ITEC 6720 - Final Project: Screenshots Log

Iris Step-by-Step Project

End-to-End (iris_end_to_end_SVM_model.py): Code+output

This project walks through a complete, end-to-end machine learning workflow on the classic Iris flower dataset to predict species from four measurements (sepal length/width, petal length/width). After loading the data and exploring it with simple plots (boxplots, histograms, scatter matrix), I created a stratified train/test split and compared several baseline algorithms using 10-fold cross-validation (LR, LDA, KNN, CART, Naive Bayes, SVM). I then tuned an SVM pipeline with grid search over kernel, C, and gamma, selected the best configuration (linear kernel, C=0.1, gamma="scale"), and evaluated it on the held-out test set. The tuned SVM achieved strong, balanced performance (≈0.92 test accuracy) with only a few confusions between Versicolor and Virginica. Finally, I saved the trained model to disk and verified it by reloading and re-scoring it for a reproducible result.

1)Script and libraries used.

scripts > step_by_step > iris_end_to_end_SVM_model.py > ...

```python
# scripts/step_by_step/iris_end_to_end.py
"""
Iris End-to-End Project (for Final Project submission)

Outputs created:
- figures/eda_boxplots.png
- figures/eda_hist.png
- figures/eda_scatter_matrix.png
- figures/algo_boxplot.png
- figures/cv_vs_test.png
- figures/confusion_matrix.png
- reports/spotcheck_cv.csv
- reports/step_by_step_run_summary.txt
- models/iris_best_model.joblib
"""

from pathlib import Path
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from pandas import read_csv

from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    classification_report, confusion_matrix, ConfusionMatrixDisplay, accuracy_score
)
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from joblib import dump, load

```

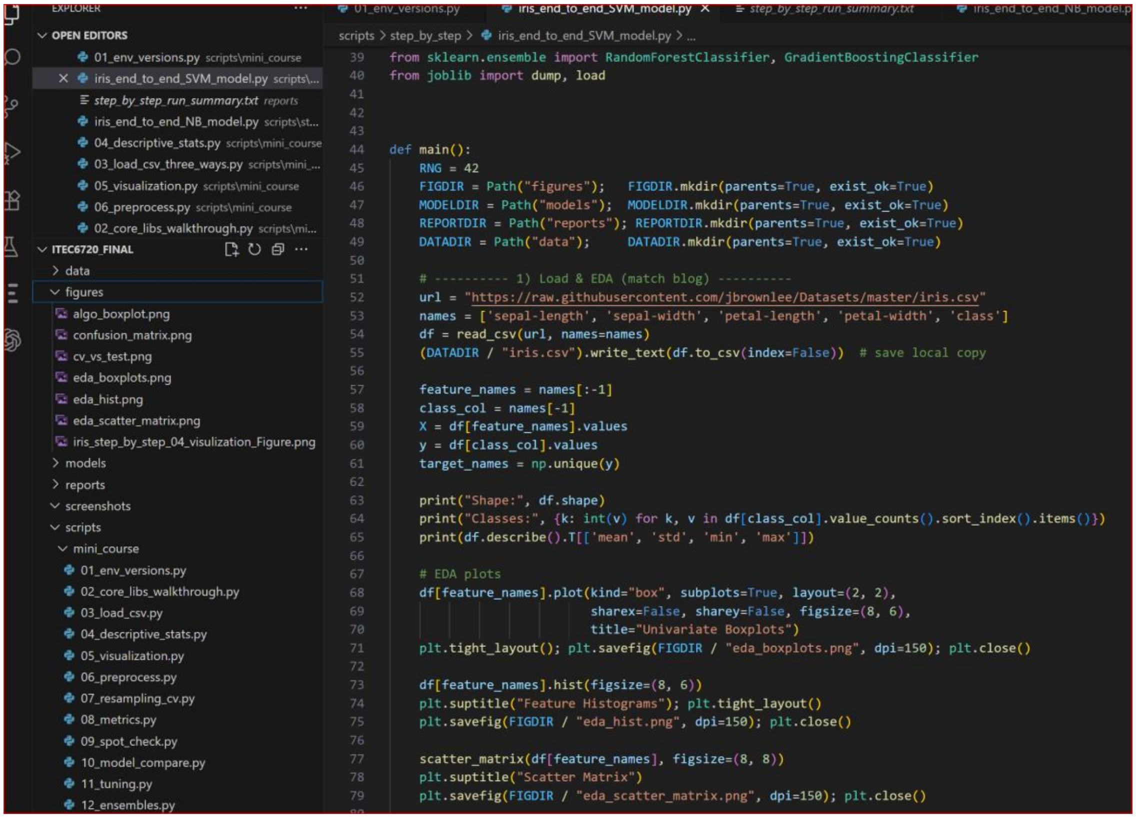
2)Loaded Iris dataset from URL and defined features/labels.

```python
def main():
    RNG = 42
    FIGDIR = Path("figures");   FIGDIR.mkdir(parents=True, exist_ok=True)
    MODELDIR = Path("models");  MODELDIR.mkdir(parents=True, exist_ok=True)
    REPORTDIR = Path("reports"); REPORTDIR.mkdir(parents=True, exist_ok=True)
    DATADIR = Path("data");     DATADIR.mkdir(parents=True, exist_ok=True)

    # ---------- 1) Load & EDA (match blog) ----------
    url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
    names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
    df = read_csv(url, names=names)
    (DATADIR / "iris.csv").write_text(df.to_csv(index=False))  # save local copy

    feature_names = names[:-1]
    class_col = names[-1]
    X = df[feature_names].values
    y = df[class_col].values
    target_names = np.unique(y)

    print("Shape:", df.shape)
    print("Classes:", {k: int(v) for k, v in df[class_col].value_counts().sort_index().items()})
    print(df.describe().T[['mean', 'std', 'min', 'max']])
```

Output

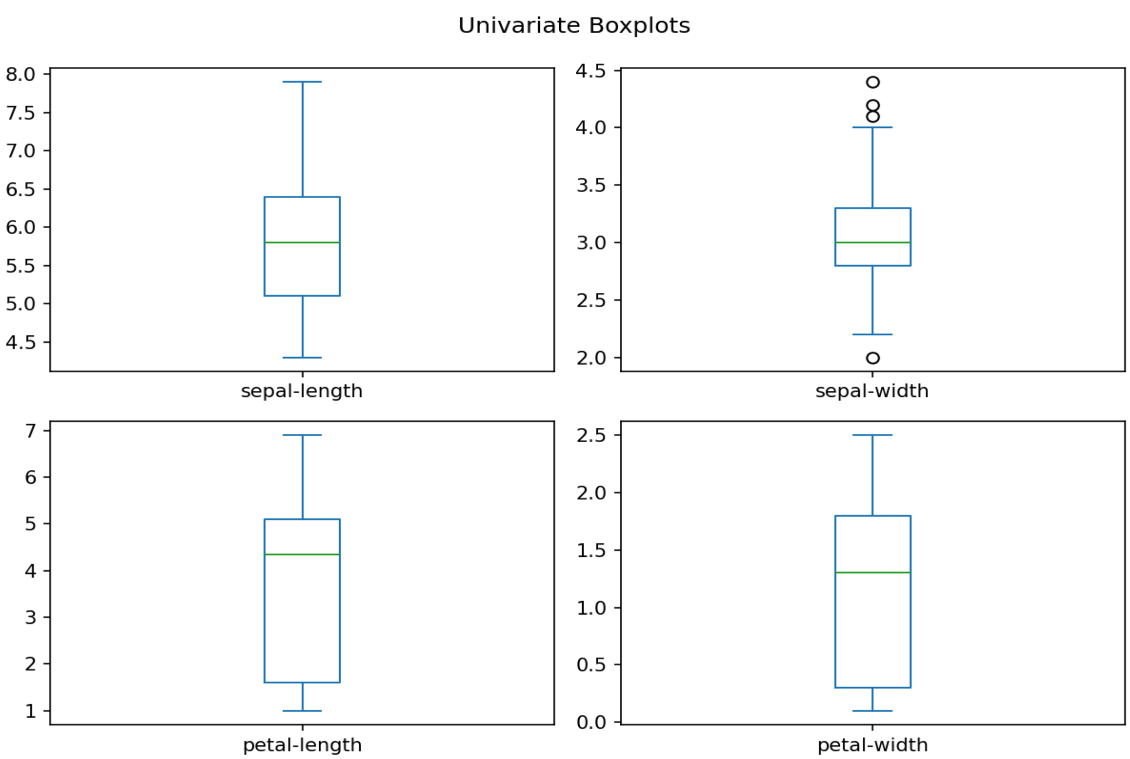
```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts/step_by_step/iris_end_to_end_SVM_model.py
Shape: (150, 5)
Classes: {'Iris-setosa': 50, 'Iris-versicolor': 50, 'Iris-virginica': 50}
                  mean       std  min  max
sepal-length  5.843333  0.828066  4.3  7.9
sepal-width   3.054000  0.433594  2.0  4.4
petal-length  3.758667  1.764420  1.0  6.9
petal-width   1.198667  0.763161  0.1  2.5
```

3)EDA: boxplots, histograms, scatter matrix generated.

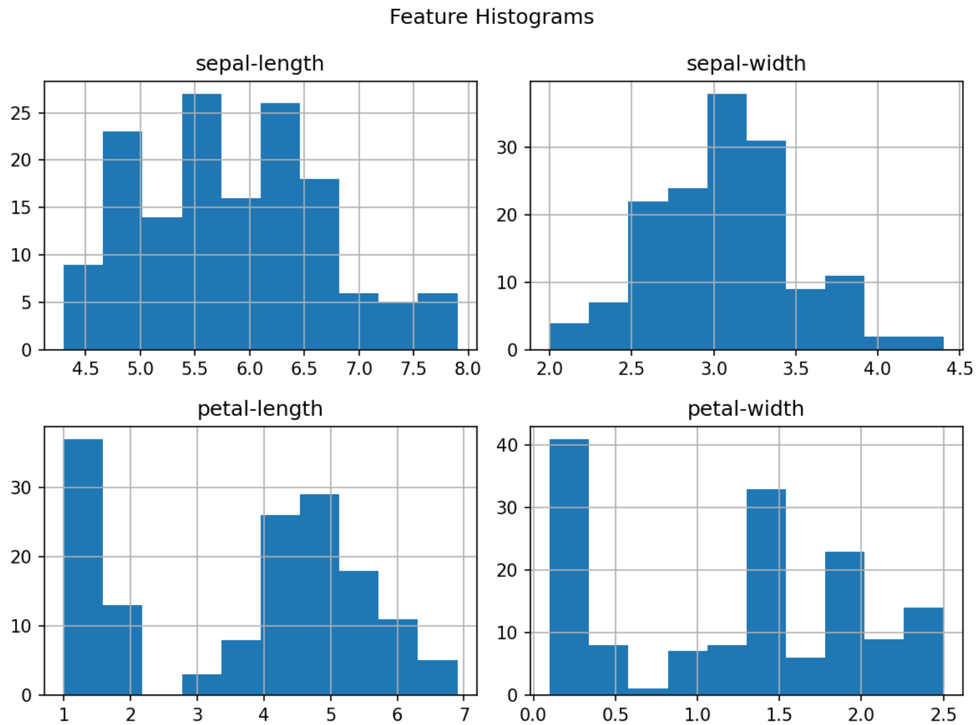


4) EDA figures (3 images)

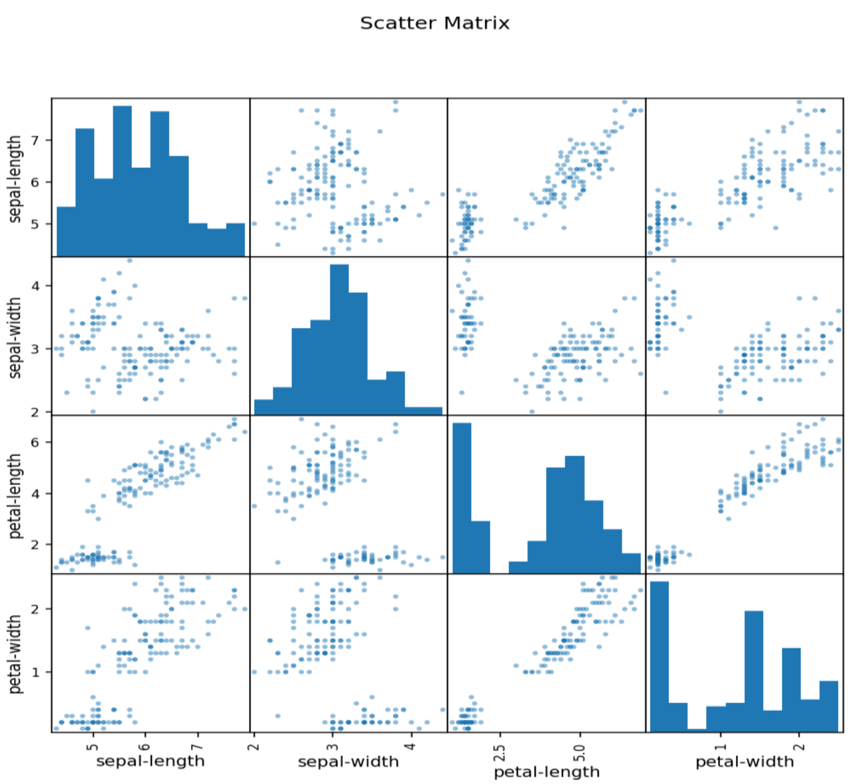
figures/eda_boxplots.png



figures/eda_hist.png



figures/eda_scatter_matrix.png



5)Stratified split; 10-fold CV; candidate models (LR, LDA, KNN, CART, NB, SVM).

```python
# ---------- 2) Split (stratified) ----------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, stratify=y, random_state=RNG
)

# ---------- 3) CV harness + spot-check ----------
cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=RNG)
models = [
    ("LR",  Pipeline([("scaler", StandardScaler()),
                      ("clf", LogisticRegression(max_iter=2000, solver="lbfgs", multi_class="auto"))])),
    ("LDA", LinearDiscriminantAnalysis()),
    ("KNN", Pipeline([("scaler", StandardScaler()), ("clf", KNeighborsClassifier())])),
    ("CART", DecisionTreeClassifier(random_state=RNG)),
    ("NB",  GaussianNB()),
    ("SVM", Pipeline([("scaler", StandardScaler()), ("clf", SVC())])),
]
```

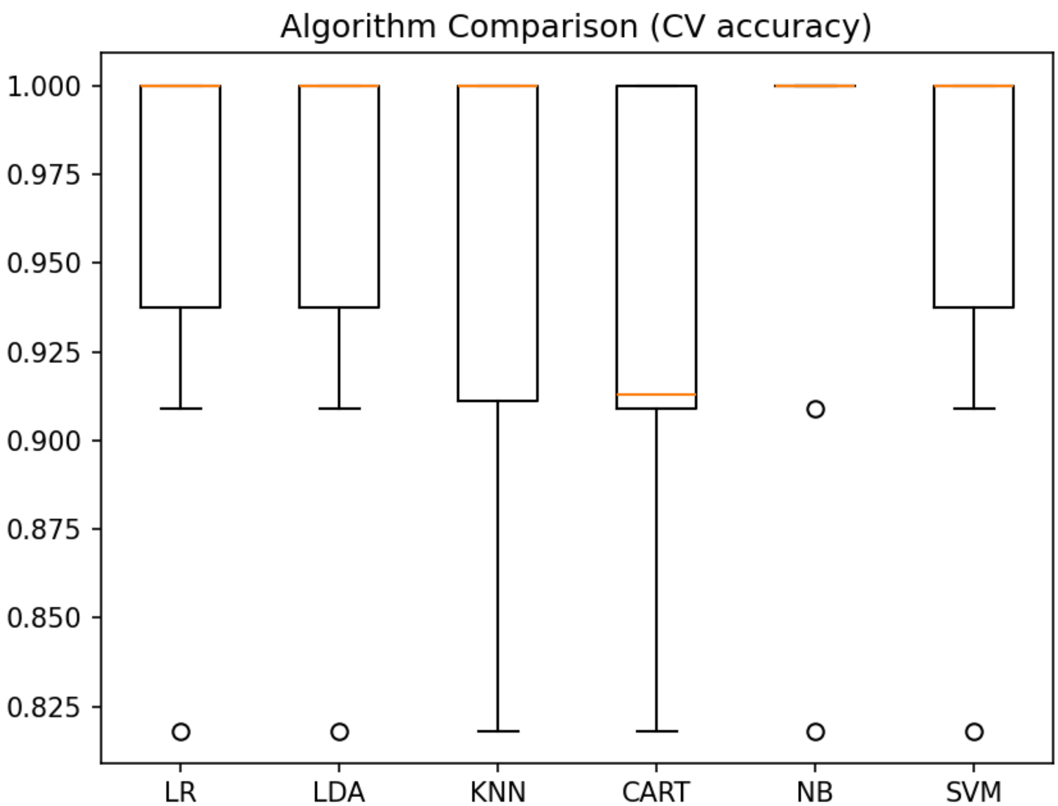
6) Spot-check CV results.

```python
print("\nSpot-check (10-fold CV accuracy):")
model_names, results = [], []
for name, model in models:
    scores = cross_val_score(model, X_train, y_train, cv=cv, scoring="accuracy")
    model_names.append(name); results.append(scores)
    print(f"{name:>4}: {scores.mean():.3f} (+/- {scores.std():.3f})")

pd.DataFrame({n: r for n, r in zip(model_names, results)}).to_csv(REPORTDIR / "spotcheck_cv.csv", index=False)

plt.figure()
plt.boxplot(results, tick_labels=model_names)
plt.title("Algorithm Comparison (CV accuracy)")
plt.savefig(FIGDIR / "algo_boxplot.png", dpi=150); plt.close()
```

Output Model comparison via 10-fold CV accuracy

```
Spot-check (10-fold CV accuracy):
  LR: 0.964 (+/- 0.060)
 LDA: 0.964 (+/- 0.060)
 KNN: 0.955 (+/- 0.060)
CART: 0.928 (+/- 0.068)
  NB: 0.973 (+/- 0.058)
 SVM: 0.964 (+/- 0.060)
```

7)Algorithm comparison boxplot



8)Grid search section (SVM)

```python
# ---------- 4) Hyperparameter tuning (SVM pipeline) ----------
svm_pipe = Pipeline([("scaler", StandardScaler()), ("clf", SVC())])
param_grid = {
    "clf__kernel": ["rbf", "linear"],
    "clf__C": [0.1, 1, 10, 100],
    "clf__gamma": ["scale", "auto", 0.1, 0.01, 0.001],
}
grid = GridSearchCV(svm_pipe, param_grid, scoring="accuracy", cv=cv, n_jobs=-1, return_train_score=True)
grid.fit(X_train, y_train)
print("\nBest SVM params:", grid.best_params_)
print("Best CV accuracy:", round(grid.best_score_, 3))
```

9)Console output: best SVM params + best CV

```
Best SVM params: {'clf__C': 0.1, 'clf__gamma': 'scale', 'clf__kernel': 'linear'}
Best CV accuracy: 0.973
```

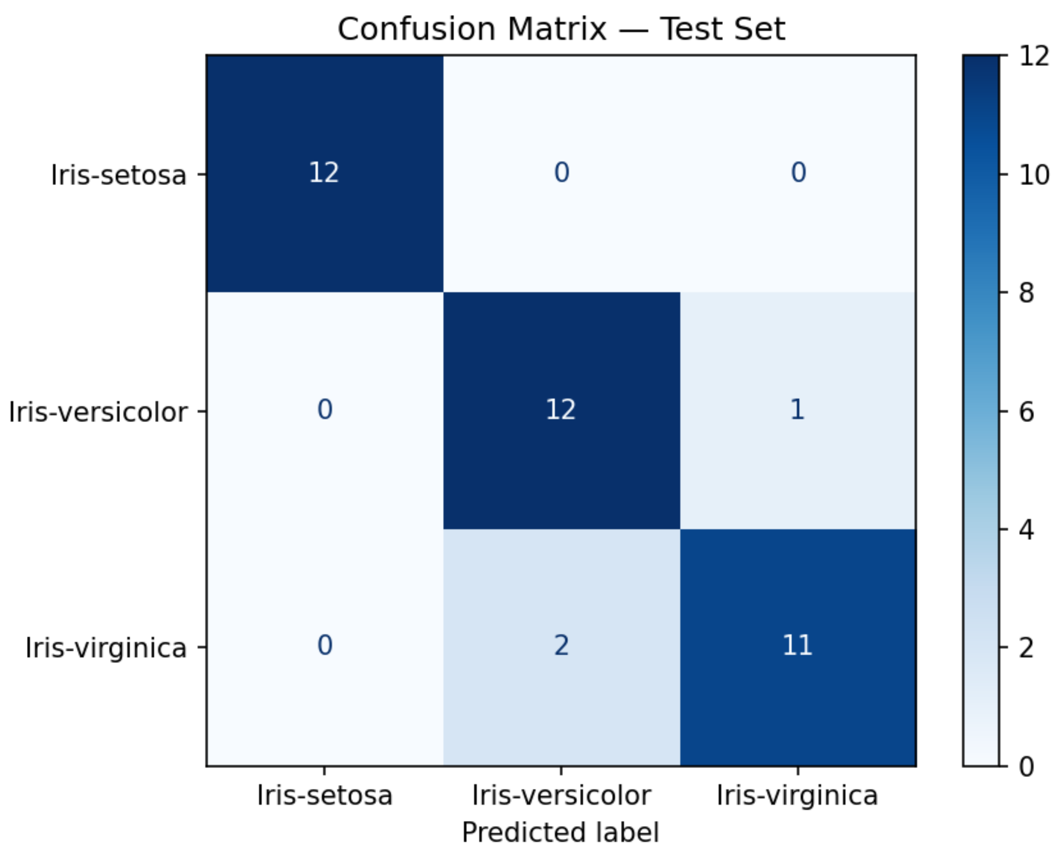
10) Hold-out test accuracy and class-level metrics (confusion matrix and classification report).

```python
# ---------- 6) Finalize & evaluate on test ----------
best_model = grid.best_estimator_
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

test_acc = accuracy_score(y_test, y_pred)
print("\nTEST accuracy:", test_acc)
print("Confusion matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification report:\n", classification_report(y_test, y_pred, target_names=target_names))
```

Output

```
TEST accuracy: 0.9210526315789473
Confusion matrix:
 [[12  0  0]
 [ 0 12  1]
 [ 0  2 11]]
Classification report:
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       0.86      0.92      0.89        13
 Iris-virginica       0.92      0.85      0.88        13

       accuracy                           0.92        38
      macro avg       0.92      0.92      0.92        38
   weighted avg       0.92      0.92      0.92        38
```

11)Confusion matrix on test set.



12)Saved and reloaded best model.

```
# ---------- 7) Persist & reload ----------
model_path = MODELDIR / "iris_best_model.joblib"
dump(best_model, model_path)
reloaded = load(model_path)
print("Loaded model test accuracy (sanity check):", accuracy_score(y_test, reloaded.predict(X_test)))
```

Output

```
Loaded model test accuracy (sanity check): 0.9210526315789473
```

13) CV mean vs. test accuracy for the selected model, (2) test set confusion matrix, and (3) a brief run summary saved to reports/step_by_step_run_summary.txt.”

```python
    # ---------- 8) Result figures ----------
    plt.figure()
    plt.title("CV Mean vs Test Accuracy (Best Model)")
    plt.bar(["CV mean", "Test"], [grid.best_score_, test_acc])
    plt.ylim(0.8, 1.0)
    plt.savefig(FIGDIR / "cv_vs_test.png", dpi=150); plt.close()

    ConfusionMatrixDisplay.from_predictions(y_test, y_pred, display_labels=target_names, cmap="Blues", values_format="d")
    plt.title("Confusion Matrix – Test Set")
    plt.savefig(FIGDIR / "confusion_matrix.png", dpi=150); plt.close()

    with open(REPORTDIR / "step_by_step_run_summary.txt", "w", encoding="utf-8") as f:
        f.write("Iris Step-by-Step Project – Run Summary\n")
        f.write(f"Best CV accuracy: {grid.best_score_:.3f}\n")
        f.write(f"Best params: {grid.best_params_}\n")
        f.write(f"Test accuracy: {test_acc:.3f}\n")


if __name__ == "__main__":
    main()
```

Conclusion: I chose SVM because it achieved the best cross-validated accuracy after tuning and delivered about 0.92 test accuracy; the confusion matrix shows strong overall performance with only a few Versicolor ↔ Virginica mix-ups.