EXPERIMENT 8: Model Evaluation and Improvement: Hyperparameter Tuning with Grid Search and Cross-Validation

AIM:

To demonstrate key techniques for model evaluation and improvement:

1. Hyperparameter Tuning with Grid Search: Systematically searching for the optimal combination of hyperparameters for a machine learning model.

2. Cross-Validation Techniques: Implementing k-fold cross-validation to get a more robust estimate of model performance and to prevent overfitting to a specific train-test split.

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
print("--- Part 1: Hyperparameter Tuning with Grid Search ---")
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names
print(f"\nDataset Features (X) shape: {X.shape}")
print(f"Dataset Labels (y) shape: {y.shape}")
print(f"Feature Names: {feature_names}")
print(f"Target Names: {target_names}")
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y)
print(f"\nTraining set size: {X_train.shape[0]} samples")
print(f"Test set size: {X_test.shape[0]} samples")
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
print("\nFeatures standardized.")
param_grid = {
    'C': [0.1, 1, 10],
    'gamma': [0.1, 0.01],
    'kernel': ['rbf', 'linear']
}
print("\nReduced Hyperparameter grid defined:")
for param, values in param_grid.items():
    print(f" {param}: {values}")
grid_search = GridSearchCV(
    SVC(),
    param_grid,
    cv=3,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1
)

print("\nStarting Grid Search with 3-fold Cross-Validation...")
grid_search.fit(X_train_scaled, y_train)
print("\nGrid Search completed.")
print(f"\nBest hyperparameters found: {grid_search.best_params_}")
print(f"Best cross-validation accuracy: {grid_search.best_score_:.4f}")
best_model = grid_search.best_estimator_
y_pred_tuned = best_model.predict(X_test_scaled)
test_accuracy_tuned = accuracy_score(y_test, y_pred_tuned)
print(f"\nTest set accuracy with tuned model: {test_accuracy_tuned:.4f}")

print("\n--- Classification Report for Tuned Model ---")
```

```
print(classification_report(y_test, y_pred_tuned, target_names=target_names))

print("\n--- Confusion Matrix for Tuned Model ---")
cm_tuned = confusion_matrix(y_test, y_pred_tuned)
plt.figure(figsize=(6, 5))
sns.heatmap(cm_tuned, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_names, yticklabels=target_names)
plt.title('Confusion Matrix (Tuned SVM)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
results_df = pd.DataFrame(grid_search.cv_results_)
print("\n--- Top 5 Grid Search Results ---")
print(results_df[['param_C', 'param_gamma', 'param_kernel', 'mean_test_score',
                 'rank_test_score']].sort_values(by='rank_test_score').head())

print("\n--- Part 2: Cross-Validation Techniques (k-fold) ---")
model_cv = SVC(random_state=42)
k_folds = 3
kf = KFold(n_splits=k_folds, shuffle=True, random_state=42)
print(f"\nPerforming {k_folds}-fold cross-validation...")
cv_scores = cross_val_score(model_cv, X_train_scaled, y_train, cv=kf, scoring='accuracy')
print(f"\nCross-validation scores for each fold: {cv_scores}")
print(f"Mean CV accuracy: {np.mean(cv_scores):.4f}")
print(f"Std dev of CV accuracy: {np.std(cv_scores):.4f}")
plt.figure(figsize=(6, 4))
plt.bar(range(1, k_folds + 1), cv_scores, color='skyblue')
plt.axhline(y=np.mean(cv_scores), color='r', linestyle='--',
            label=f'Mean Accuracy ({np.mean(cv_scores):.4f})')
plt.title(f'{k_folds}-Fold Cross-Validation Accuracy Scores')
plt.xlabel('Fold Number')
plt.ylabel('Accuracy')
```

```
plt.ylim(0.8, 1.0)
plt.legend()
plt.grid(axis='y', linestyle='--')
plt.show()
print("\n--- Why is Cross-Validation Important? ---")
print("1. More Reliable Performance Estimate")
print("2. Better Generalization")
print("3. Efficient Data Usage")
print("4. Detects Overfitting/Underfitting")
```

OUTPUT:

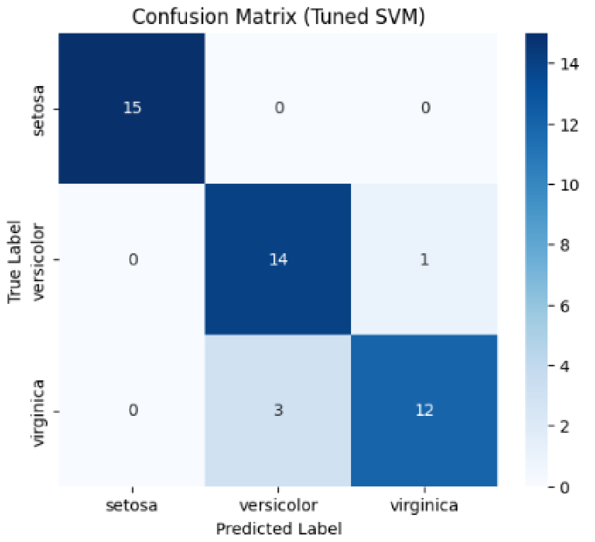
```
Test set accuracy with tuned model: 0.9111

--- Classification Report for Tuned Model ---
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.82      0.93      0.88        15
   virginica       0.92      0.80      0.86        15

    accuracy                           0.91        45
   macro avg       0.92      0.91      0.91        45
weighted avg       0.92      0.91      0.91        45


--- Confusion Matrix for Tuned Model ---
```




```
--- Top 5 Grid Search Results ---
    param_C  param_gamma param_kernel  mean_test_score  rank_test_score
7       1.0         0.01       linear         0.980952                1
5       1.0         0.10       linear         0.980952                1
10     10.0         0.01          rbf         0.980952                1
8      10.0         0.10          rbf         0.971429                4
4       1.0         0.10          rbf         0.961905                5
```