

Date	3 oct 2025
Team ID	LTVIP2025TMIDS67798
Project Name	RAINFALL PREDICTION USING MACHINE LEARNING
Maximum Marks	10 Marks

## 1. Initial Model Training Code

This template demonstrates how to train a machine learning model for rainfall prediction using Python's scikit-learn library.

File Edit Selection View Go Run Terminal Help ↺ ↻ 🔍 rain (2) 08 29C BLACKBOX ...

EXPLORER: RAIN (2)

- Demo video
- Rainfall prediction.mp4
- README.md

rain

- static
  - bg\_1.jpg
  - bg\_rain.jpg
- templates
  - index.html
  - results.html
- venv
  - include
  - Lib
  - Scripts
    - gunicorn
    - pyenv.cfg

app.py 9+ TODO.md

```
rain > app.py > ...
1   from flask import Flask, render_template, request, jsonify, session, redirect, url_for
2   import os
3   import pickle
4   import requests
5   import numpy as np
6   from datetime import datetime, timedelta
7   from sklearn.ensemble import RandomForestRegressor
8   from sklearn.model_selection import train_test_split
9
10  app = Flask(__name__)
11  app.secret_key = 'your_secret_key_here'
12  MODEL_PATH = "model.pkl"
13
14  # ----- Model Setup -----
15  def create_and_save_dummy_model(path=MODEL_PATH):
16      np.random.seed(42)
17      N = 8000
18      lats = np.random.uniform(-60, 60, N)
19      lons = np.random.uniform(-180, 180, N)
20      months = np.random.randint(1, 13, N)
21      days = np.random.randint(1, 29, N)
22
23      X, y = [], []
24      for lat, lon, m, d in zip(lats, lons, months, days):
25          temp = 30 - abs(lat)/3 + 5*np.sin(*np.pi*(m-1)/12)
26          humidity = 60 + 20*np.cos(2*np.pi*(m-1)/12) - (abs(lat)/90)**30
27          wind = 5 + 2*np.sin(2*np.pi*d/30) + (abs(lon)/180)**2
28          elevation = max(0, 500 - abs(lat)*5 + (lon % 90))
29          x.append([temp, humidity, wind, elevation])
30          rainfall = max(0, 0.6*humidity - 0.4*temp + 0.02*elevation + np.random.normal(0,8))/4
31          y.append(rainfall)
32
33  X_train, X_test, y_train, y_test = train_test_split(np.array(X), np.array(y), test_size=0.2, random_state=42)
34  model = RandomForestRegressor(n_estimators=100, random_state=42)
35  model.fit(X_train, y_train)
36
37  with open(path, "wb") as f:
```

Generate Commit Message 📁 sras-junnu19 (20 hours ago) Ln 10, Col 22 Spaces: 4 UTF-8 CR LF 🐍 Python 🏛 3.13.8 (Microsoft Store) ⚡ Go Live 🔍 BLACKBOXAI: Open Chat

29C BLACKBOX Agent Open Website

7 29C Rain showers

Search ENG IN 13-10-2025

## 2. Model Validation and Evaluation Template

After training your model, it's crucial to validate and evaluate its performance. Here's how you can do it:

## a. Cross-Validation

Implementing cross-validation helps in assessing the model's performance across different subsets of the dataset.

```
from sklearn.model_selection import cross_val_score

cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_absolute_error')
print(f'Cross-validated MAE: {-cv_scores.mean()}')
```

## b. Hyperparameter Tuning

Fine-tuning the model's hyperparameters can enhance its performance.

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

print(f'Best Parameters: {grid_search.best_params_}')
```

## c. Model Evaluation Metrics

Utilize various metrics to evaluate your model's performance comprehensively.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred = grid_search.best_estimator_.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

## d. Residual Analysis

Analyzing residuals helps in understanding the model's prediction errors.

```
residuals = y_test - y_pred
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Rainfall')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```

## 3. Model Deployment and Monitoring

Once your model is trained and evaluated, consider the following steps for deployment and monitoring:

- **Deployment:** Use frameworks like Flask or FastAPI to create a web service for your model.
- **Monitoring:** Implement logging and monitoring to track the model's performance over time.
- **Retraining:** Periodically retrain the model with new data to maintain its accuracy.