Rising Waters: A Machine Learning Approach to Flood Prediction

Project Structure:

```
rising-waters-flood-prediction/
│
├── data/
│   └── flood.csv
├── models/
│   └── model.pkl
├── static/
├── templates/
│   ├── index.html
│   └── result.html
├── train_model.py
├── app.py
├── requirements.txt
├── README.md
└── .gitignore
```

requirements.txt:

```
pandas
numpy
matplotlib
seaborn
scikit-learn
flask
joblib
```

data/flood.csv:

```
Rainfall,Temperature,Humidity,River_Level,Wind_Speed,Flood
120,30,85,4.5,15,1
80,28,70,3.2,10,0
200,32,90,5.8,20,1
60,25,60,2.5,8,0
150,29,88,4.9,18,1
90,27,75,3.8,12,0
```

110,31,82,4.1,14,1
70,26,65,3.0,9,0

train_model.py:

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

data = pd.read_csv("data/flood.csv")

plt.figure()
sns.heatmap(data.corr(), annot=True)
plt.title("Feature Correlation")
plt.savefig("static/correlation.png")
plt.close()

X = data.drop("Flood", axis=1)
y = data["Flood"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
   X_scaled, y, test_size=0.2, random_state=42
)

model = RandomForestClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
joblib.dump(model, "models/model.pkl")
joblib.dump(scaler, "models/scaler.pkl")
```

app.py:

```
from flask import Flask, render_template, request
import numpy as np
import joblib

app = Flask(__name__)

model = joblib.load("models/model.pkl")
scaler = joblib.load("models/scaler.pkl")

@app.route('/')
def home():
    return render_template("index.html")

@app.route('/predict', methods=['POST'])
def predict():
    features = [
        float(request.form['rainfall']),
        float(request.form['temperature']),
        float(request.form['humidity']),
        float(request.form['river_level']),
        float(request.form['wind_speed'])
    ]

    features = np.array([features])
    features_scaled = scaler.transform(features)

    prediction = model.predict(features_scaled)

    if prediction[0] == 1:
        result = "High Risk of Flood!"
    else:
        result = "No Flood Risk."

    return render_template("result.html", prediction=result)

if __name__ == "__main__":
    app.run(debug=True)
```

templates/index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Flood Prediction</title>
</head>
<body>
  <h2>Rising Waters - Flood Prediction</h2>

  <form action="/predict" method="post">
    Rainfall: <input type="number" step="any" name="rainfall"><br><br>
    Temperature: <input type="number" step="any" name="temperature"><br><br>
    Humidity: <input type="number" step="any" name="humidity"><br><br>
    River Level: <input type="number" step="any" name="river_level"><br><br>
    Wind Speed: <input type="number" step="any" name="wind_speed"><br><br>
    <input type="submit" value="Predict">
  </form>

  <br>
  <img src="/static/correlation.png" width="500">
</body>
</html>
```

templates/result.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Prediction Result</title>
</head>
<body>
  <h2>Prediction Result</h2>
  <h3>{{ prediction }}</h3>
  <a href="/">Go Back</a>
</body>
</html>
```

.gitignore:

__pycache__/
*.pyc
models/model.pkl
models/scaler.pkl

README.md:

Rising Waters: Flood Prediction using Machine Learning

This project predicts flood occurrence using environmental parameters such as rainfall, humidity, river level, temperature, and wind speed.

How to Run:
1. pip install -r requirements.txt
2. python train_model.py
3. python app.py
4. Open http://127.0.0.1:5000/