🌧️ Model Optimization and Tuning Phase

Project Name: Rainfall Prediction

Team: Lakshmi Sravya Savaram

Mohammad Shouqat Azeez

N Gokul Chowdary

Nallabotula Vijaya Karthik

Date: 13/10/2025

Objective

The objective of this phase is to improve the predictive performance of the selected model by optimizing hyperparameters, performing feature selection, and applying model-specific tuning strategies. This ensures better accuracy, generalization, and robustness for rainfall prediction.

Selected Model for Tuning

Model	Reason for Selection
[e.g., XGBoost Regressor]	Achieved highest R^2 and lowest error metrics during the Model Selection Phase; handles non-linear relationships effectively

---

Hyperparameter Tuning

a. Grid Search

Systematically explores combinations of hyperparameters to find the best configuration.

Example (XGBoost):

```python
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV

model = XGBRegressor(random_state=42)
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.7, 0.8, 1.0]
}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
                           scoring='neg_mean_absolute_error', cv=5, verbose=2, n_jobs=-1)
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

b. Randomized Search

Randomly samples hyperparameters, useful for large search spaces.

```python
from sklearn.model_selection import RandomizedSearchCV

random_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid,
                                   n_iter=20, scoring='neg_mean_absolute_error',
                                   cv=5, verbose=2, random_state=42, n_jobs=-1)
random_search.fit(X_train, y_train)

print("Best Parameters:", random_search.best_params_)
```

Feature Selection and Engineering

```python
import matplotlib.pyplot as plt
import pandas as pd

importance = grid_search.best_estimator_.feature_importances_
features = X_train.columns
plt.barh(features, importance)
plt.title("Feature Importance")
plt.show()
```

- Remove low-importance features to reduce overfitting and improve generalization.
- Create new features (lag variables, rolling averages) if needed for temporal data.

Regularization and Model-Specific Tuning

- XGBoost / Gradient Boosting: Tune gamma, min_child_weight, colsample_bytree for regularization.
- Random Forest: Adjust max_depth, min_samples_split, and max_features to avoid overfitting.
- SVR / Linear Models: Tune regularization parameter C and kernel parameters.

Validation

- Use k-fold cross-validation to evaluate model performance with tuned hyperparameters.
- Compare MAE, MSE, RMSE, R^2 before and after tuning to assess improvement.

```
from sklearn.model_selection import cross_val_score
import numpy as np

cv_scores = cross_val_score(grid_search.best_estimator_, X_train, y_train, cv=5,
                            scoring='neg_mean_absolute_error')
print("Mean CV MAE:", -np.mean(cv_scores))
```

Results Summary

Metric	Before Tuning	After Tuning
MAE	[value]	[value]
MSE	[value]	[value]
RMSE	[value]	[value]
R^2	[value]	[value]

Note: Fill [value] with actual results from your experiments.

Conclusion

- Optimized hyperparameters and refined features improved model performance.
- Selected model is now ready for **final evaluation and deployment**.

- Future steps: monitor model on new data, periodically retrain, and update features if necessary.