**Model Evaluation and Tuning

Problem Statement: Use cross-validation and grid search to tune hyperparameters of a classification model (e.g., SVM or Random Forest).

Activities:

Split data using K-Fold

Implement GridSearchCV

Analyze accuracy improvement**

```
 # ------------------------------------------
# MODEL EVALUATION AND TUNING
# Using K-Fold Cross Validation and GridSearchCV
# ------------------------------------------

# Step 1: Import Libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, KFold, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import load_iris

# Step 2: Load Dataset
iris = load_iris()
X = iris.data
y = iris.target

# Step 3: Split Data into Training and Testing Sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Step 4: Define Base Model
base_model = RandomForestClassifier(random_state=42)
base_model.fit(X_train, y_train)
base_pred = base_model.predict(X_test)

# Step 5: Evaluate Base Model
base_accuracy = accuracy_score(y_test, base_pred)
print("Base Model Accuracy:", round(base_accuracy, 4))

# Step 6: Set Up K-Fold Cross Validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

# Step 7: Define Parameter Grid for Tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

# Step 8: Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(random_state=42),
    param_grid=param_grid,
    cv=kfold,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)

# Step 9: Fit GridSearchCV
grid_search.fit(X_train, y_train)

# Step 10: Best Hyperparameters and Best Score
print("\nBest Parameters:", grid_search.best_params_)
print("Best Cross-Validation Accuracy:", round(grid_search.best_score_, 4))

# Step 11: Evaluate Tuned Model on Test Data
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
tuned_accuracy = accuracy_score(y_test, y_pred)

print("\nTuned Model Accuracy:", round(tuned_accuracy, 4))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
# Step 12: Compare Results
print("\n--- Accuracy Improvement ---")
print(f"Before Tuning: {round(base_accuracy, 4)}")
print(f"After Tuning:  {round(tuned_accuracy, 4)}")
print(f"Improvement:   {round(tuned_accuracy - base_accuracy, 4)}")
```

```
Base Model Accuracy: 0.9
Fitting 5 folds for each of 36 candidates, totalling 180 fits

Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}
Best Cross-Validation Accuracy: 0.9417

Tuned Model Accuracy: 0.9667

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30


--- Accuracy Improvement ---
Before Tuning: 0.9
After Tuning:  0.9667
Improvement:   0.0667
```