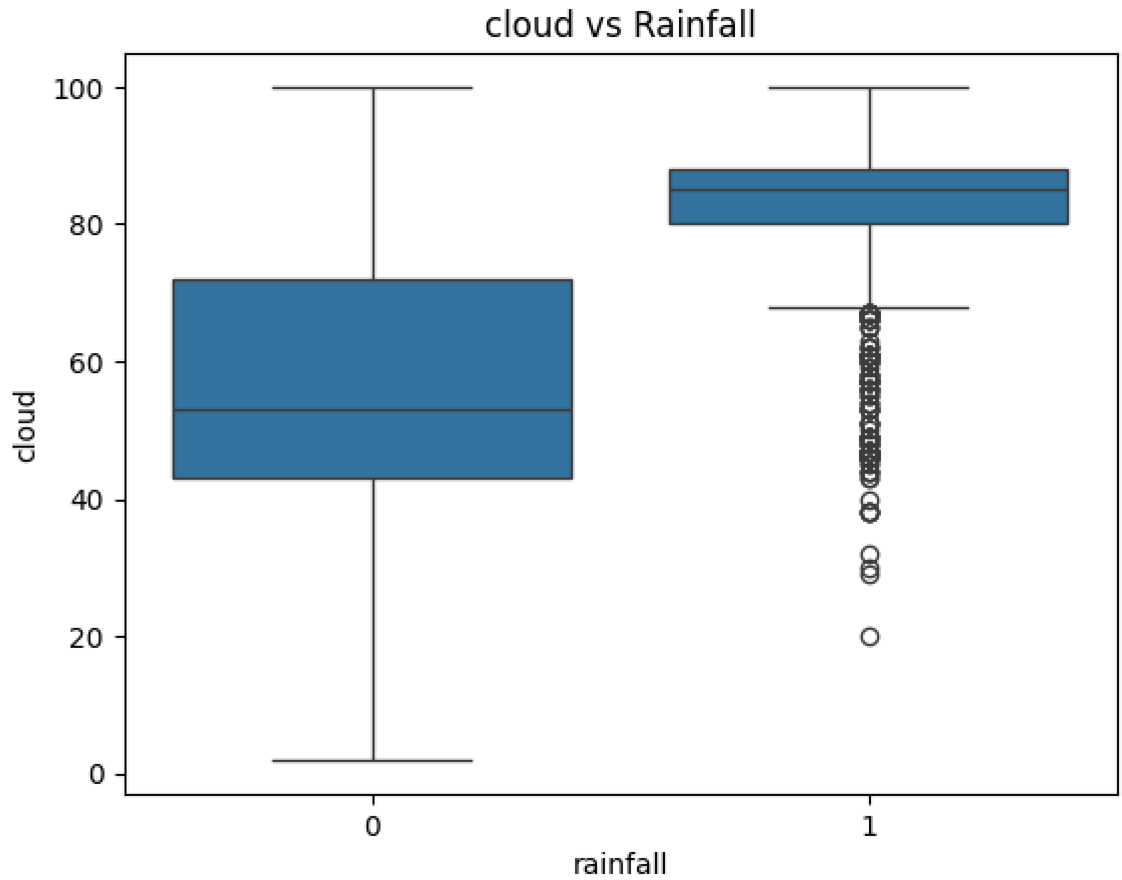

maxtemp vs Rainfall
35
30
25
20
15
10
maxtemp
0
1
rainfall

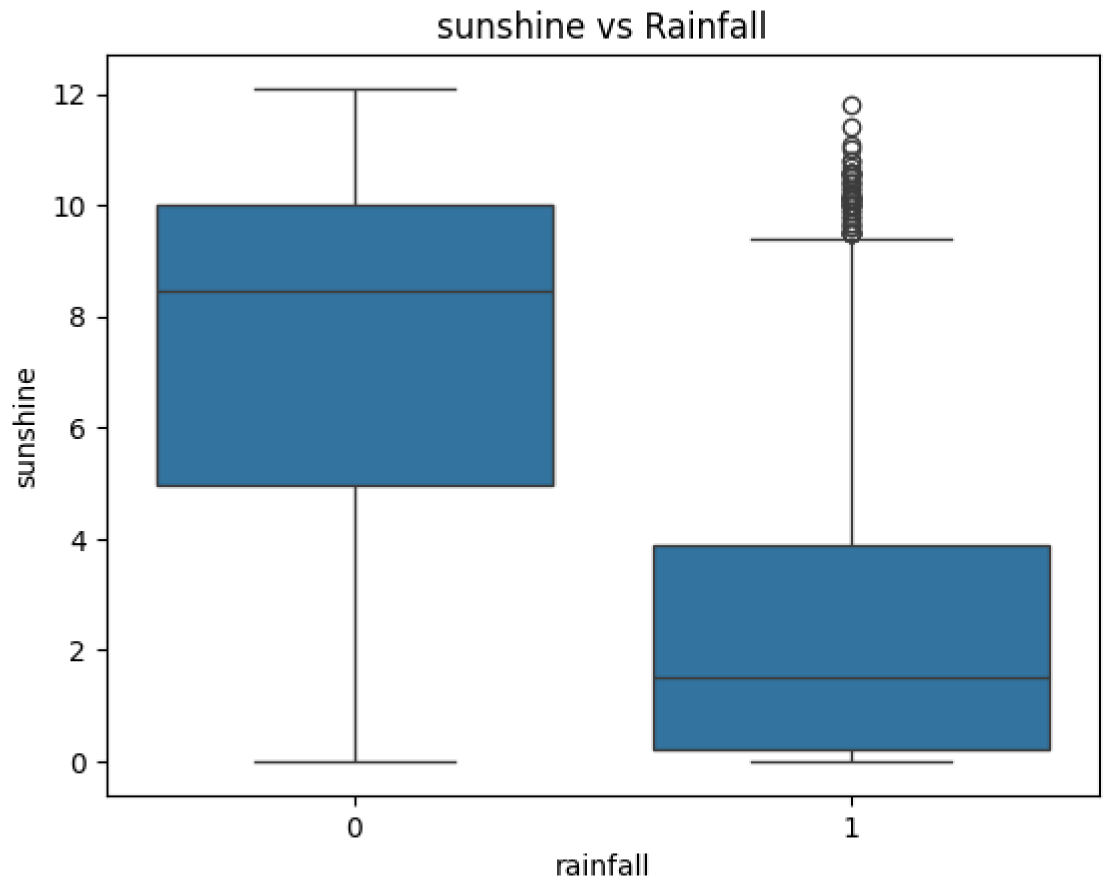

temparature vs Rainfall
30
25
20
15
10
temparature
0
1
rainfall

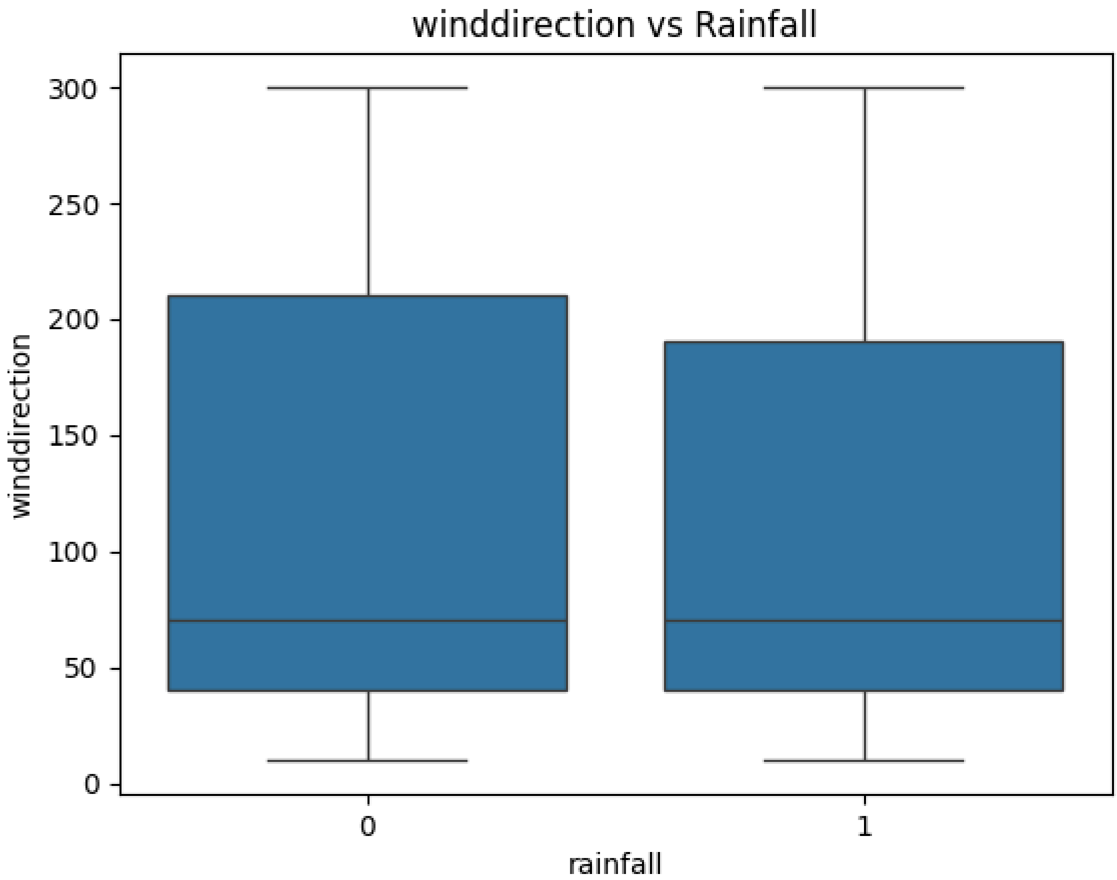

mintemp vs Rainfall
30
25
20
15
10
5
mintemp
0
1
rainfall

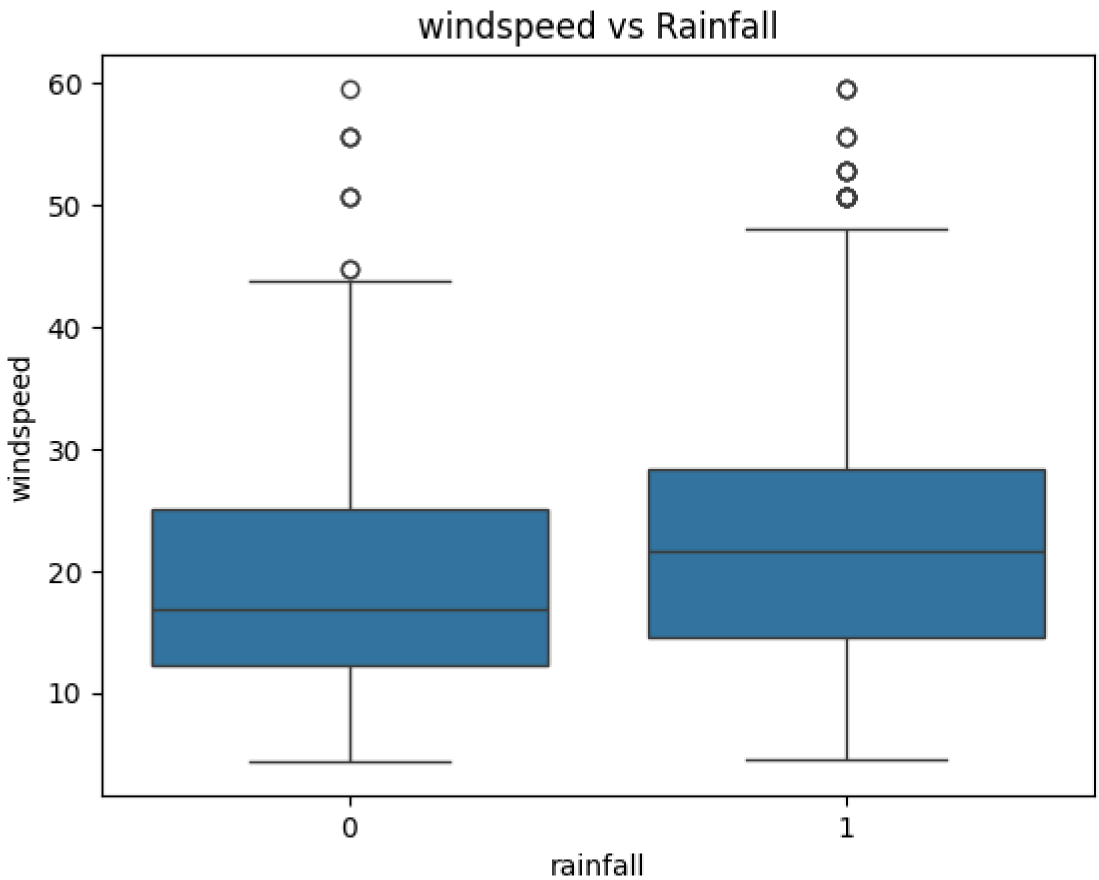

dewpoint vs Rainfall
dewpoint
rainfall
0
1
0
5
10
15
20
25


humidity vs Rainfall
humidity
rainfall
0
1
40
50
60
70
80
90
100


cloud vs Rainfall
100
80
60
40
20
0
cloud
0
1
rainfall


sunshine vs Rainfall
12
10
8
6
4
2
0
sunshine
0
1
rainfall


winddirection vs Rainfall
300
250
200
150
100
50
0
winddirection
0
1
rainfall


windspeed vs Rainfall
60
50
40
30
20
10
windspeed
0
1
rainfall

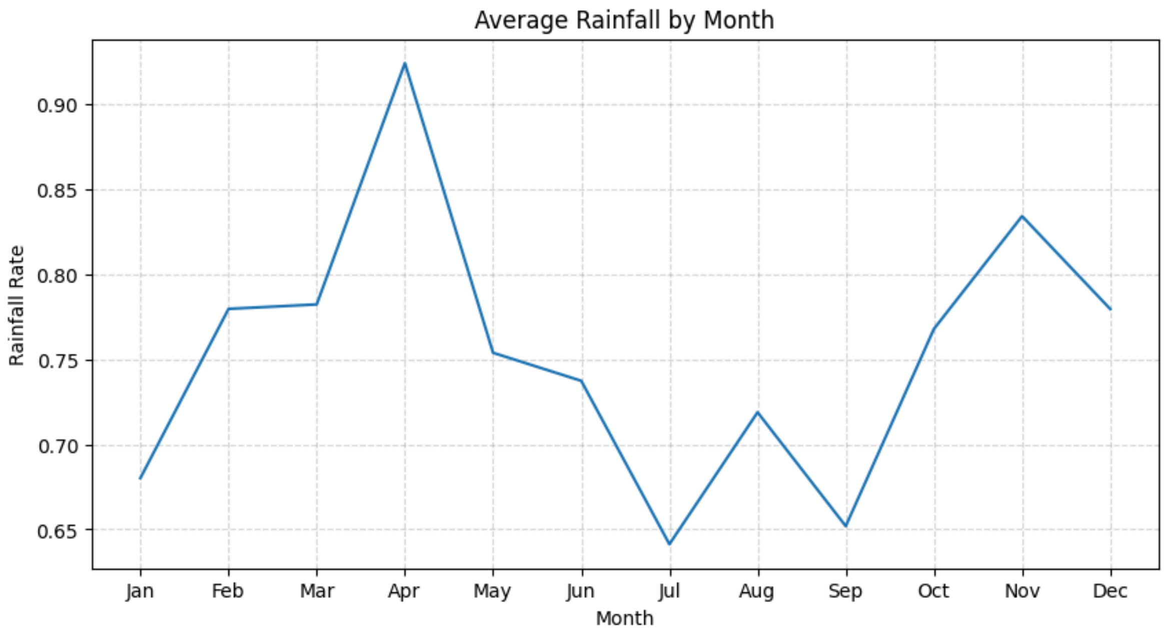
In [17]:
```
# Categorical vs Target
for col in cat_cols:
    sns.countplot(x=col, hue='rainfall', data=df_train)
    plt.title(f"{col} vs Rainfall")
    plt.show()
```

In [18]:
```
def day_to_month(day):
    if day <= 31: return 1
    elif day <= 59: return 2
    elif day <= 90: return 3
    elif day <= 120: return 4
    elif day <= 151: return 5
    elif day <= 181: return 6
    elif day <= 212: return 7
    elif day <= 243: return 8
    elif day <= 273: return 9
    elif day <= 304: return 10
    elif day <= 334: return 11
    else: return 12

df_train["month"] = df_train["day"].apply(day_to_month)
```

In [19]:
```
monthly_rainfall = df_train.groupby("month")["rainfall"].mean()
```

In [20]:
```
import calendar
monthly_rainfall.index = monthly_rainfall.index.map(lambda x: calendar.month_abb
```

In [21]:
```
plt.figure(figsize=(10,5))
plt.plot(monthly_rainfall.index, monthly_rainfall.values)
plt.grid(True, linestyle='--', alpha=0.5)
plt.xlabel("Month")
plt.ylabel("Rainfall Rate")
plt.title("Average Rainfall by Month")
plt.show()
```



Outcome: After this EDA, you'll know:

Which features are useful or need transformation

If target is imbalanced

Outliers or missing values to handle

Relationships that could inspire new features

6.Train-Test Split

In [22]:
```
# Dropping 'id' column as it has not predictive value

X = df_train.drop(columns=["id", "rainfall"])
y = df_train["rainfall"]

print(X.columns)
print(X.shape)
```

```
Index(['day', 'pressure', 'maxtemp', 'temparature', 'mintemp', 'dewpoint',
       'humidity', 'cloud', 'sunshine', 'winddirection', 'windspeed', 'month'],
      dtype='object')
(2190, 12)
```

In [23]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_st
)

# Performing a sanity check

print("Train distribution:")
print(y_train.value_counts(normalize=True))

print("\nValidation distribution:")
print(y_val.value_counts(normalize=True))
```

```
Train distribution:
rainfall
1    0.753425
0    0.246575
Name: proportion, dtype: float64

Validation distribution:
rainfall
1    0.753425
0    0.246575
Name: proportion, dtype: float64
```

In [24]:
```
for df in [X_train, X_val]:
    if "temparature" in df.columns:
        df.rename(columns={"temparature": "temperature"}, inplace=True)
```

7. Column Transformation/Preprocessing Pipline

In [25]:
```
# Step 1: Feature Engineering

class FeatureEngineer(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
```

```
        # Learn the median for winddirection to avoid leakage
        if "winddirection" in X.columns:
            self.winddirection_median_ = X["winddirection"].median()
        return self

    def transform(self, X):
        X = X.copy()

        # 1️⃣ Fill missing winddirection values with median from training set
        if "winddirection" in X.columns:
            X["winddirection"] = X["winddirection"].fillna(self.winddirection_me

        # 2️⃣ Windspeed category using quantiles (Low, Medium, High)
        if "windspeed" in X.columns:
            X["windspeed_category"] = pd.qcut(
                X["windspeed"],
                q=3,  # splits data into 3 equal-sized bins
                labels=["Low", "Medium", "High"]
            )

        # 3️⃣ Temperature range
        if {"maxtemp", "mintemp"}.issubset(X.columns):
            X["temp_range"] = X["maxtemp"] - X["mintemp"]

        # 4️⃣ Cyclical day encoding
        if "day" in X.columns:
            X["day_sin"] = np.sin(2 * np.pi * X["day"] / 365)
            X["day_cos"] = np.cos(2 * np.pi * X["day"] / 365)
            X.drop(columns=["day"], inplace=True)

        return X
```

In [26]:
```
# Step 2: Correlation Filter (numeric only)

class CorrelationFilter(BaseEstimator, TransformerMixin):
    def __init__(self, threshold=0.9, keep_columns=None):
        self.threshold = threshold
        self.keep_columns = keep_columns
        self.to_drop_ = None

    def fit(self, X, y=None):
        X = pd.DataFrame(X)
        numeric_columns = X.select_dtypes(include=[np.number]).columns
        corr_matrix = X[numeric_columns].corr().abs()
        upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astyp
        self.to_drop_ = [col for col in upper.columns if any(upper[col] > self.t
        if self.keep_columns:
            self.to_drop_ = [c for c in self.to_drop_ if c not in self.keep_colu
        return self

    def transform(self, X):
        X = pd.DataFrame(X).copy()
        return X.drop(columns=self.to_drop_, errors="ignore")
```

In [27]:
```
# Step 3: Column Transformer (joblib-safe)

numeric_features = [
    "windspeed", "temperature", "maxtemp", "mintemp",
    "humidity", "pressure", "dewpoint", "cloud", "sunshine",
```

```
    "temp_range", "day_sin", "day_cos"
]

categorical_features = [
    "windspeed_category", "winddirection"
]

# Named function for Inf → NaN (joblib-safe)
def inf_to_nan_func(X):
    return np.where(np.isinf(X), np.nan, X)

inf_to_nan = FunctionTransformer(inf_to_nan_func, feature_names_out="one-to-one"

numeric_transformer = Pipeline([
    ("inf_to_nan", inf_to_nan),
    ("imputer", SimpleImputer(strategy="median"))
])

categorical_transformer = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore", sparse_output=False))
])

preprocessor = ColumnTransformer([
    ("num", numeric_transformer, numeric_features),
    ("cat", categorical_transformer, categorical_features)
], remainder="drop")
```

8. Pipeline Construction

In [28]:
```
# Step 4: Full Pipeline (preprocessing + MI selection)

preprocessing_mi_pipeline = Pipeline([
    ("feature_engineering", FeatureEngineer()),
    ("correlation_filter", CorrelationFilter(threshold=0.9, keep_columns=numeric
    ("preprocessing", preprocessor),# joblib-safe now
    ("mi_selection", SelectKBest(score_func=mutual_info_classif, k=20))
])
```

9. Model Training, Hyperparameter Tuning & Model evalauation

In [29]:
```
# Step 5: Transform

X_train_transformed = preprocessing_mi_pipeline.fit_transform(X_train, y_train)
X_val_transformed = preprocessing_mi_pipeline.transform(X_val)
```

In [30]:
```
# Get ALL feature names from ColumnTransformer

column_transformer = preprocessing_mi_pipeline.named_steps["preprocessing"]
all_feature_names = column_transformer.get_feature_names_out()

# Apply MI selector mask

mi_selector = preprocessing_mi_pipeline.named_steps["mi_selection"]
selected_mask = mi_selector.get_support()
```

```
selected_feature_names = all_feature_names[selected_mask]

print(f"Total features after MI selection: {len(selected_feature_names)}")
```

Total features after MI selection: 20

In [31]:
```
# 1 Define Optuna objective functions for hyperparameter tuning

def rf_objective(trial):
    model = RandomForestClassifier(
        n_estimators=trial.suggest_int("n_estimators", 50, 300),
        max_depth=trial.suggest_int("max_depth", 3, 20),
        min_samples_split=trial.suggest_int("min_samples_split", 2, 20),
        min_samples_leaf=trial.suggest_int("min_samples_leaf", 1, 10),
        random_state=42,
        n_jobs=-1
    )
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    return cross_val_score(model, X_train_transformed, y_train, cv=cv, scoring="

def gb_objective(trial):
    model = GradientBoostingClassifier(
        n_estimators=trial.suggest_int("n_estimators", 50, 300),
        learning_rate=trial.suggest_float("learning_rate", 0.01, 0.3),
        max_depth=trial.suggest_int("max_depth", 2, 10),
        min_samples_split=trial.suggest_int("min_samples_split", 2, 20),
        min_samples_leaf=trial.suggest_int("min_samples_leaf", 1, 10),
        random_state=42
    )
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    return cross_val_score(model, X_train_transformed, y_train, cv=cv, scoring="

def ada_objective(trial):
    model = AdaBoostClassifier(
        n_estimators=trial.suggest_int("n_estimators", 50, 300),
        learning_rate=trial.suggest_float("learning_rate", 0.01, 2.0),
        random_state=42
    )
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    return cross_val_score(model, X_train_transformed, y_train, cv=cv, scoring="

def et_objective(trial):
    model = ExtraTreesClassifier(
        n_estimators=trial.suggest_int("n_estimators", 50, 300),
        max_depth=trial.suggest_int("max_depth", 3, 20),
        min_samples_split=trial.suggest_int("min_samples_split", 2, 20),
        min_samples_leaf=trial.suggest_int("min_samples_leaf", 1, 10),
        random_state=42,
        n_jobs=-1
    )
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    return cross_val_score(model, X_train_transformed, y_train, cv=cv, scoring="

def lr_objective(trial):
    C = trial.suggest_loguniform("C", 1e-3, 1e3)
    penalty = trial.suggest_categorical("penalty", ["l1", "l2"])
    solver = "saga"  # ✅ saga supports both l1 and l2
    model = LogisticRegression(
        C=C, penalty=penalty, solver=solver, max_iter=2000, random_state=42
```

```
    )
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    return cross_val_score(model, X_train_transformed, y_train, cv=cv, scoring="

# 2. Tune models with Optuna

def tune_model(objective_fn, n_trials=30):
    study = optuna.create_study(direction="maximize")
    study.optimize(objective_fn, n_trials=n_trials)
    return study.best_trial.params

best_params = {}
best_params["RandomForest"] = tune_model(rf_objective)
best_params["GradientBoosting"] = tune_model(gb_objective)
best_params["AdaBoost"] = tune_model(ada_objective)
best_params["ExtraTrees"] = tune_model(et_objective)
best_params["LogisticRegression"] = tune_model(lr_objective)

print("\nBest hyperparameters for each model:")
for name, params in best_params.items():
    print(f"{name}: {params}")


# 3. Train final tuned models and evaluate

models = {
    "RandomForest": RandomForestClassifier(**best_params["RandomForest"], random
    "GradientBoosting": GradientBoostingClassifier(**best_params["GradientBoosti
    "AdaBoost": AdaBoostClassifier(**best_params["AdaBoost"], random_state=42),
    "ExtraTrees": ExtraTreesClassifier(**best_params["ExtraTrees"], random_state
    "LogisticRegression": LogisticRegression(**best_params["LogisticRegression"]
}

for name, model in models.items():
    print(f"\n==== {name} ====")
    model.fit(X_train_transformed, y_train)
    y_pred = model.predict(X_val_transformed)
    y_pred_proba = model.predict_proba(X_val_transformed)[:,1] if hasattr(model,

    accuracy = accuracy_score(y_val, y_pred)
    roc_auc = roc_auc_score(y_val, y_pred_proba) if y_pred_proba is not None els
    report = classification_report(y_val, y_pred)

    print(f"Accuracy: {accuracy:.4f}")
    if roc_auc is not None:
        print(f"ROC AUC: {roc_auc:.4f}")
    print("Classification Report:\n", report)

    # Feature importance
    if hasattr(model, "feature_importances_"):
        fi = pd.DataFrame({
            "Feature": selected_feature_names,
            "Importance": model.feature_importances_
        }).sort_values(by="Importance", ascending=False)
        plt.figure(figsize=(12,6))
        sns.barplot(x="Importance", y="Feature", data=fi, palette="viridis")
        plt.title(f"{name} Feature Importance")
        plt.show()
    elif hasattr(model, "coef_"):
        coefs = abs(model.coef_).flatten()
```

```
        fi = pd.DataFrame({
            "Feature": selected_feature_names,
            "Importance": coefs
        }).sort_values(by="Importance", ascending=False)
        plt.figure(figsize=(12,6))
        sns.barplot(x="Importance", y="Feature", data=fi, palette="coolwarm")
        plt.title(f"{name} Coefficient Magnitude (Feature Importance)")
        plt.show()
```

```
[I 2026-02-14 09:26:59,748] A new study created in memory with name: no-name-295e
598a-a4ac-4a4f-b991-300e4bc96325
[I 2026-02-14 09:27:01,379] Trial 0 finished with value: 0.8879735607883162 and p
arameters: {'n_estimators': 139, 'max_depth': 4, 'min_samples_split': 6, 'min_sam
ples_leaf': 3}. Best is trial 0 with value: 0.8879735607883162.
[I 2026-02-14 09:27:02,621] Trial 1 finished with value: 0.8881314143844217 and p
arameters: {'n_estimators': 91, 'max_depth': 8, 'min_samples_split': 14, 'min_sam
ples_leaf': 5}. Best is trial 1 with value: 0.8881314143844217.
[I 2026-02-14 09:27:03,746] Trial 2 finished with value: 0.8872800782484023 and p
arameters: {'n_estimators': 94, 'max_depth': 4, 'min_samples_split': 10, 'min_sam
ples_leaf': 10}. Best is trial 1 with value: 0.8881314143844217.
[I 2026-02-14 09:27:05,193] Trial 3 finished with value: 0.8897592606093008 and p
arameters: {'n_estimators': 110, 'max_depth': 10, 'min_samples_split': 11, 'min_s
amples_leaf': 8}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:07,266] Trial 4 finished with value: 0.8870523600074522 and p
arameters: {'n_estimators': 153, 'max_depth': 4, 'min_samples_split': 3, 'min_sam
ples_leaf': 4}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:10,503] Trial 5 finished with value: 0.8884268709549383 and p
arameters: {'n_estimators': 237, 'max_depth': 4, 'min_samples_split': 14, 'min_sa
mples_leaf': 4}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:12,895] Trial 6 finished with value: 0.882870302868298 and pa
rameters: {'n_estimators': 214, 'max_depth': 3, 'min_samples_split': 5, 'min_samp
les_leaf': 1}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:15,839] Trial 7 finished with value: 0.8878660299871205 and p
arameters: {'n_estimators': 234, 'max_depth': 4, 'min_samples_split': 4, 'min_sam
ples_leaf': 10}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:19,527] Trial 8 finished with value: 0.8897064064866791 and p
arameters: {'n_estimators': 289, 'max_depth': 19, 'min_samples_split': 14, 'min_s
amples_leaf': 10}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:21,215] Trial 9 finished with value: 0.8795258114424114 and p
arameters: {'n_estimators': 81, 'max_depth': 18, 'min_samples_split': 20, 'min_sa
mples_leaf': 1}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:22,255] Trial 10 finished with value: 0.8861324755169984 and
parameters: {'n_estimators': 52, 'max_depth': 13, 'min_samples_split': 9, 'min_sa
mples_leaf': 7}. Best is trial 3 with value: 0.8897592606093008.
[I 2026-02-14 09:27:25,924] Trial 11 finished with value: 0.8899612605606991 and
parameters: {'n_estimators': 272, 'max_depth': 20, 'min_samples_split': 15, 'min_
samples_leaf': 8}. Best is trial 11 with value: 0.8899612605606991.
[I 2026-02-14 09:27:29,731] Trial 12 finished with value: 0.8891986221477 and par
ameters: {'n_estimators': 298, 'max_depth': 13, 'min_samples_split': 19, 'min_sam
ples_leaf': 7}. Best is trial 11 with value: 0.8899612605606991.
[I 2026-02-14 09:27:32,190] Trial 13 finished with value: 0.8899017237329186 and
parameters: {'n_estimators': 187, 'max_depth': 10, 'min_samples_split': 17, 'min_
samples_leaf': 7}. Best is trial 11 with value: 0.8899612605606991.
[I 2026-02-14 09:27:35,597] Trial 14 finished with value: 0.8893358201096774 and
parameters: {'n_estimators': 205, 'max_depth': 16, 'min_samples_split': 17, 'min_
samples_leaf': 8}. Best is trial 11 with value: 0.8899612605606991.
[I 2026-02-14 09:27:39,129] Trial 15 finished with value: 0.8901254931026383 and
parameters: {'n_estimators': 263, 'max_depth': 8, 'min_samples_split': 17, 'min_s
amples_leaf': 6}. Best is trial 15 with value: 0.8901254931026383.
[I 2026-02-14 09:27:42,355] Trial 16 finished with value: 0.890617684462913 and p
arameters: {'n_estimators': 261, 'max_depth': 7, 'min_samples_split': 17, 'min_sa
mples_leaf': 6}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:27:45,611] Trial 17 finished with value: 0.8906175832098047 and
parameters: {'n_estimators': 262, 'max_depth': 7, 'min_samples_split': 17, 'min_s
amples_leaf': 6}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:27:49,949] Trial 18 finished with value: 0.8898538310126121 and
parameters: {'n_estimators': 260, 'max_depth': 8, 'min_samples_split': 12, 'min_s
amples_leaf': 5}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:27:51,937] Trial 19 finished with value: 0.8902533757786364 and
```

```
parameters: {'n_estimators': 158, 'max_depth': 6, 'min_samples_split': 8, 'min_sa
mples_leaf': 6}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:27:54,808] Trial 20 finished with value: 0.8902929657440485 and
parameters: {'n_estimators': 237, 'max_depth': 6, 'min_samples_split': 19, 'min_s
amples_leaf': 3}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:27:57,718] Trial 21 finished with value: 0.8902929657440485 and
parameters: {'n_estimators': 237, 'max_depth': 6, 'min_samples_split': 19, 'min_s
amples_leaf': 3}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:28:01,105] Trial 22 finished with value: 0.8901533377074677 and
parameters: {'n_estimators': 249, 'max_depth': 6, 'min_samples_split': 18, 'min_s
amples_leaf': 2}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:28:04,547] Trial 23 finished with value: 0.888910253294776 and p
arameters: {'n_estimators': 216, 'max_depth': 10, 'min_samples_split': 16, 'min_s
amples_leaf': 5}. Best is trial 16 with value: 0.890617684462913.
[I 2026-02-14 09:28:07,981] Trial 24 finished with value: 0.8911411630337052 and
parameters: {'n_estimators': 281, 'max_depth': 7, 'min_samples_split': 20, 'min_s
amples_leaf': 4}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:11,590] Trial 25 finished with value: 0.8874239589155387 and
parameters: {'n_estimators': 283, 'max_depth': 12, 'min_samples_split': 20, 'min_
samples_leaf': 4}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:16,221] Trial 26 finished with value: 0.8903644504386283 and
parameters: {'n_estimators': 298, 'max_depth': 8, 'min_samples_split': 12, 'min_s
amples_leaf': 6}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:18,864] Trial 27 finished with value: 0.8896101147805238 and
parameters: {'n_estimators': 190, 'max_depth': 15, 'min_samples_split': 16, 'min_
samples_leaf': 9}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:22,363] Trial 28 finished with value: 0.8909177986764194 and
parameters: {'n_estimators': 274, 'max_depth': 9, 'min_samples_split': 18, 'min_s
amples_leaf': 5}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:26,548] Trial 29 finished with value: 0.8860826589876309 and
parameters: {'n_estimators': 275, 'max_depth': 11, 'min_samples_split': 7, 'min_s
amples_leaf': 4}. Best is trial 24 with value: 0.8911411630337052.
[I 2026-02-14 09:28:26,550] A new study created in memory with name: no-name-71be
754c-76a0-478b-9600-f5e94ad240d3
[I 2026-02-14 09:28:34,370] Trial 0 finished with value: 0.8715665070917679 and p
arameters: {'n_estimators': 144, 'learning_rate': 0.05312478767797437, 'max_dept
h': 8, 'min_samples_split': 12, 'min_samples_leaf': 5}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:28:38,327] Trial 1 finished with value: 0.8635386543866896 and p
arameters: {'n_estimators': 149, 'learning_rate': 0.26967300360907565, 'max_dept
h': 5, 'min_samples_split': 19, 'min_samples_leaf': 6}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:28:45,759] Trial 2 finished with value: 0.8668911448081456 and p
arameters: {'n_estimators': 300, 'learning_rate': 0.15251212966003974, 'max_dept
h': 4, 'min_samples_split': 19, 'min_samples_leaf': 5}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:28:47,954] Trial 3 finished with value: 0.8668825382939257 and p
arameters: {'n_estimators': 67, 'learning_rate': 0.23152748860001304, 'max_dept
h': 6, 'min_samples_split': 15, 'min_samples_leaf': 7}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:28:52,907] Trial 4 finished with value: 0.8613359942650239 and p
arameters: {'n_estimators': 293, 'learning_rate': 0.2616221377779615, 'max_dept
h': 3, 'min_samples_split': 4, 'min_samples_leaf': 6}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:28:58,250] Trial 5 finished with value: 0.8675047386454764 and p
arameters: {'n_estimators': 96, 'learning_rate': 0.23705237102265775, 'max_dept
h': 9, 'min_samples_split': 8, 'min_samples_leaf': 10}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:29:06,366] Trial 6 finished with value: 0.8683727815443933 and p
arameters: {'n_estimators': 256, 'learning_rate': 0.28596956432611104, 'max_dept
```

```
h': 6, 'min_samples_split': 20, 'min_samples_leaf': 2}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:29:14,107] Trial 7 finished with value: 0.8649851562942981 and p
arameters: {'n_estimators': 248, 'learning_rate': 0.21197884486681876, 'max_dept
h': 5, 'min_samples_split': 9, 'min_samples_leaf': 5}. Best is trial 0 with valu
e: 0.8715665070917679.
[I 2026-02-14 09:29:16,275] Trial 8 finished with value: 0.8904951682016637 and p
arameters: {'n_estimators': 81, 'learning_rate': 0.08794826508985659, 'max_dept
h': 2, 'min_samples_split': 8, 'min_samples_leaf': 4}. Best is trial 8 with valu
e: 0.8904951682016637.
[I 2026-02-14 09:29:21,269] Trial 9 finished with value: 0.8692247251990637 and p
arameters: {'n_estimators': 121, 'learning_rate': 0.14937181663588361, 'max_dept
h': 7, 'min_samples_split': 17, 'min_samples_leaf': 1}. Best is trial 8 with valu
e: 0.8904951682016637.
[I 2026-02-14 09:29:24,452] Trial 10 finished with value: 0.8907136217831887 and
parameters: {'n_estimators': 192, 'learning_rate': 0.025413994003582185, 'max_dep
th': 2, 'min_samples_split': 4, 'min_samples_leaf': 2}. Best is trial 10 with val
ue: 0.8907136217831887.
[I 2026-02-14 09:29:26,911] Trial 11 finished with value: 0.8908965355236406 and
parameters: {'n_estimators': 197, 'learning_rate': 0.015321479841586317, 'max_dep
th': 2, 'min_samples_split': 2, 'min_samples_leaf': 2}. Best is trial 11 with val
ue: 0.8908965355236406.
[I 2026-02-14 09:29:29,453] Trial 12 finished with value: 0.8909991555490754 and
parameters: {'n_estimators': 202, 'learning_rate': 0.014340685898447041, 'max_dep
th': 2, 'min_samples_split': 2, 'min_samples_leaf': 3}. Best is trial 12 with val
ue: 0.8909991555490754.
[I 2026-02-14 09:29:32,808] Trial 13 finished with value: 0.8887391861680154 and
parameters: {'n_estimators': 190, 'learning_rate': 0.012506087306849276, 'max_dep
th': 3, 'min_samples_split': 2, 'min_samples_leaf': 3}. Best is trial 12 with val
ue: 0.8909991555490754.
[I 2026-02-14 09:29:54,759] Trial 14 finished with value: 0.8666684892226192 and
parameters: {'n_estimators': 221, 'learning_rate': 0.09485500506482177, 'max_dept
h': 10, 'min_samples_split': 2, 'min_samples_leaf': 1}. Best is trial 12 with val
ue: 0.8909991555490754.
[I 2026-02-14 09:29:58,482] Trial 15 finished with value: 0.8802023847132107 and
parameters: {'n_estimators': 218, 'learning_rate': 0.0724242077121705, 'max_dept
h': 3, 'min_samples_split': 5, 'min_samples_leaf': 3}. Best is trial 12 with valu
e: 0.8909991555490754.
[I 2026-02-14 09:30:00,522] Trial 16 finished with value: 0.8874536260763207 and
parameters: {'n_estimators': 166, 'learning_rate': 0.1232268652795309, 'max_dept
h': 2, 'min_samples_split': 6, 'min_samples_leaf': 8}. Best is trial 12 with valu
e: 0.8909991555490754.
[I 2026-02-14 09:30:06,663] Trial 17 finished with value: 0.8806144848646854 and
parameters: {'n_estimators': 233, 'learning_rate': 0.0425827836677503, 'max_dept
h': 4, 'min_samples_split': 11, 'min_samples_leaf': 3}. Best is trial 12 with val
ue: 0.8909991555490754.
[I 2026-02-14 09:30:10,885] Trial 18 finished with value: 0.8699286368091501 and
parameters: {'n_estimators': 188, 'learning_rate': 0.19618061503036865, 'max_dept
h': 4, 'min_samples_split': 2, 'min_samples_leaf': 2}. Best is trial 12 with valu
e: 0.8909991555490754.
[I 2026-02-14 09:30:14,293] Trial 19 finished with value: 0.8914468967947317 and
parameters: {'n_estimators': 273, 'learning_rate': 0.011782613983568804, 'max_dep
th': 2, 'min_samples_split': 13, 'min_samples_leaf': 4}. Best is trial 19 with va
lue: 0.8914468967947317.
[I 2026-02-14 09:30:19,908] Trial 20 finished with value: 0.8810142321369266 and
parameters: {'n_estimators': 274, 'learning_rate': 0.05895875067175775, 'max_dept
h': 3, 'min_samples_split': 13, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:22,483] Trial 21 finished with value: 0.8913836136019377 and
parameters: {'n_estimators': 207, 'learning_rate': 0.02177478458702009, 'max_dept
```

```
h': 2, 'min_samples_split': 15, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:25,519] Trial 22 finished with value: 0.8911259244408803 and
parameters: {'n_estimators': 245, 'learning_rate': 0.03553978054743935, 'max_dept
h': 2, 'min_samples_split': 15, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:30,657] Trial 23 finished with value: 0.8738510809781861 and
parameters: {'n_estimators': 266, 'learning_rate': 0.10969219028240622, 'max_dept
h': 3, 'min_samples_split': 15, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:37,582] Trial 24 finished with value: 0.8784354167172932 and
parameters: {'n_estimators': 239, 'learning_rate': 0.03979217607972785, 'max_dept
h': 5, 'min_samples_split': 14, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:44,649] Trial 25 finished with value: 0.8735815452034377 and
parameters: {'n_estimators': 284, 'learning_rate': 0.06756170082157004, 'max_dept
h': 4, 'min_samples_split': 17, 'min_samples_leaf': 7}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:48,117] Trial 26 finished with value: 0.8902659817906411 and
parameters: {'n_estimators': 264, 'learning_rate': 0.03594473448759752, 'max_dept
h': 2, 'min_samples_split': 17, 'min_samples_leaf': 6}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:30:51,759] Trial 27 finished with value: 0.8715930354061869 and
parameters: {'n_estimators': 216, 'learning_rate': 0.1291803576633805, 'max_dept
h': 3, 'min_samples_split': 15, 'min_samples_leaf': 5}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:31:01,920] Trial 28 finished with value: 0.873231816966781 and p
arameters: {'n_estimators': 230, 'learning_rate': 0.09669604397864989, 'max_dept
h': 7, 'min_samples_split': 10, 'min_samples_leaf': 4}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:31:09,590] Trial 29 finished with value: 0.8726874802556438 and
parameters: {'n_estimators': 162, 'learning_rate': 0.05256322592116986, 'max_dept
h': 8, 'min_samples_split': 12, 'min_samples_leaf': 5}. Best is trial 19 with val
ue: 0.8914468967947317.
[I 2026-02-14 09:31:09,593] A new study created in memory with name: no-name-475b
cf33-2b93-49df-a4bc-de007a38f61f
[I 2026-02-14 09:31:11,326] Trial 0 finished with value: 0.8898573748714085 and p
arameters: {'n_estimators': 93, 'learning_rate': 0.08984868829153067}. Best is tr
ial 0 with value: 0.8898573748714085.
[I 2026-02-14 09:31:15,326] Trial 1 finished with value: 0.8764529314799964 and p
arameters: {'n_estimators': 283, 'learning_rate': 1.573089806165302}. Best is tri
al 0 with value: 0.8898573748714085.
[I 2026-02-14 09:31:18,703] Trial 2 finished with value: 0.8880942544936129 and p
arameters: {'n_estimators': 258, 'learning_rate': 0.4498666503745843}. Best is tr
ial 0 with value: 0.8898573748714085.
[I 2026-02-14 09:31:21,878] Trial 3 finished with value: 0.8758105817598599 and p
arameters: {'n_estimators': 246, 'learning_rate': 1.6037865913786884}. Best is tr
ial 0 with value: 0.8898573748714085.
[I 2026-02-14 09:31:24,943] Trial 4 finished with value: 0.8589140705369654 and p
arameters: {'n_estimators': 172, 'learning_rate': 1.9070672641097264}. Best is tr
ial 0 with value: 0.8898573748714085.
[I 2026-02-14 09:31:27,349] Trial 5 finished with value: 0.8910924096619766 and p
arameters: {'n_estimators': 161, 'learning_rate': 0.13811002440205775}. Best is t
rial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:28,937] Trial 6 finished with value: 0.8809393048366585 and p
arameters: {'n_estimators': 123, 'learning_rate': 1.3694310436648687}. Best is tr
ial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:29,713] Trial 7 finished with value: 0.8814752375397925 and p
arameters: {'n_estimators': 58, 'learning_rate': 1.7615904145646333}. Best is tri
al 5 with value: 0.8910924096619766.
```

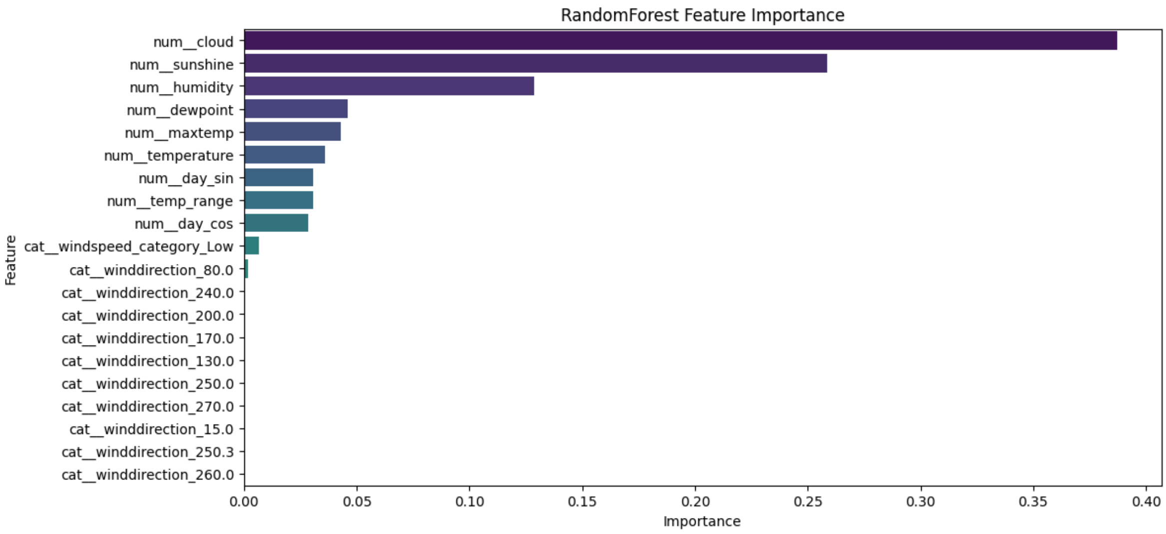
```
[I 2026-02-14 09:31:31,912] Trial 8 finished with value: 0.8865093902132795 and p
arameters: {'n_estimators': 169, 'learning_rate': 0.551544886053529}. Best is tri
al 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:34,111] Trial 9 finished with value: 0.8583985403351884 and p
arameters: {'n_estimators': 168, 'learning_rate': 1.9818214775841065}. Best is tr
ial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:37,375] Trial 10 finished with value: 0.8863179205851619 and
parameters: {'n_estimators': 215, 'learning_rate': 1.0081703346947195}. Best is t
rial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:39,202] Trial 11 finished with value: 0.8837680635545512 and
parameters: {'n_estimators': 96, 'learning_rate': 0.022554986559257695}. Best is
trial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:40,873] Trial 12 finished with value: 0.8895430852227163 and
parameters: {'n_estimators': 125, 'learning_rate': 0.05177329978943576}. Best is
trial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:41,558] Trial 13 finished with value: 0.8886813200165244 and
parameters: {'n_estimators': 51, 'learning_rate': 0.3809071875808233}. Best is tr
ial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:42,912] Trial 14 finished with value: 0.8818303828177525 and
parameters: {'n_estimators': 105, 'learning_rate': 0.7576107888574175}. Best is t
rial 5 with value: 0.8910924096619766.
[I 2026-02-14 09:31:44,716] Trial 15 finished with value: 0.8912070281807651 and
parameters: {'n_estimators': 138, 'learning_rate': 0.18265124489785745}. Best is
trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:47,310] Trial 16 finished with value: 0.8859683948547221 and
parameters: {'n_estimators': 197, 'learning_rate': 0.98061711149807}. Best is tri
al 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:49,175] Trial 17 finished with value: 0.8908089515848137 and
parameters: {'n_estimators': 143, 'learning_rate': 0.29073120053174706}. Best is
trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:52,951] Trial 18 finished with value: 0.8880462605201981 and
parameters: {'n_estimators': 210, 'learning_rate': 0.6987899833178266}. Best is t
rial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:54,818] Trial 19 finished with value: 0.891005129482475 and p
arameters: {'n_estimators': 144, 'learning_rate': 0.22436922812618937}. Best is t
rial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:55,871] Trial 20 finished with value: 0.8838102861007833 and
parameters: {'n_estimators': 80, 'learning_rate': 1.042977635551596}. Best is tri
al 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:57,758] Trial 21 finished with value: 0.8904864604343354 and
parameters: {'n_estimators': 144, 'learning_rate': 0.24065961458597362}. Best is
trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:31:59,759] Trial 22 finished with value: 0.8909838157031421 and
parameters: {'n_estimators': 145, 'learning_rate': 0.21278484260653185}. Best is
trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:02,223] Trial 23 finished with value: 0.8876843819105247 and
parameters: {'n_estimators': 190, 'learning_rate': 0.5481607493792615}. Best is t
rial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:04,296] Trial 24 finished with value: 0.8828225114011001 and
parameters: {'n_estimators': 123, 'learning_rate': 0.7488866898678586}. Best is t
rial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:06,926] Trial 25 finished with value: 0.8906851696597086 and
parameters: {'n_estimators': 156, 'learning_rate': 0.1761562776891065}. Best is t
rial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:09,811] Trial 26 finished with value: 0.8906687666561363 and
parameters: {'n_estimators': 222, 'learning_rate': 0.36762996118706637}. Best is
trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:12,252] Trial 27 finished with value: 0.8874993924813491 and
parameters: {'n_estimators': 189, 'learning_rate': 0.5483219882197348}. Best is t
rial 15 with value: 0.8912070281807651.
```

```
[I 2026-02-14 09:32:13,771] Trial 28 finished with value: 0.8905594132989882 and parameters: {'n_estimators': 117, 'learning_rate': 0.14151962884447256}. Best is trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:14,911] Trial 29 finished with value: 0.8637972548257231 and parameters: {'n_estimators': 87, 'learning_rate': 0.012485967379583962}. Best is trial 15 with value: 0.8912070281807651.
[I 2026-02-14 09:32:14,912] A new study created in memory with name: no-name-5eae3cbd-b404-4689-9b33-576b04555a06
[I 2026-02-14 09:32:15,658] Trial 0 finished with value: 0.8930459365102509 and parameters: {'n_estimators': 82, 'max_depth': 10, 'min_samples_split': 7, 'min_samples_leaf': 7}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:18,515] Trial 1 finished with value: 0.8910137866232493 and parameters: {'n_estimators': 285, 'max_depth': 17, 'min_samples_split': 12, 'min_samples_leaf': 10}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:20,608] Trial 2 finished with value: 0.8921106615473094 and parameters: {'n_estimators': 219, 'max_depth': 6, 'min_samples_split': 11, 'min_samples_leaf': 4}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:21,322] Trial 3 finished with value: 0.8916907649064827 and parameters: {'n_estimators': 80, 'max_depth': 17, 'min_samples_split': 18, 'min_samples_leaf': 8}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:22,232] Trial 4 finished with value: 0.8926309000186305 and parameters: {'n_estimators': 97, 'max_depth': 13, 'min_samples_split': 16, 'min_samples_leaf': 6}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:24,471] Trial 5 finished with value: 0.8912430236608264 and parameters: {'n_estimators': 273, 'max_depth': 10, 'min_samples_split': 10, 'min_samples_leaf': 10}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:25,087] Trial 6 finished with value: 0.887565409508072 and parameters: {'n_estimators': 58, 'max_depth': 16, 'min_samples_split': 19, 'min_samples_leaf': 1}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:25,603] Trial 7 finished with value: 0.8917124330716953 and parameters: {'n_estimators': 52, 'max_depth': 13, 'min_samples_split': 6, 'min_samples_leaf': 9}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:26,326] Trial 8 finished with value: 0.8905940924886394 and parameters: {'n_estimators': 75, 'max_depth': 13, 'min_samples_split': 6, 'min_samples_leaf': 3}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:26,912] Trial 9 finished with value: 0.8852563323694037 and parameters: {'n_estimators': 66, 'max_depth': 3, 'min_samples_split': 9, 'min_samples_leaf': 7}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:28,082] Trial 10 finished with value: 0.8922186986140475 and parameters: {'n_estimators': 137, 'max_depth': 8, 'min_samples_split': 2, 'min_samples_leaf': 5}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:29,206] Trial 11 finished with value: 0.8921519728155655 and parameters: {'n_estimators': 126, 'max_depth': 20, 'min_samples_split': 15, 'min_samples_leaf': 6}. Best is trial 0 with value: 0.8930459365102509.
[I 2026-02-14 09:32:30,497] Trial 12 finished with value: 0.8931797931196488 and parameters: {'n_estimators': 130, 'max_depth': 10, 'min_samples_split': 16, 'min_samples_leaf': 7}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:32,438] Trial 13 finished with value: 0.8919987768624497 and parameters: {'n_estimators': 174, 'max_depth': 9, 'min_samples_split': 14, 'min_samples_leaf': 8}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:33,679] Trial 14 finished with value: 0.8907570087401684 and parameters: {'n_estimators': 125, 'max_depth': 6, 'min_samples_split': 7, 'min_samples_leaf': 7}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:35,155] Trial 15 finished with value: 0.8929799194835282 and parameters: {'n_estimators': 169, 'max_depth': 10, 'min_samples_split': 2, 'min_samples_leaf': 4}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:36,897] Trial 16 finished with value: 0.8916504661693114 and parameters: {'n_estimators': 211, 'max_depth': 7, 'min_samples_split': 13, 'min_samples_leaf': 8}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:37,774] Trial 17 finished with value: 0.8866522077227772 and
```

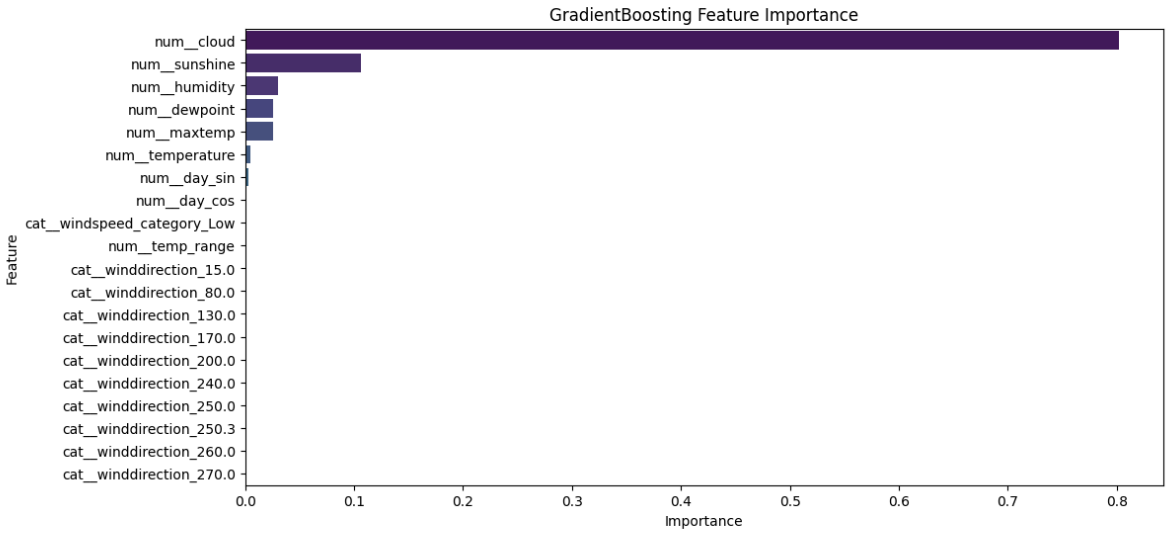
```
parameters: {'n_estimators': 102, 'max_depth': 4, 'min_samples_split': 17, 'min_s
amples_leaf': 7}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:39,089] Trial 18 finished with value: 0.8908772974330311 and
parameters: {'n_estimators': 149, 'max_depth': 11, 'min_samples_split': 20, 'min_
samples_leaf': 5}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:41,143] Trial 19 finished with value: 0.8872149724996558 and
parameters: {'n_estimators': 216, 'max_depth': 15, 'min_samples_split': 8, 'min_s
amples_leaf': 1}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:41,987] Trial 20 finished with value: 0.8883762444007031 and
parameters: {'n_estimators': 100, 'max_depth': 5, 'min_samples_split': 4, 'min_sa
mples_leaf': 9}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:43,577] Trial 21 finished with value: 0.8927349882141382 and
parameters: {'n_estimators': 170, 'max_depth': 10, 'min_samples_split': 2, 'min_s
amples_leaf': 4}. Best is trial 12 with value: 0.8931797931196488.
[I 2026-02-14 09:32:45,858] Trial 22 finished with value: 0.8931918422395567 and
parameters: {'n_estimators': 193, 'max_depth': 11, 'min_samples_split': 4, 'min_s
amples_leaf': 3}. Best is trial 22 with value: 0.8931918422395567.
[I 2026-02-14 09:32:47,895] Trial 23 finished with value: 0.8917955618737496 and
parameters: {'n_estimators': 198, 'max_depth': 12, 'min_samples_split': 4, 'min_s
amples_leaf': 3}. Best is trial 22 with value: 0.8931918422395567.
[I 2026-02-14 09:32:50,060] Trial 24 finished with value: 0.8924960308781479 and
parameters: {'n_estimators': 252, 'max_depth': 8, 'min_samples_split': 4, 'min_sa
mples_leaf': 2}. Best is trial 22 with value: 0.8931918422395567.
[I 2026-02-14 09:32:51,378] Trial 25 finished with value: 0.8924331526977879 and
parameters: {'n_estimators': 150, 'max_depth': 14, 'min_samples_split': 6, 'min_s
amples_leaf': 6}. Best is trial 22 with value: 0.8931918422395567.
[I 2026-02-14 09:32:53,103] Trial 26 finished with value: 0.8919423788810317 and
parameters: {'n_estimators': 194, 'max_depth': 12, 'min_samples_split': 8, 'min_s
amples_leaf': 5}. Best is trial 22 with value: 0.8931918422395567.
[I 2026-02-14 09:32:55,141] Trial 27 finished with value: 0.8932708196641638 and
parameters: {'n_estimators': 238, 'max_depth': 11, 'min_samples_split': 10, 'min_
samples_leaf': 7}. Best is trial 27 with value: 0.8932708196641638.
[I 2026-02-14 09:32:57,370] Trial 28 finished with value: 0.8919425813872486 and
parameters: {'n_estimators': 245, 'max_depth': 8, 'min_samples_split': 12, 'min_s
amples_leaf': 3}. Best is trial 27 with value: 0.8932708196641638.
[I 2026-02-14 09:33:00,037] Trial 29 finished with value: 0.8916114837225504 and
parameters: {'n_estimators': 248, 'max_depth': 11, 'min_samples_split': 15, 'min_
samples_leaf': 9}. Best is trial 27 with value: 0.8932708196641638.
[I 2026-02-14 09:33:00,039] A new study created in memory with name: no-name-2889
a58f-4fdf-418b-9d88-d33c7a283fb7
[I 2026-02-14 09:33:04,928] Trial 0 finished with value: 0.8951055259896478 and p
arameters: {'C': 0.634900553321891, 'penalty': 'l2'}. Best is trial 0 with value:
0.8951055259896478.
[I 2026-02-14 09:33:05,870] Trial 1 finished with value: 0.8860337537362397 and p
arameters: {'C': 0.045768741532036215, 'penalty': 'l1'}. Best is trial 0 with val
ue: 0.8951055259896478.
[I 2026-02-14 09:33:07,752] Trial 2 finished with value: 0.8920399868775972 and p
arameters: {'C': 0.03303057646666202, 'penalty': 'l2'}. Best is trial 0 with valu
e: 0.8951055259896478.
[I 2026-02-14 09:33:15,202] Trial 3 finished with value: 0.8949560764015455 and p
arameters: {'C': 436.1034259701939, 'penalty': 'l1'}. Best is trial 0 with value:
0.8951055259896478.
[I 2026-02-14 09:33:20,146] Trial 4 finished with value: 0.8949914137364017 and p
arameters: {'C': 0.71488295752957, 'penalty': 'l2'}. Best is trial 0 with value:
0.8951055259896478.
[I 2026-02-14 09:33:20,526] Trial 5 finished with value: 0.886289519088236 and pa
rameters: {'C': 0.003858872298230565, 'penalty': 'l2'}. Best is trial 0 with valu
e: 0.8951055259896478.
[I 2026-02-14 09:33:20,930] Trial 6 finished with value: 0.8850811644917499 and p
arameters: {'C': 0.028335668997371267, 'penalty': 'l1'}. Best is trial 0 with val
```

```
ue: 0.8951055259896478.
[I 2026-02-14 09:33:28,279] Trial 7 finished with value: 0.8951577725936186 and p
arameters: {'C': 5.7805221265016895, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:33:28,733] Trial 8 finished with value: 0.8866945315221176 and p
arameters: {'C': 0.004661070310171968, 'penalty': 'l2'}. Best is trial 7 with val
ue: 0.8951577725936186.
[I 2026-02-14 09:33:34,426] Trial 9 finished with value: 0.8950876041894487 and p
arameters: {'C': 3.1472738561516707, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:33:41,934] Trial 10 finished with value: 0.8949473686342172 and
parameters: {'C': 57.76301047698917, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:33:46,803] Trial 11 finished with value: 0.8950085255117333 and
parameters: {'C': 3.119761308196007, 'penalty': 'l2'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:33:52,691] Trial 12 finished with value: 0.8949472673811085 and
parameters: {'C': 29.146538656976315, 'penalty': 'l2'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:33:58,225] Trial 13 finished with value: 0.8937358751913684 and
parameters: {'C': 0.3626380316661745, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:34:04,414] Trial 14 finished with value: 0.8950523681077008 and
parameters: {'C': 11.622855049139504, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:34:10,087] Trial 15 finished with value: 0.8947920463658233 and
parameters: {'C': 0.26928769872502517, 'penalty': 'l2'}. Best is trial 7 with val
ue: 0.8951577725936186.
[I 2026-02-14 09:34:14,930] Trial 16 finished with value: 0.8949560764015455 and
parameters: {'C': 394.2151168765312, 'penalty': 'l2'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:34:22,297] Trial 17 finished with value: 0.8951137274914341 and
parameters: {'C': 7.2536571005000425, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:34:28,324] Trial 18 finished with value: 0.8949473686342172 and
parameters: {'C': 58.27036634429128, 'penalty': 'l1'}. Best is trial 7 with valu
e: 0.8951577725936186.
[I 2026-02-14 09:34:35,698] Trial 19 finished with value: 0.8951667841202726 and
parameters: {'C': 4.308661382633495, 'penalty': 'l1'}. Best is trial 19 with valu
e: 0.8951667841202726.
[I 2026-02-14 09:34:38,569] Trial 20 finished with value: 0.8906109005046454 and
parameters: {'C': 0.1435945661074243, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:34:44,823] Trial 21 finished with value: 0.8951229415243048 and
parameters: {'C': 3.6950872878853023, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:34:51,778] Trial 22 finished with value: 0.8951052222303225 and
parameters: {'C': 3.2742108980554674, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:34:57,887] Trial 23 finished with value: 0.8950084242586248 and
parameters: {'C': 21.141022508016658, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:35:05,329] Trial 24 finished with value: 0.8949560764015455 and
parameters: {'C': 136.91256441087077, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:35:11,031] Trial 25 finished with value: 0.8950623921654394 and
parameters: {'C': 1.7404465400336795, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:35:18,408] Trial 26 finished with value: 0.8950522668545926 and
parameters: {'C': 10.033346241683715, 'penalty': 'l1'}. Best is trial 19 with val
```

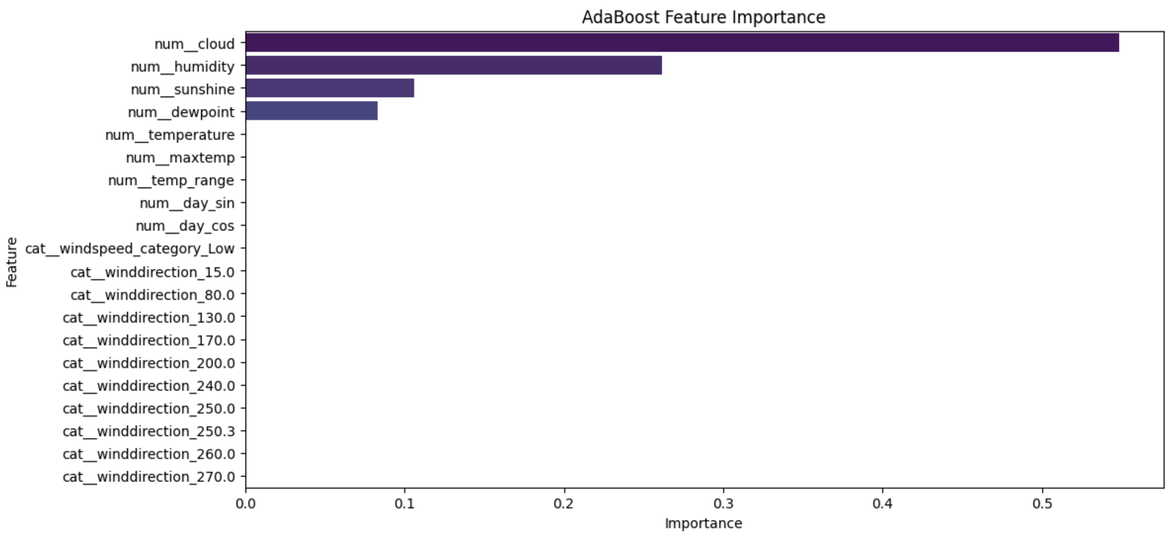
```
ue: 0.8951667841202726.
[I 2026-02-14 09:35:23,984] Trial 27 finished with value: 0.8948007541331519 and
parameters: {'C': 1.0442243816127308, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:35:27,516] Trial 28 finished with value: 0.8884338574194228 and
parameters: {'C': 0.1012445562390169, 'penalty': 'l1'}. Best is trial 19 with val
ue: 0.8951667841202726.
[I 2026-02-14 09:35:33,553] Trial 29 finished with value: 0.8949560764015455 and
parameters: {'C': 133.6640919857623, 'penalty': 'l1'}. Best is trial 19 with valu
e: 0.8951667841202726.
Best hyperparameters for each model:
RandomForest: {'n_estimators': 281, 'max_depth': 7, 'min_samples_split': 20, 'min
_samples_leaf': 4}
GradientBoosting: {'n_estimators': 273, 'learning_rate': 0.011782613983568804, 'm
ax_depth': 2, 'min_samples_split': 13, 'min_samples_leaf': 4}
AdaBoost: {'n_estimators': 138, 'learning_rate': 0.18265124489785745}
ExtraTrees: {'n_estimators': 238, 'max_depth': 11, 'min_samples_split': 10, 'min_
samples_leaf': 7}
LogisticRegression: {'C': 4.308661382633495, 'penalty': 'l1'}

==== RandomForest ====
Accuracy: 0.8767
ROC AUC: 0.8796
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.65      0.72       108
           1       0.89      0.95      0.92       330

    accuracy                           0.88       438
   macro avg       0.85      0.80      0.82       438
weighted avg       0.87      0.88      0.87       438
```



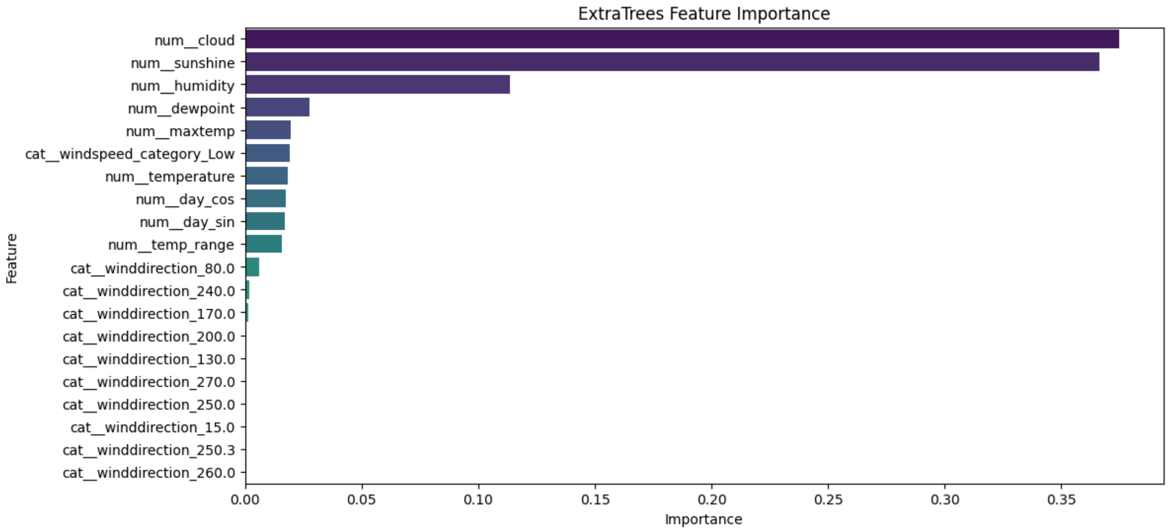
```
==== GradientBoosting ====
Accuracy: 0.8699
ROC AUC: 0.8753
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.62      0.70       108
           1       0.88      0.95      0.92       330

    accuracy                           0.87       438
   macro avg       0.85      0.79      0.81       438
weighted avg       0.87      0.87      0.86       438
```



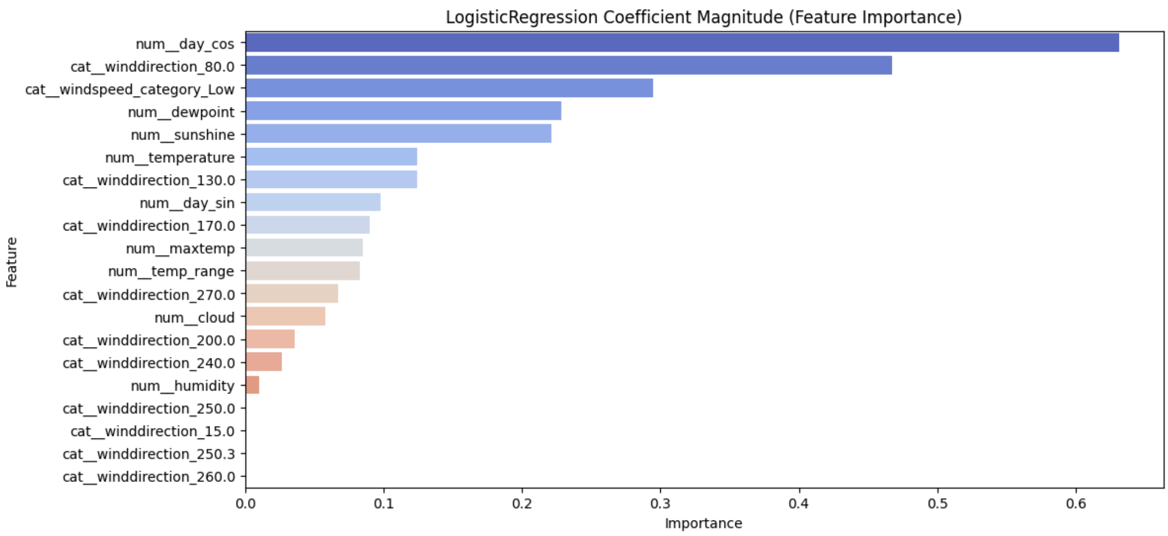

```
==== AdaBoost ====
Accuracy: 0.8676
ROC AUC: 0.8774
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.67      0.71       108
           1       0.90      0.93      0.91       330

    accuracy                           0.87       438
   macro avg       0.83      0.80      0.81       438
weighted avg       0.86      0.87      0.86       438
```



```
==== ExtraTrees ====
Accuracy: 0.8653
ROC AUC: 0.8789
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.59      0.68       108
           1       0.88      0.95      0.91       330

    accuracy                           0.87       438
   macro avg       0.84      0.77      0.80       438
weighted avg       0.86      0.87      0.86       438
```




```
==== LogisticRegression ====
Accuracy: 0.8630
ROC AUC: 0.8678
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.62      0.69       108
           1       0.88      0.94      0.91       330

    accuracy                           0.86       438
   macro avg       0.83      0.78      0.80       438
weighted avg       0.86      0.86      0.86       438
```



In [32]:
```
# STEP 4: Compare all models & select best

results = []
trained_models = {}

for name, model in models.items():
    # store trained model
    trained_models[name] = model

    # predictions
    y_pred = model.predict(X_val_transformed)
    if hasattr(model, "predict_proba"):
        y_pred_proba = model.predict_proba(X_val_transformed)[:, 1]
        roc_auc = roc_auc_score(y_val, y_pred_proba)
    else:
        roc_auc = None
    accuracy = accuracy_score(y_val, y_pred)

    # append results
    results.append({
        "Model": name,
        "Accuracy": accuracy,
        "ROC_AUC": roc_auc
    })

# Convert to DataFrame
results_df = pd.DataFrame(results).sort_values(by="ROC_AUC", ascending=False).re
display(results_df)

# Select best model
best_model_name = results_df.loc[0, "Model"]
best_model = trained_models[best_model_name]
print(f"Best model selected: {best_model_name}")
```

	Model	Accuracy	ROC_AUC
0	RandomForest	0.876712	0.879602
1	ExtraTrees	0.865297	0.878872
2	AdaBoost	0.867580	0.877357
3	GradientBoosting	0.869863	0.875309
4	LogisticRegression	0.863014	0.867789

```
Best model selected: RandomForest
```

10. Model Persistence/ Deployment

In [33]:
```
final_pipeline = Pipeline([
    ("preprocessing", preprocessing_mi_pipeline),  # your feature engineering +
    ("model", best_model)                           # the best trained model
])
```

In [34]:
```
# Save full pipeline (preprocessing + SMOTE + model)

MODEL_PATH = f"best_pipeline_{best_model_name}.joblib"
```

```
joblib.dump(final_pipeline, MODEL_PATH)
print(f"Pipeline saved to {MODEL_PATH}")
```

```
Pipeline saved to best_pipeline_RandomForest.joblib
```

In [35]:
```
# Fix column typo
if "temparature" in df_test.columns:
    df_test.rename(columns={"temparature": "temperature"}, inplace=True)
```

In [36]:
```
print(df_test.columns)
```

```
Index(['id', 'day', 'pressure', 'maxtemp', 'temperature', 'mintemp',
       'dewpoint', 'humidity', 'cloud', 'sunshine', 'winddirection',
       'windspeed'],
      dtype='object')
```

In [37]:
```
X_test = df_test  # no need to drop anything
```

In [38]:
```
loaded_pipeline = joblib.load(MODEL_PATH)

# Predicted class labels (rainfall or no rainfall)
y_pred = loaded_pipeline.predict(X_test)

# Predicted probabilities (for the positive class, e.g., rainfall)
y_proba = loaded_pipeline.predict_proba(X_test)[:, 1]
```

In [39]:
```
# Save predictions to CSV
output_df = X_test.copy()
output_df["predicted_rainfall"] = y_pred
output_df["rainfall_prob"] = y_proba

output_df.to_csv("rainfall_predictions.csv", index=False)
print("Predictions saved to rainfall_predictions.csv")
```

```
Predictions saved to rainfall_predictions.csv
```