EXPERIMENT 4A :Support Vector Machines (SVM)

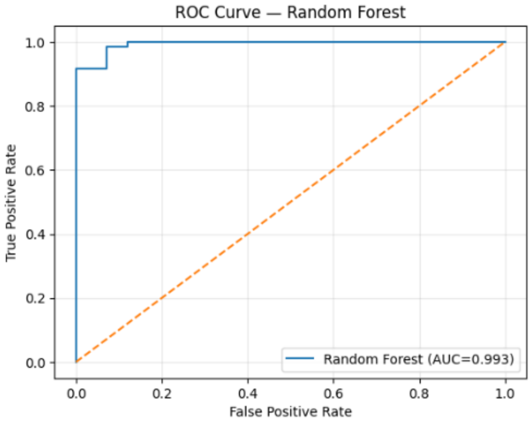
AIM:

To build an SVM model for a binary classification task, tune its hyperparameters, and evaluate it using accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC.

SOURCE CODE:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report, roc_auc_score, roc_curve
)
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="target") # 1 = malignant? (Check dataset doc: in this set,
0==malignant, 1==benign
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.20, random_state=42, stratify=y
)
scaler = StandardScaler()
X_train_sc = scaler.fit_transform(X_train)
X_test_sc = scaler.transform(X_test)
svm = SVC(kernel='rbf', probability=True, random_state=42)
param_grid = {
"C": [0.1, 1, 10, 100],
"gamma": ["scale", 0.01, 0.001, 0.0001]
}
grid = GridSearchCV(
estimator=svm,
param_grid=param_grid,
scoring='f1', # you can change to 'accuracy' or 'roc_auc' as needed
cv=5,
n_jobs=-1,
verbose=0
)
grid.fit(X_train_sc, y_train)
print("Best Params (CV):", grid.best_params_)
best_svm = grid.best_estimator_
best_svm.fit(X_train_sc, y_train)
y_pred = best_svm.predict(X_test_sc)
y_prob = best_svm.predict_proba(X_test_sc)[:, 1]
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, zero_division=0)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_prob)
cm = confusion_matrix(y_test, y_pred)
print("\n=== SVM (RBF) — Test Metrics ===")
print(f"Accuracy : {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall : {rec:.4f}")
print(f"F1-Score : {f1:.4f}")
print(f"ROC-AUC : {auc:.4f}")
print("\nConfusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=0))
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
plt.figure()
plt.plot(fpr, tpr, label=f"SVM (AUC={auc:.3f})")
plt.plot([0, 1], [0, 1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve — SVM (RBF)")
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

OUTPUT:



EXPERIMENT 4B : Ensemble Methods: Random Forest

AIM :

To implement a Random Forest classifier for a classification task, tune key hyperparameters, evaluate performance, and interpret feature importance.

SOURCE CODE :

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report, roc_auc_score, roc_curve
)
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="target")
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.20, random_state=42, stratify=y
)
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
param_grid = {
"n_estimators": [100, 300, 500],
"max_depth": [None, 5, 10, 20],
"min_samples_split": [2, 5, 10],
"min_samples_leaf": [1, 2, 4]
}
grid = GridSearchCV(
estimator=rf,
param_grid=param_grid,
scoring="f1",
cv=5,
n_jobs=-1,
verbose=0
)
grid.fit(X_train, y_train)
print("Best Params (CV):", grid.best_params_)
best_rf = grid.best_estimator_
best_rf.fit(X_train, y_train)
y_pred = best_rf.predict(X_test)
y_prob = best_rf.predict_proba(X_test)[:, 1]
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, zero_division=0)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
auc = roc_auc_score(y_test, y_prob)
cm = confusion_matrix(y_test, y_pred)
print("\n=== Random Forest — Test Metrics ===")
print(f"Accuracy : {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall : {rec:.4f}")
print(f"F1-Score : {f1:.4f}")
print(f"ROC-AUC : {auc:.4f}")
print("\nConfusion Matrix:\n", cm)
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=0))
importances = pd.Series(best_rf.feature_importances_, index=X.columns)
top10 = importances.sort_values(ascending=False).head(10)
plt.figure()
top10[::-1].plot(kind="barh")
plt.xlabel("Importance")
plt.title("Top 10 Feature Importances — Random Forest")
plt.grid(axis="x", alpha=0.3)
```

plt.show()

from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_prob)

plt.figure()

plt.plot(fpr, tpr, label=f"Random Forest (AUC={auc:.3f})")

plt.plot([0, 1], [0, 1], linestyle="--")

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

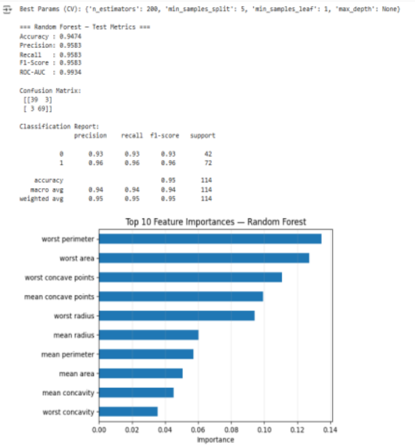
plt.title("ROC Curve — Random Forest")

plt.legend()

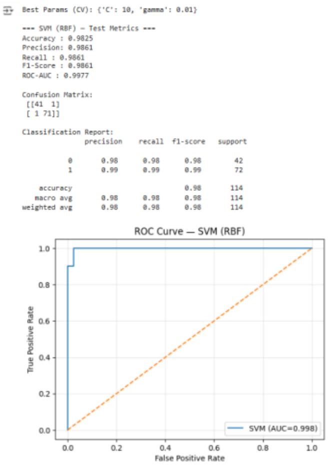
plt.grid(True, alpha=0.3)

plt.show()

OUTPUT:



```
Best Params (CV): {'n_estimators': 200, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': None}

=== Random Forest — Test Metrics ===
Accuracy : 0.9474
Precision: 0.9583
Recall   : 0.9583
F1-Score : 0.9583
ROC-AUC  : 0.9934

Confusion Matrix:
 [[39  3]
 [ 3 69]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        42
           1       0.96      0.96      0.96        72

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```



```
Best Params (CV): {'C': 10, 'gamma': 0.01}

=== SVM (RBF) — Test Metrics ===
Accuracy : 0.9825
Precision: 0.9861
Recall : 0.9861
F1-Score : 0.9861
ROC-AUC : 0.9977

Confusion Matrix:
 [[41  1]
 [ 1 71]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        42
           1       0.99      0.99      0.99        72

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```