بخش اول : آموزش و تست مدل

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
import joblib
```

In [2]:
```
# 1. بارگذاری دیتاست
# دیتاست CSV فایل
data = pd.read_csv('weather_forecast_data.csv')
data.head()
```

Out[2]:

	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain
0	23.720338	89.592641	7.335604	50.501694	1032.378759	rain
1	27.879734	46.489704	5.952484	4.990053	992.614190	no rain
2	25.069084	83.072843	1.371992	14.855784	1007.231620	no rain
3	23.622080	74.367758	7.050551	67.255282	982.632013	rain
4	20.591370	96.858822	4.643921	47.676444	980.825142	no rain

In [3]:
```
# 2. تبدیل ستون Rain به مقدار عددی
label_encoder = LabelEncoder()
data['Rain'] = label_encoder.fit_transform(data['Rain'])
data.head()
```

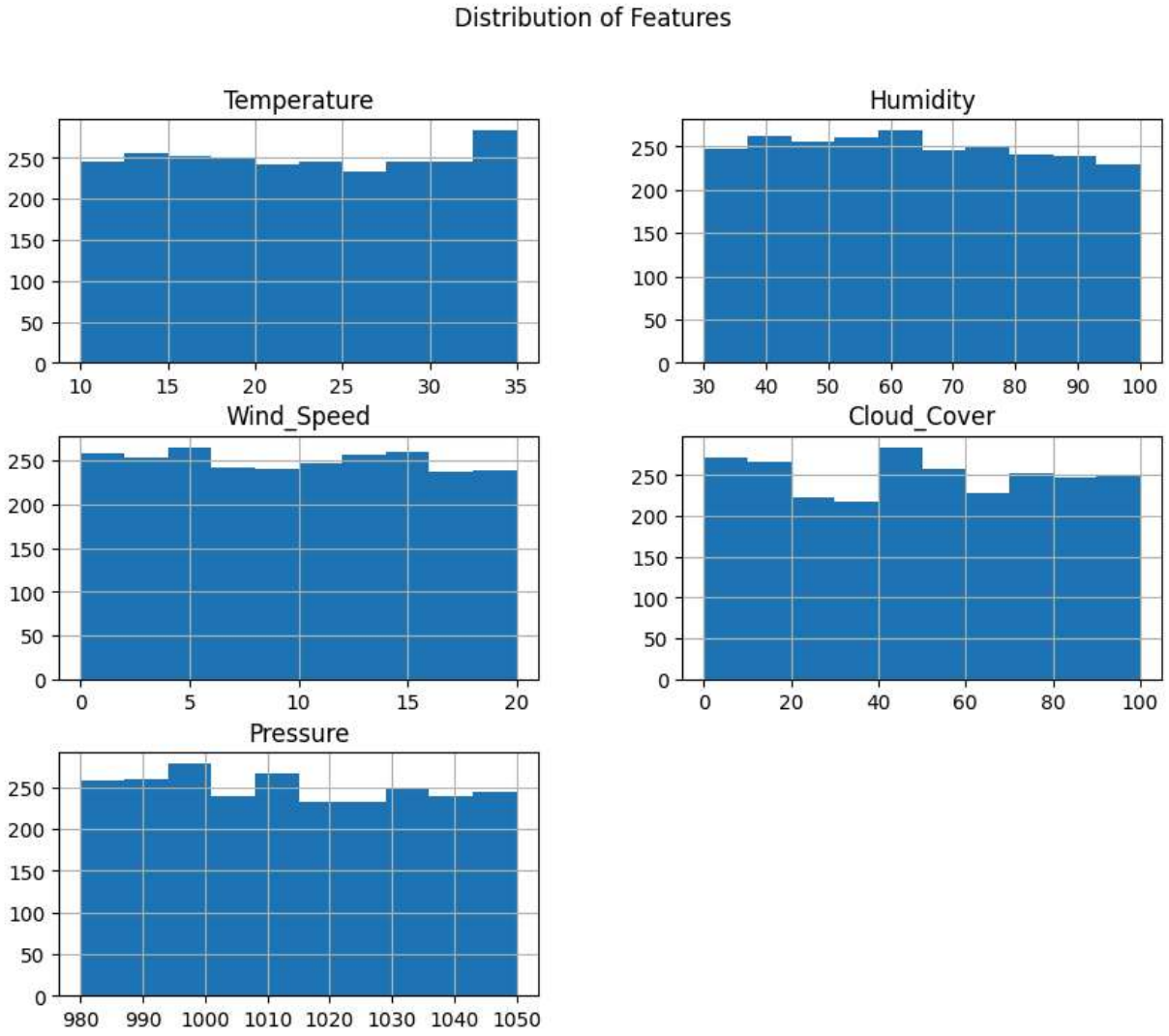
Out[3]:

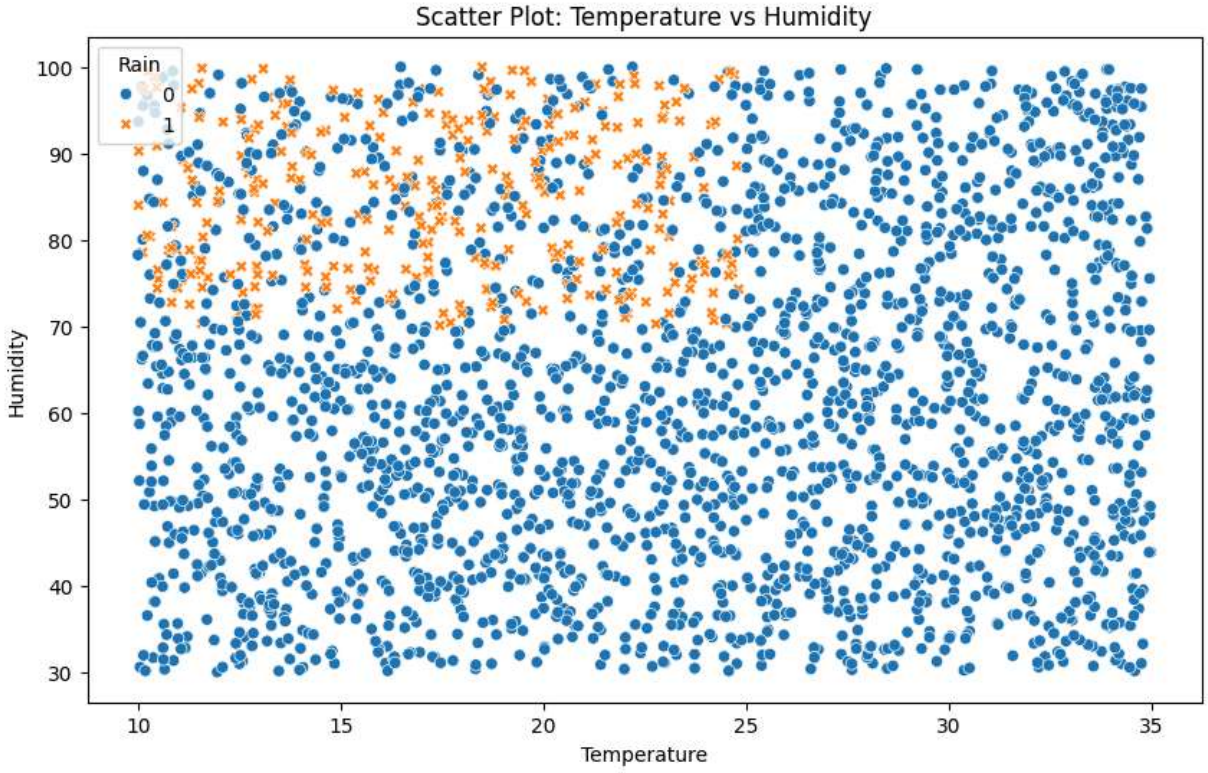
	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain
0	23.720338	89.592641	7.335604	50.501694	1032.378759	1
1	27.879734	46.489704	5.952484	4.990053	992.614190	0
2	25.069084	83.072843	1.371992	14.855784	1007.231620	0
3	23.622080	74.367758	7.050551	67.255282	982.632013	1
4	20.591370	96.858822	4.643921	47.676444	980.825142	0

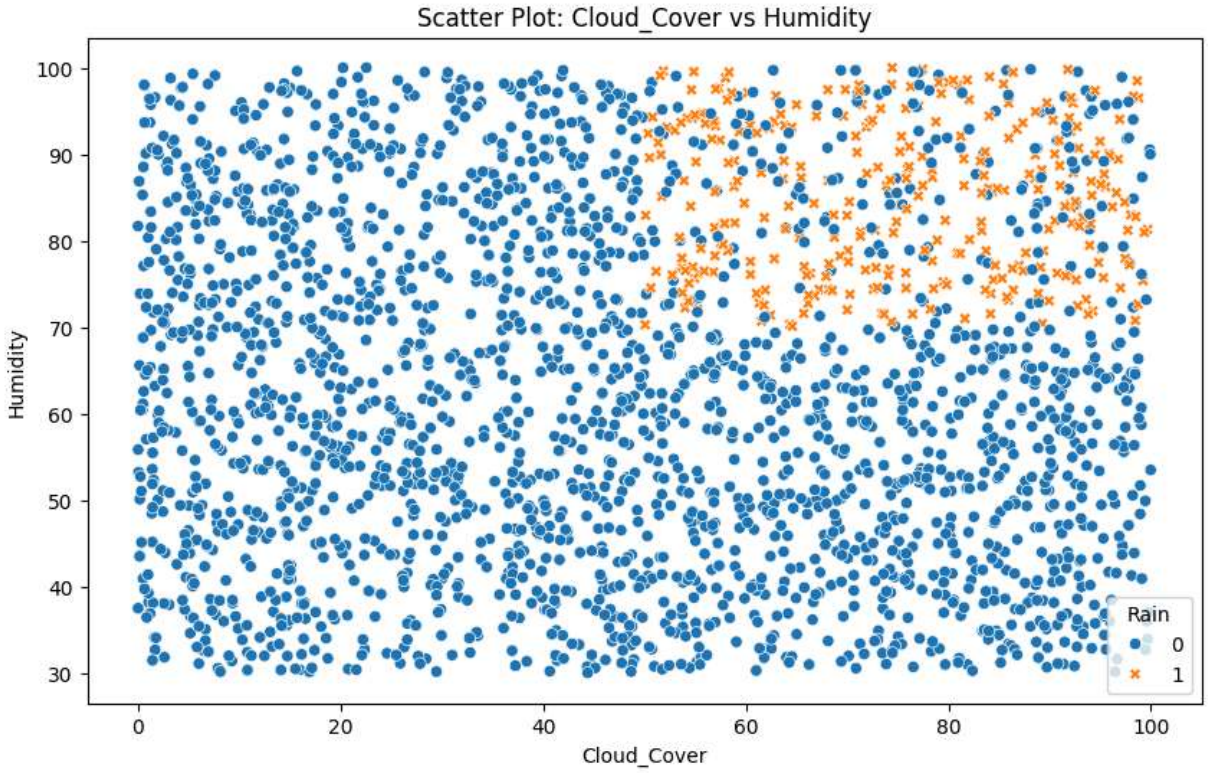
In [4]:
```
# 2.1 ذخیره کردن label encoder
joblib.dump(label_encoder, 'label_encoder.pkl')
```

Out[4]:
```
['label_encoder.pkl']
```

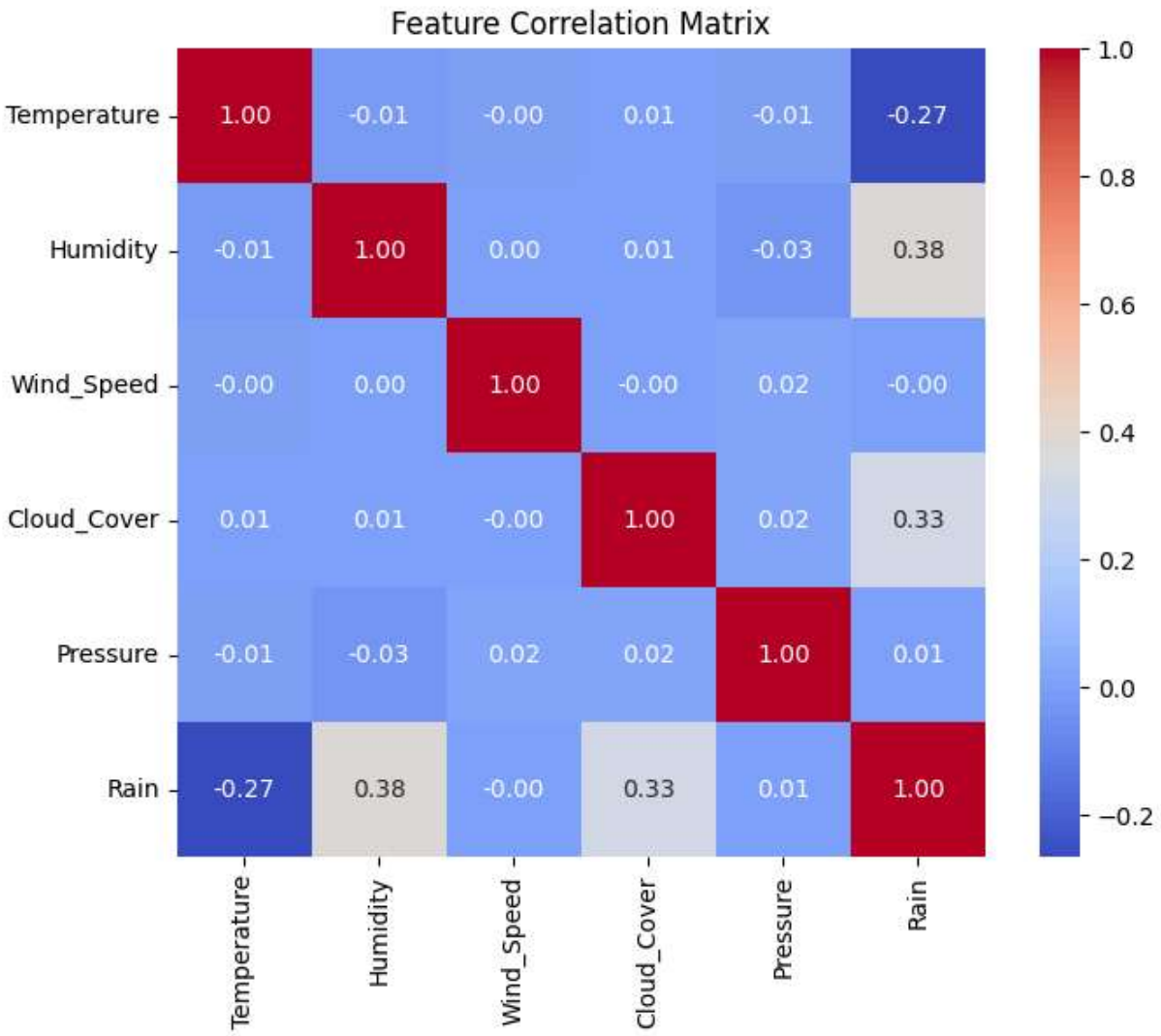
In [5]:
```
# 2.2 نمودار توزیع (Histogram) برای هر ویژگی
data.drop('Rain', axis=1).hist(bins=10, figsize=(10, 8))
plt.suptitle('Distribution of Features')
plt.show()

# 2.3 نمودار پراکندگی (Scatter Plot) بین ویژگی‌ها
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Temperature', y='Humidity',
                data=data, hue='Rain', style='Rain')
plt.title('Scatter Plot: Temperature vs Humidity')
plt.show()

# 2.4 نمودار پراکندگی (Scatter Plot) بین ویژگی‌ها
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Cloud_Cover', y='Humidity',
                data=data, hue='Rain', style='Rain')
plt.title('Scatter Plot: Cloud_Cover vs Humidity')
plt.show()
```


Distribution of Features
Temperature
Humidity
Wind_Speed
Cloud_Cover
Pressure


Scatter Plot: Temperature vs Humidity
Rain
0
1
Humidity
Temperature



In [6]:
```
# 3.1 محاسبه ماتریس همبستگی
correlation_matrix = data.corr()

# 3.2 رسم Heatmap از ماتریس همبستگی
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix,
            annot=True,
            fmt='.2f',
            cmap='coolwarm',
            cbar=True,
            square=True)
plt.title('Feature Correlation Matrix')
plt.show()
```



In [7]:
```
# 4.1 جداسازی ویژگی‌ها و برچسب هدف
X = data[['Temperature', 'Humidity', 'Wind_Speed', 'Cloud_Cover', 'Pressure']]
y = data['Rain']
```

In [8]:
```
# 4.2 تقسیم داده‌ها به مجموعه‌های آموزش و تست
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=42)
```

In [9]:
```
# 4.3 استانداردسازی داده‌ها
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4.4 ذخیره کردن scaler
joblib.dump(scaler, 'scaler.pkl')
```

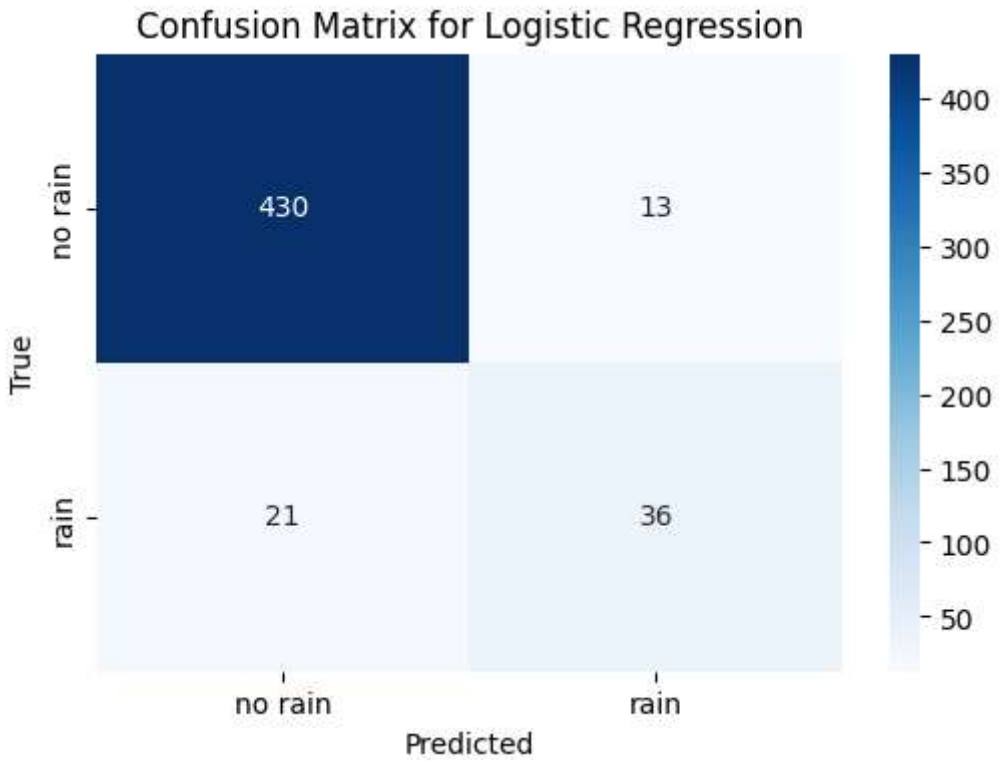
Out[9]:
```
['scaler.pkl']
```

In [10]:
```
# 5. تعریف مدل‌ها
models = {
    'Logistic Regression': LogisticRegression(),
```

```
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB(),
    'MLP': MLPClassifier(hidden_layer_sizes=(50, 50), max_iter=500, random_state=42
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingClassifier(n_estimators=100, random_state=
    'XGBoost': XGBClassifier(eval_metric='logloss', random_state=42),
    'KNN': KNeighborsClassifier(n_neighbors=5),
    'SVM': SVC(kernel='linear', random_state=42),
}
```

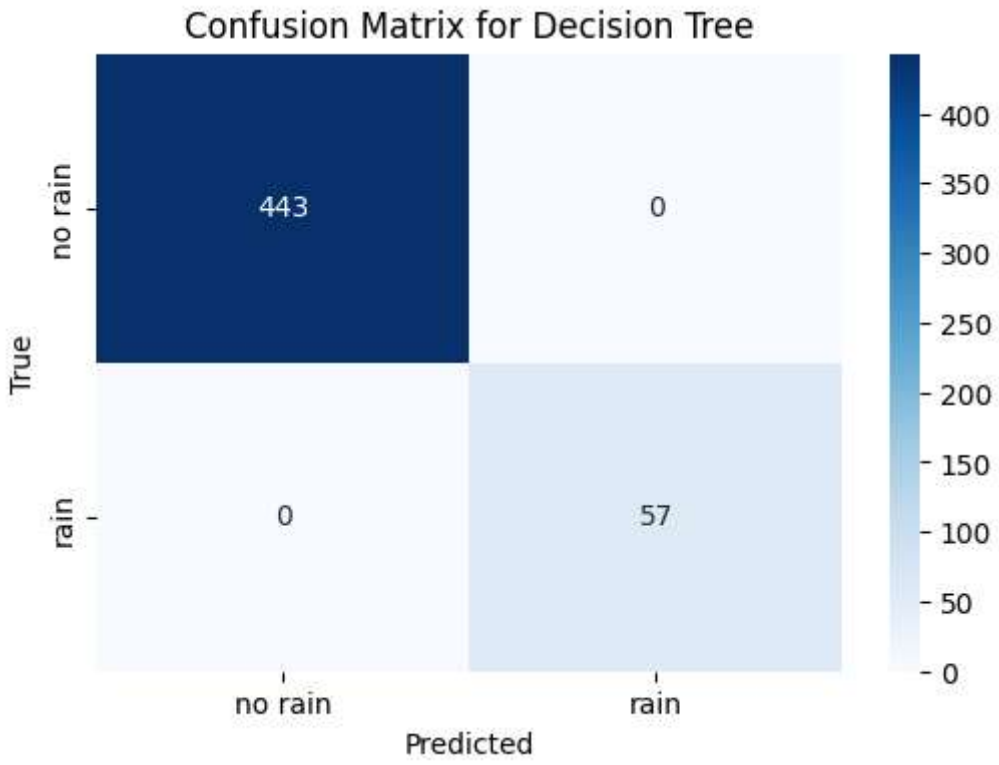
In [11]:
```
# 6. آموزش و ارزیابی مدل‌ها
results = {}
for model_name, model in models.items():
    print(f"Model: {model_name}")
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    # گزارش مدل
    print(classification_report(y_test, y_pred, zero_division=1))

    # ماتریس درهم‌ریختگی
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6, 4))
    sns.heatmap(cm, annot=True, fmt='d',
                cmap='Blues',
                xticklabels=label_encoder.classes_,
                yticklabels=label_encoder.classes_)
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.show()

    # دقت مدل
    accuracy = accuracy_score(y_test, y_pred)
    results[model_name] = accuracy

    # ذخیره مدل
    joblib.dump(model, f'{model_name}_model.pkl')
```

```
Model: Logistic Regression
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       443
           1       0.73      0.63      0.68        57

    accuracy                           0.93       500
   macro avg       0.84      0.80      0.82       500
weighted avg       0.93      0.93      0.93       500
```



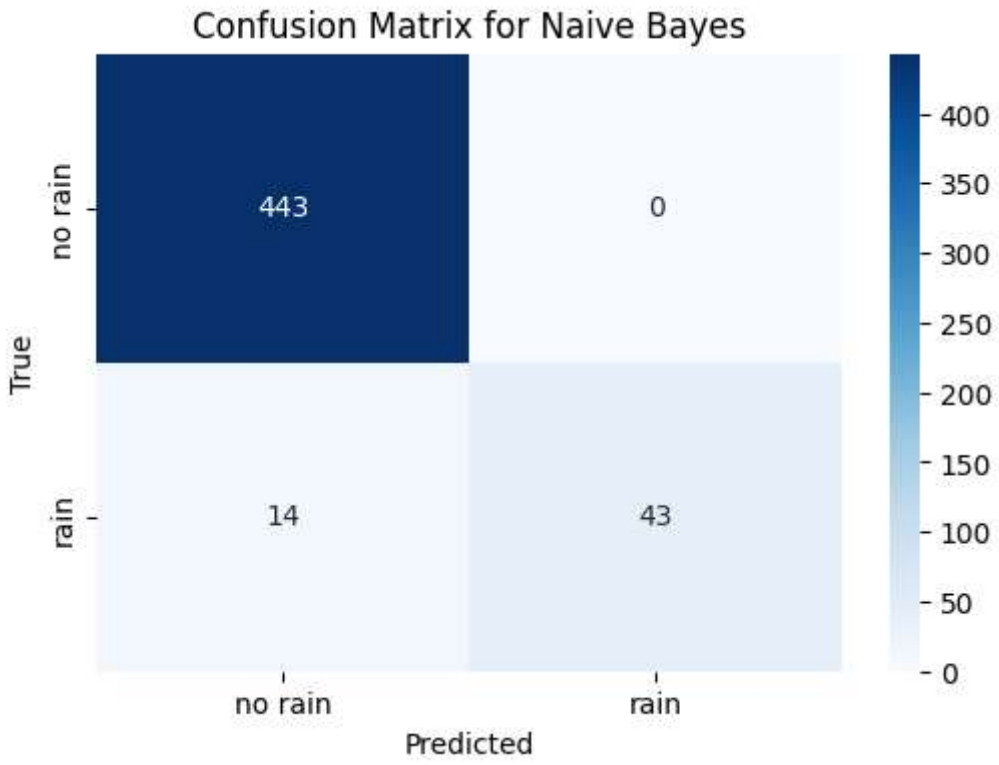
Model: Decision Tree

	precision	recall	f1-score	support
0	1.00	1.00	1.00	443
1	1.00	1.00	1.00	57
accuracy			1.00	500
macro avg	1.00	1.00	1.00	500
weighted avg	1.00	1.00	1.00	500



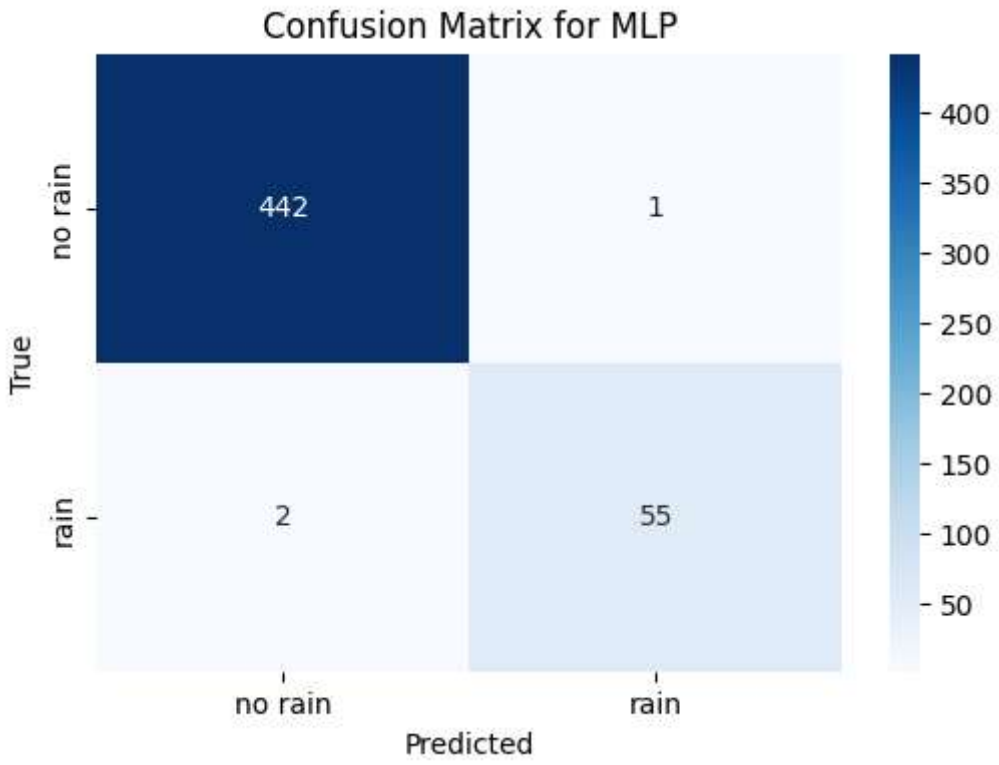
Model: Naive Bayes

	precision	recall	f1-score	support
0	0.97	1.00	0.98	443
1	1.00	0.75	0.86	57
accuracy			0.97	500
macro avg	0.98	0.88	0.92	500
weighted avg	0.97	0.97	0.97	500



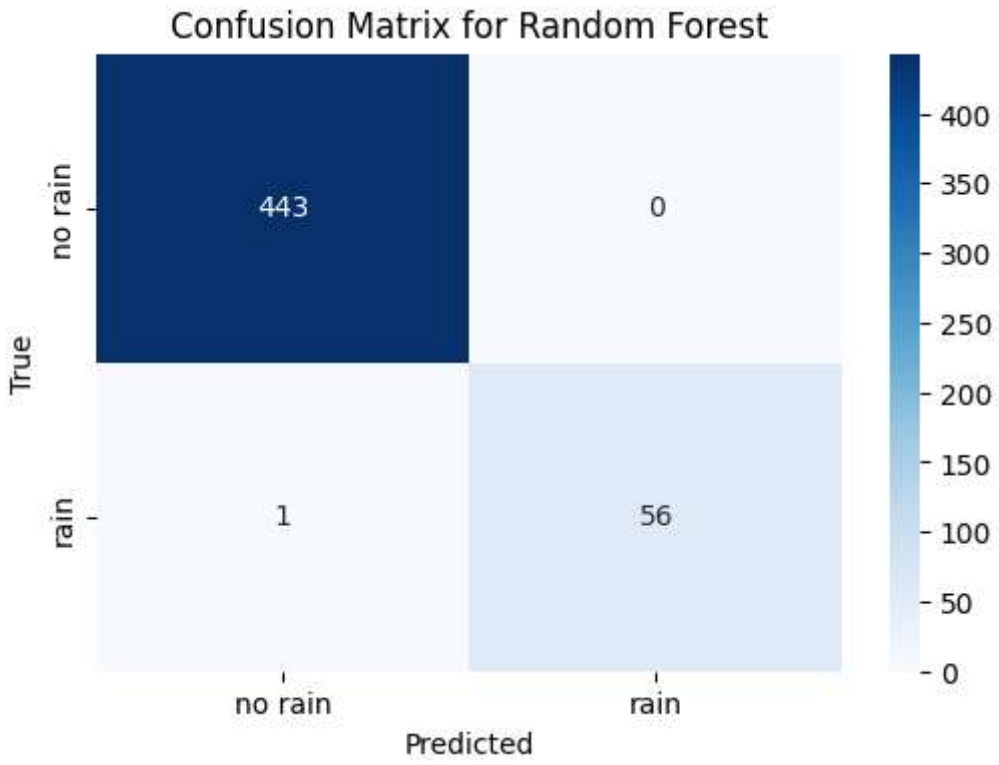
Model: MLP

	precision	recall	f1-score	support
0	1.00	1.00	1.00	443
1	0.98	0.96	0.97	57
accuracy			0.99	500
macro avg	0.99	0.98	0.99	500
weighted avg	0.99	0.99	0.99	500



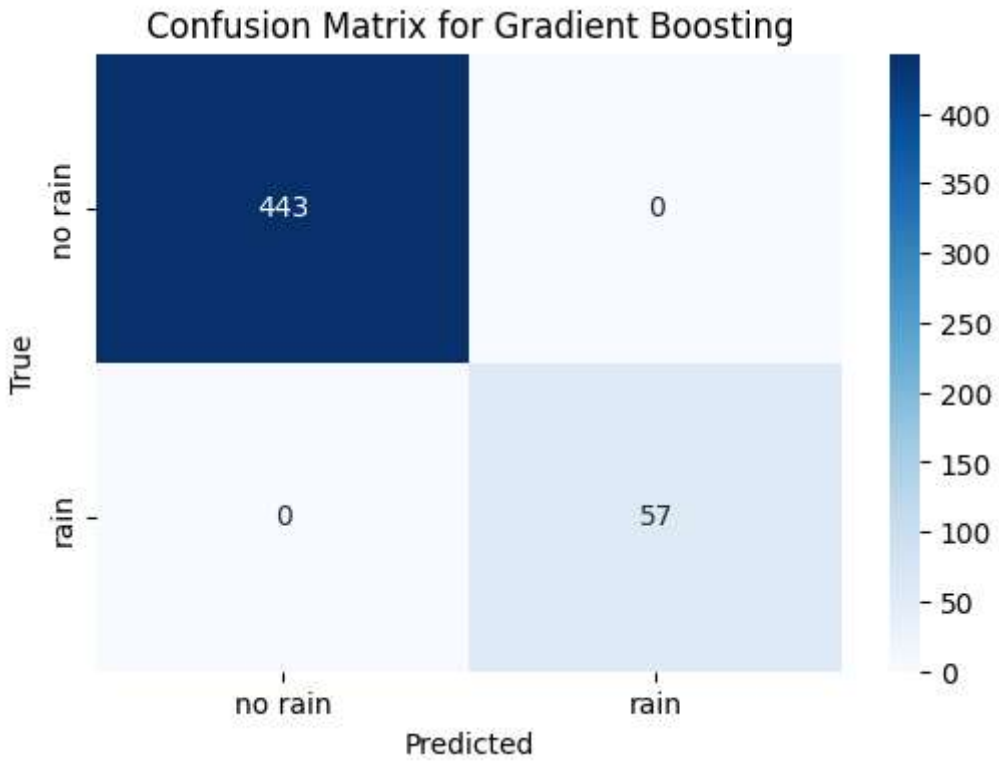
Model: Random Forest

	precision	recall	f1-score	support
0	1.00	1.00	1.00	443
1	1.00	0.98	0.99	57
accuracy			1.00	500
macro avg	1.00	0.99	1.00	500
weighted avg	1.00	1.00	1.00	500



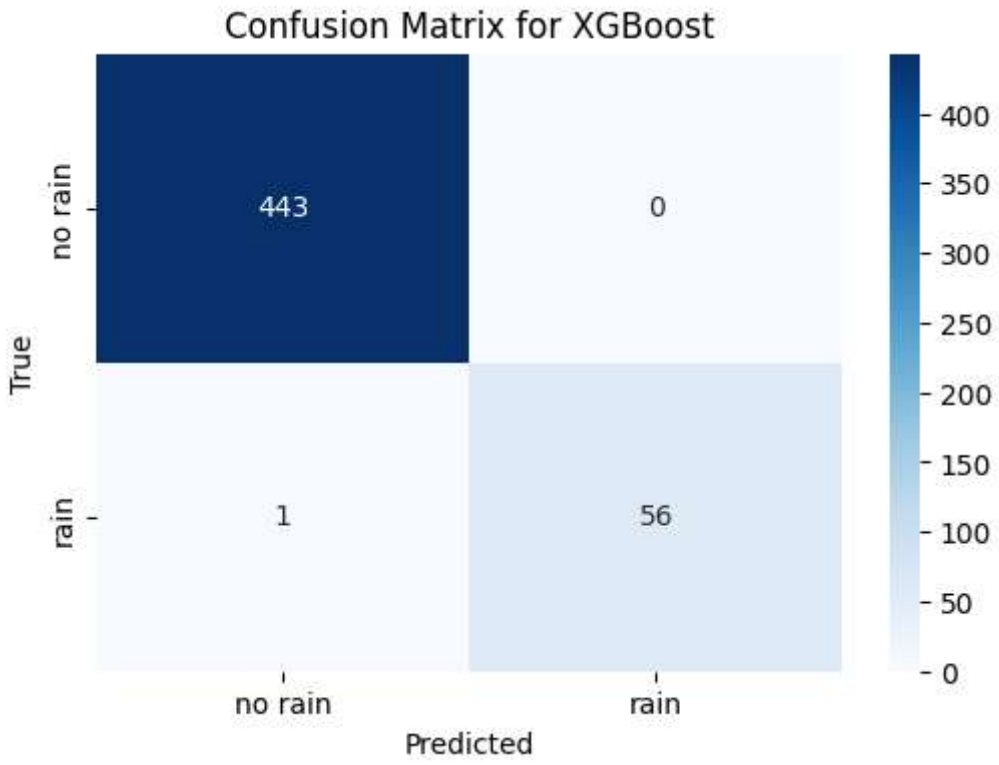
Model: Gradient Boosting

	precision	recall	f1-score	support
0	1.00	1.00	1.00	443
1	1.00	1.00	1.00	57
accuracy			1.00	500
macro avg	1.00	1.00	1.00	500
weighted avg	1.00	1.00	1.00	500



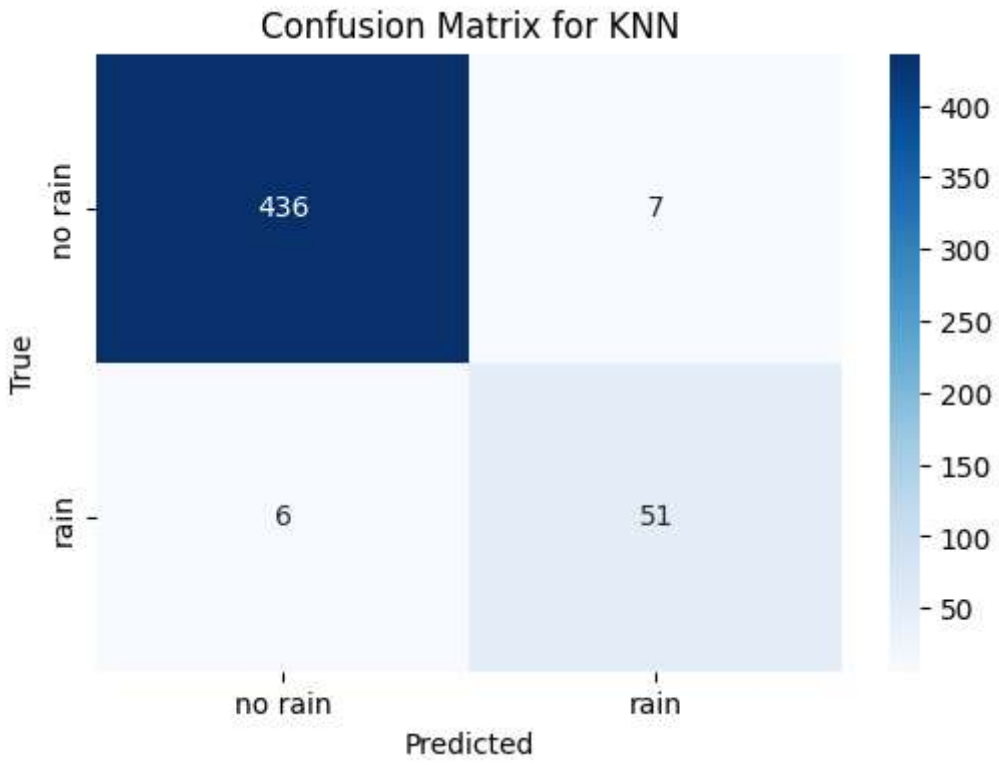
Model: XGBoost

	precision	recall	f1-score	support
0	1.00	1.00	1.00	443
1	1.00	0.98	0.99	57
accuracy			1.00	500
macro avg	1.00	0.99	1.00	500
weighted avg	1.00	1.00	1.00	500



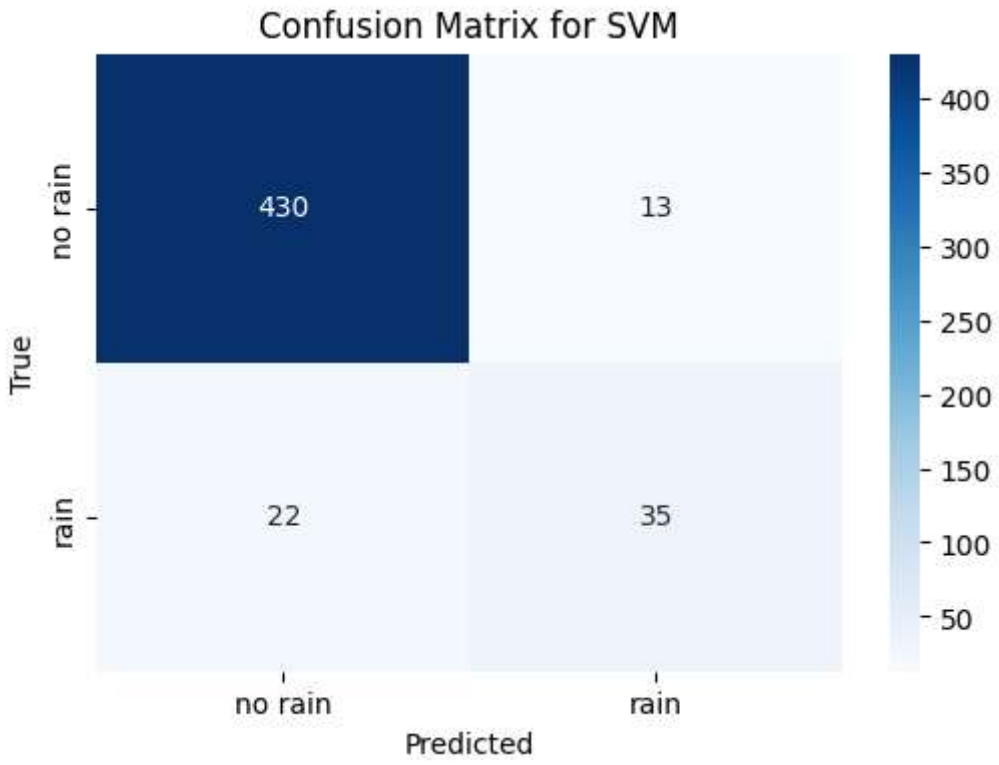
Model: KNN

	precision	recall	f1-score	support
0	0.99	0.98	0.99	443
1	0.88	0.89	0.89	57
accuracy			0.97	500
macro avg	0.93	0.94	0.94	500
weighted avg	0.97	0.97	0.97	500



Model: SVM

	precision	recall	f1-score	support
0	0.95	0.97	0.96	443
1	0.73	0.61	0.67	57
accuracy			0.93	500
macro avg	0.84	0.79	0.81	500
weighted avg	0.93	0.93	0.93	500



In [12]:
```
# 7. نمایش نتایج
print("Accuracy Results:")
for model_name, accuracy in results.items():
    print(f"{model_name}: {accuracy:.2f}")
```

```
Accuracy Results:
Logistic Regression: 0.93
Decision Tree: 1.00
Naive Bayes: 0.97
MLP: 0.99
Random Forest: 1.00
Gradient Boosting: 1.00
XGBoost: 1.00
KNN: 0.97
SVM: 0.93
```

پایان آموزش و ذخیره مدل

بخش دوم: مراحل استفاده از مدل ذخیره شده

In [13]:
```
# 1. بارگذاری مدل‌ها (اگر نیاز دارید از مدل‌های ذخیره‌شده استفاده کنید)
models_loaded = {}
for model_name in models.keys():
    models_loaded[model_name] = joblib.load(f'{model_name}_model.pkl')
```

In [14]:
```
# 2. استفاده از مدل‌های بارگذاری‌شده
# مثلا برای پیش‌بینی با مدل 'Random Forest'
y_pred_loaded = models_loaded['Random Forest'].predict(X_test_scaled)
# گزارش مدل لود شده
print(classification_report(y_test, y_pred_loaded, zero_division=1))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       443
           1       1.00      0.98      0.99        57

    accuracy                           1.00       500
   macro avg       1.00      0.99      1.00       500
weighted avg       1.00      1.00      1.00       500
```

In [15]:
```
# 3. پیش بینی با داده های جدید
loaded_model = joblib.load('Decision Tree_model.pkl')
scaler = joblib.load('scaler.pkl')
label_encoder = joblib.load('label_encoder.pkl')
```

In [16]:
```
# 3.1 ایجاد داده تست
test_df = pd.DataFrame({
    'Temperature': [22.5, 12],
    'Humidity': [85.0, 60],
    'Wind_Speed': [5.2, 0],
    'Cloud_Cover': [65.0, 95],
    'Pressure': [1015.0, 900]
})
test_df.head()
```

Out[16]:

	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure
0	22.5	85.0	5.2	65.0	1015.0
1	12.0	60.0	0.0	95.0	900.0

In [17]:
```
# 3.2 استفاده از آنها برای پیش‌بینی روی داده‌های جدید
# داشته باشند scale درصورتیکه داده ها نیاز به
X_new_scaled = scaler.transform(test_df)
predict = loaded_model.predict(X_new_scaled)
predicted_labels = label_encoder.inverse_transform(predict)
print(predicted_labels)
```

```
['rain' 'no rain']
```

In [ ]: