```
# ============================================
# Mini Project: Compare 4 algorithms on Wine dataset
# Algorithms: Least Squares, Perceptron, SVM, PCA, Logistic Regression
# ============================================

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score, classification_report

sns.set(style="whitegrid", context="notebook")
random_state = 42
```

```
# -------------------------
# Load dataset
# -------------------------
data = datasets.load_wine()
X = data.data
y = data.target
feature_names = data.feature_names
target_names = data.target_names  # ['class_0', 'class_1', 'class_2']

# train/test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=random_state
)
```

```
# -------------------------
# 1) Manual Least Squares (using first feature only)
# -------------------------
feat_idx = 0
X_single_train = X_train[:, feat_idx]
X_single_test = X_test[:, feat_idx]

X_mean = np.mean(X_single_train)
y_mean = np.mean(y_train.astype(float))
m = np.sum((X_single_train - X_mean) * (y_train - y_mean)) / np.sum((X_single_train - X_mean)**2)
c = y_mean - m * X_mean

y_pred_lr_cont = m * X_single_test + c
# round to nearest class label (0, 1, 2)
y_pred_lr = np.clip(np.round(y_pred_lr_cont), 0, 2).astype(int)

acc_lr = accuracy_score(y_test, y_pred_lr)
report_lr = classification_report(y_test, y_pred_lr, target_names=target_names)
print("=== Manual Least Squares (1 feature) ===")
print(report_lr)
print(f"Accuracy: {acc_lr:.4f}\n")
```

```
=== Manual Least Squares (1 feature) ===
              precision    recall  f1-score   support

     class_0       1.00      0.11      0.20        18
     class_1       0.40      1.00      0.58        21
     class_2       0.00      0.00      0.00        15

    accuracy                           0.43        54
   macro avg       0.47      0.37      0.26        54
weighted avg       0.49      0.43      0.29        54

Accuracy: 0.4259

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# -------------------------
# 2) Perceptron (full features)
```

```
# -------------------------
scaler_full = StandardScaler()
X_train_full = scaler_full.fit_transform(X_train)
X_test_full = scaler_full.transform(X_test)

perceptron_full = Perceptron(random_state=random_state, max_iter=2000, tol=1e-4)
perceptron_full.fit(X_train_full, y_train)
y_pred_perc = perceptron_full.predict(X_test_full)
acc_perc = accuracy_score(y_test, y_pred_perc)
report_perc = classification_report(y_test, y_pred_perc, target_names=target_names)
print("=== Perceptron (full features) ===")
print(report_perc)
print(f"Accuracy: {acc_perc:.4f}\n")
```

```
=== Perceptron (full features) ===
              precision    recall  f1-score   support

     class_0       0.90      1.00      0.95        18
     class_1       1.00      0.90      0.95        21
     class_2       1.00      1.00      1.00        15

    accuracy                           0.96        54
   macro avg       0.97      0.97      0.97        54
weighted avg       0.97      0.96      0.96        54

Accuracy: 0.9630
```

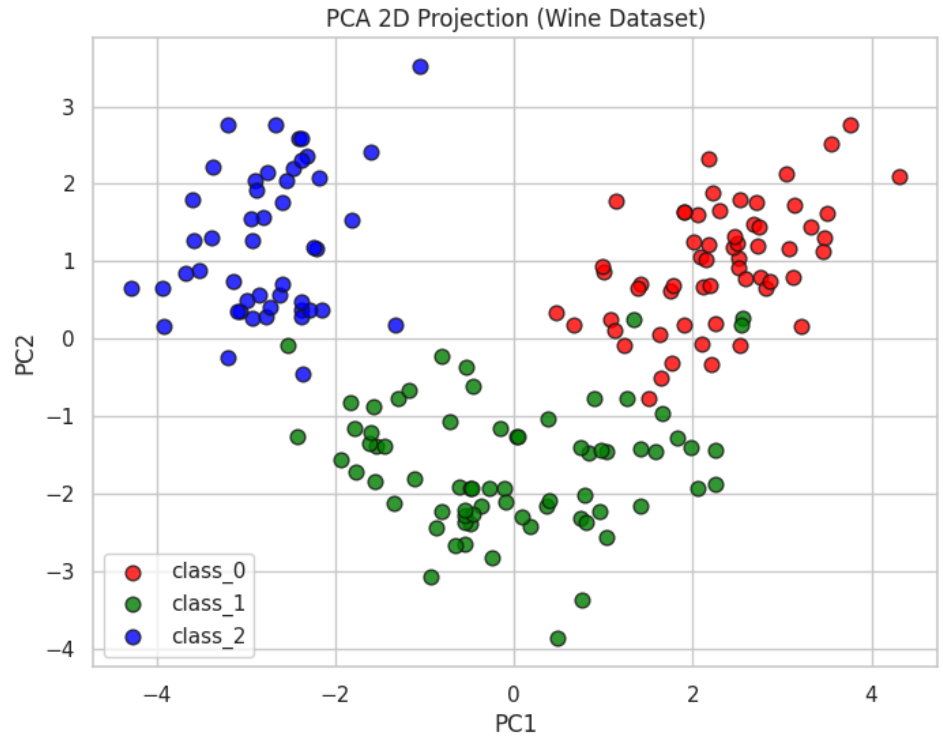
```
# -------------------------
# 3) SVM (GridSearch)
# -------------------------
svm = SVC(kernel='rbf', probability=True, random_state=random_state)
param_grid = {'C': [0.1, 1, 10], 'gamma': ['scale', 0.01, 0.1, 1]}
grid = GridSearchCV(svm, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid.fit(X_train_full, y_train)
best_svm = grid.best_estimator_
y_pred_svm = best_svm.predict(X_test_full)
acc_svm = accuracy_score(y_test, y_pred_svm)
report_svm = classification_report(y_test, y_pred_svm, target_names=target_names)
print("=== SVM (best params) ===")
print("Best params:", grid.best_params_)
print(report_svm)
print(f"Accuracy: {acc_svm:.4f}\n")
```

```
=== SVM (best params) ===
Best params: {'C': 1, 'gamma': 'scale'}
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        18
     class_1       0.95      1.00      0.98        21
     class_2       1.00      0.93      0.97        15

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54

Accuracy: 0.9815
```

```
# -------------------------
# 4) PCA (2D visualization + Perceptron)
# -------------------------
scaler_pca = StandardScaler()
X_scaled = scaler_pca.fit_transform(X)
pca2 = PCA(n_components=2, random_state=random_state)
X_pca_2d = pca2.fit_transform(X_scaled)

print("PCA(2) explained variance ratio:", pca2.explained_variance_ratio_,
      "sum:", pca2.explained_variance_ratio_.sum())

# PCA scatter plot
plt.figure(figsize=(8,6))
colors = ['red', 'green', 'blue']
for i, color, label in zip(range(3), colors, target_names):
    plt.scatter(X_pca_2d[y == i, 0], X_pca_2d[y == i, 1],
                color=color, label=label, s=60, alpha=0.8, edgecolors='k')
plt.title("PCA 2D Projection (Wine Dataset)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.legend()
```

```
plt.grid(True)
plt.show()
```

```
PCA(2) explained variance ratio: [0.36198848 0.1920749 ] sum: 0.5540633835693526
```




```
# Train Perceptron on PCA(2) features
X_pca_train, X_pca_test, y_pca_train, y_pca_test = train_test_split(
    X_pca_2d, y, test_size=0.30, stratify=y, random_state=random_state
)
perc_pca = Perceptron(random_state=random_state, max_iter=2000, tol=1e-4)
perc_pca.fit(X_pca_train, y_pca_train)
y_pred_pca = perc_pca.predict(X_pca_test)
acc_pca = accuracy_score(y_pca_test, y_pred_pca)
report_pca = classification_report(y_pca_test, y_pred_pca, target_names=target_names)
print("=== Perceptron on PCA(2) ===")
print(report_pca)
print(f"Accuracy: {acc_pca:.4f}\n")
```

```
=== Perceptron on PCA(2) ===
              precision    recall  f1-score   support

     class_0       0.94      0.89      0.91        18
     class_1       0.91      0.95      0.93        21
     class_2       1.00      1.00      1.00        15

    accuracy                           0.94        54
   macro avg       0.95      0.95      0.95        54
weighted avg       0.95      0.94      0.94        54

Accuracy: 0.9444
```

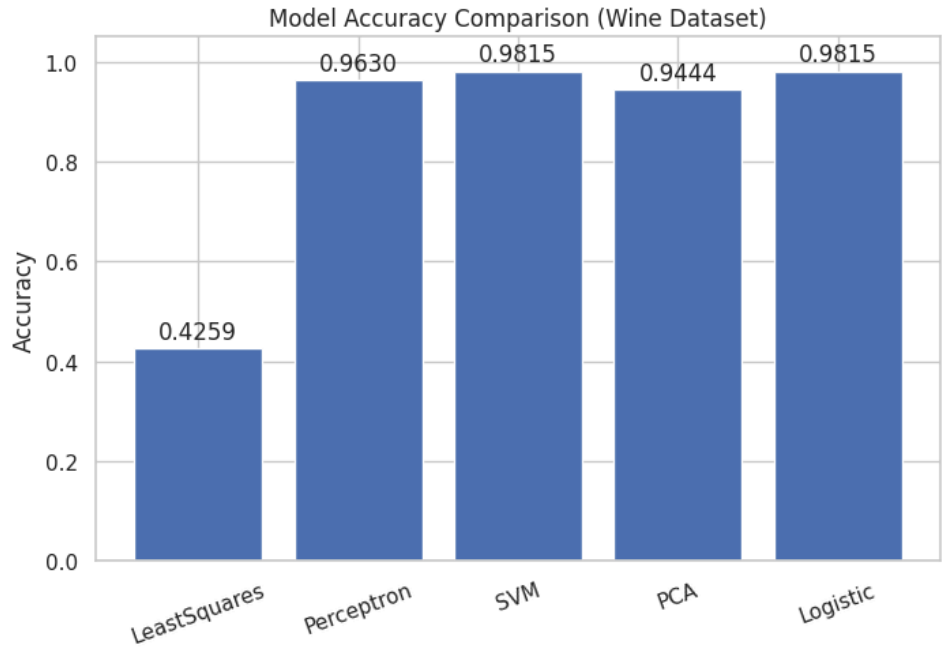
```
# -------------------------
# 5) Logistic Regression
# -------------------------
logreg = LogisticRegression(max_iter=5000, random_state=random_state)
logreg.fit(X_train_full, y_train)
y_pred_log = logreg.predict(X_test_full)
acc_log = accuracy_score(y_test, y_pred_log)
report_log = classification_report(y_test, y_pred_log, target_names=target_names)
print("=== Logistic Regression ===")
print(report_log)
print(f"Accuracy: {acc_log:.4f}\n")
```

```
=== Logistic Regression ===
              precision    recall  f1-score   support

     class_0       0.95      1.00      0.97        18
     class_1       1.00      0.95      0.98        21
     class_2       1.00      1.00      1.00        15
```

```
    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54

Accuracy: 0.9815
```

```
# -------------------------
# Summary comparison
# -------------------------
models = ['LeastSquares', 'Perceptron', 'SVM', 'PCA', 'Logistic']
accuracies = [acc_lr, acc_perc, acc_svm, acc_pca, acc_log]

summary_df = pd.DataFrame({'Model': models, 'Accuracy': accuracies})
print("\n=== Accuracy Summary ===")
print(summary_df.to_string(index=False))

plt.figure(figsize=(8,5))
bars = plt.bar(summary_df['Model'], summary_df['Accuracy'])
plt.ylim(0,1.05)
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison (Wine Dataset)')
for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width()/2, acc + 0.02, f"{acc:.4f}", ha='center')
plt.xticks(rotation=20)
plt.show()
```

```
=== Accuracy Summary ===
       Model  Accuracy
LeastSquares  0.425926
  Perceptron  0.962963
         SVM  0.981481
         PCA  0.944444
    Logistic  0.981481
```



Start coding or generate with AI.