ITEC 6720 - Final Project: Screenshots Log

1. Mini-Course (MC-01 to MC-14)
2. Iris Step-by-Step Project

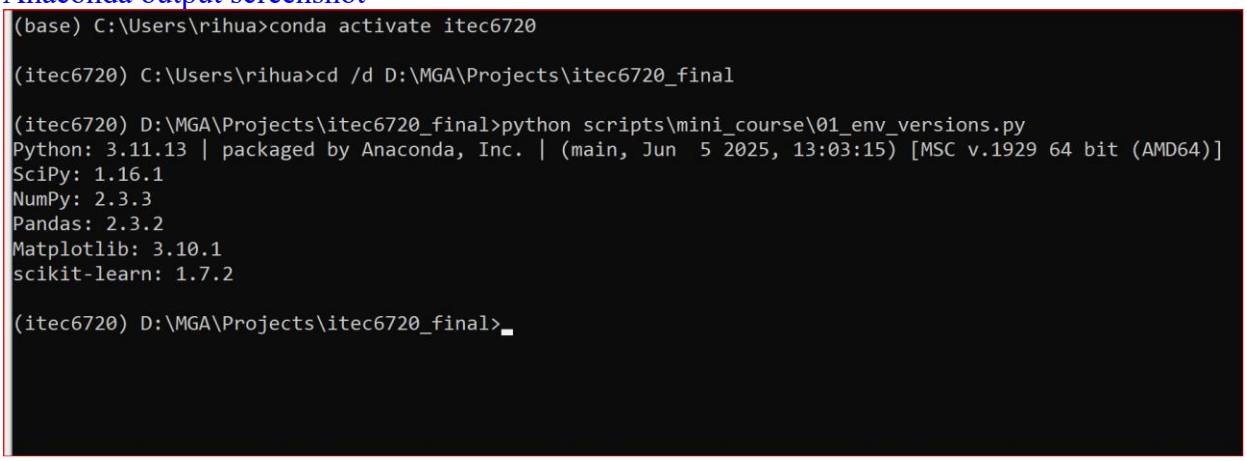
1) Mini-Course

MC-01 -- Environment Check

Caption: MC-01 -- Python, SciPy, NumPy, Pandas, Matplotlib, scikit-learn versions printed in the (itec6720) environment.

VS code screenshot

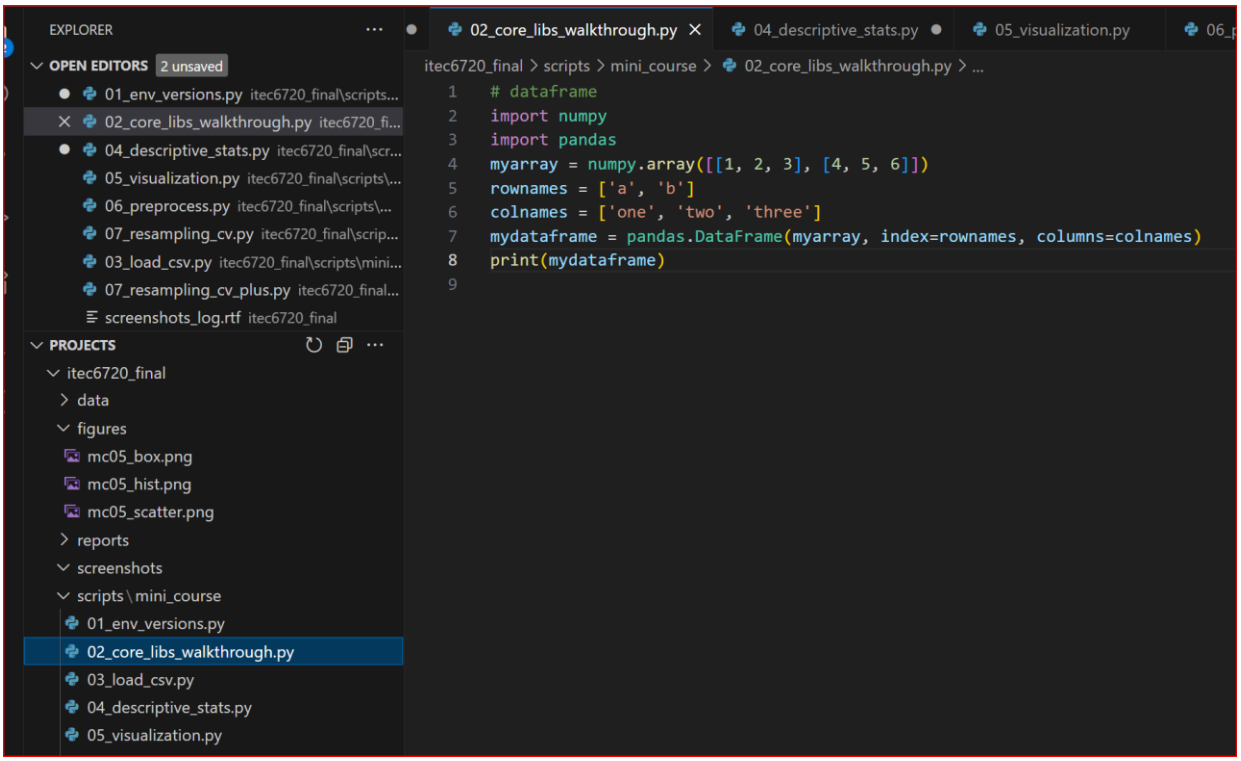
Anaconda output screenshot



MC-02 -- Core Libs Tour

Caption: MC-02 — Core libs tour: created a small NumPy array and printed a labeled 2×3 Pandas DataFrame (rows a,b; columns one,two,three).

Code



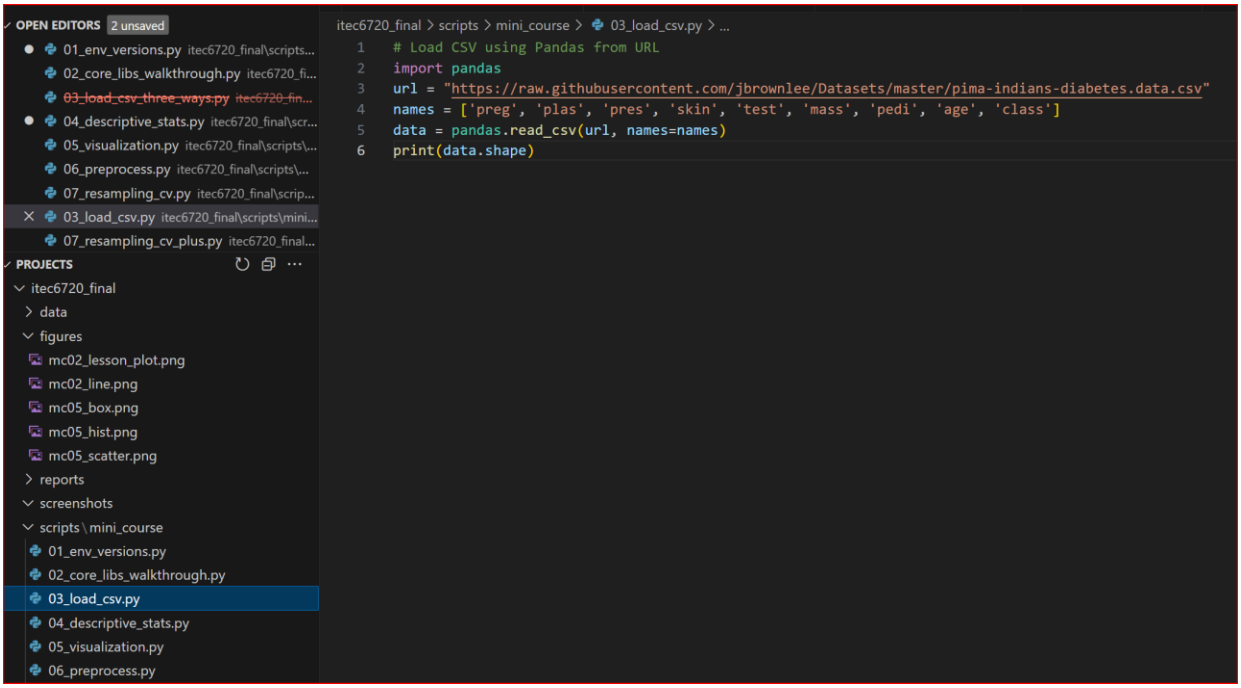
Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\02_core_libs_walkthrough.py
   one  two  three
a    1    2      3
b    4    5      6
```

MC-03 -- Load CSV

Caption: MC-03 — Loaded the Pima Indians Diabetes CSV from the URL with pandas.read_csv (using column names) and verified the shape (768, 9).

Code



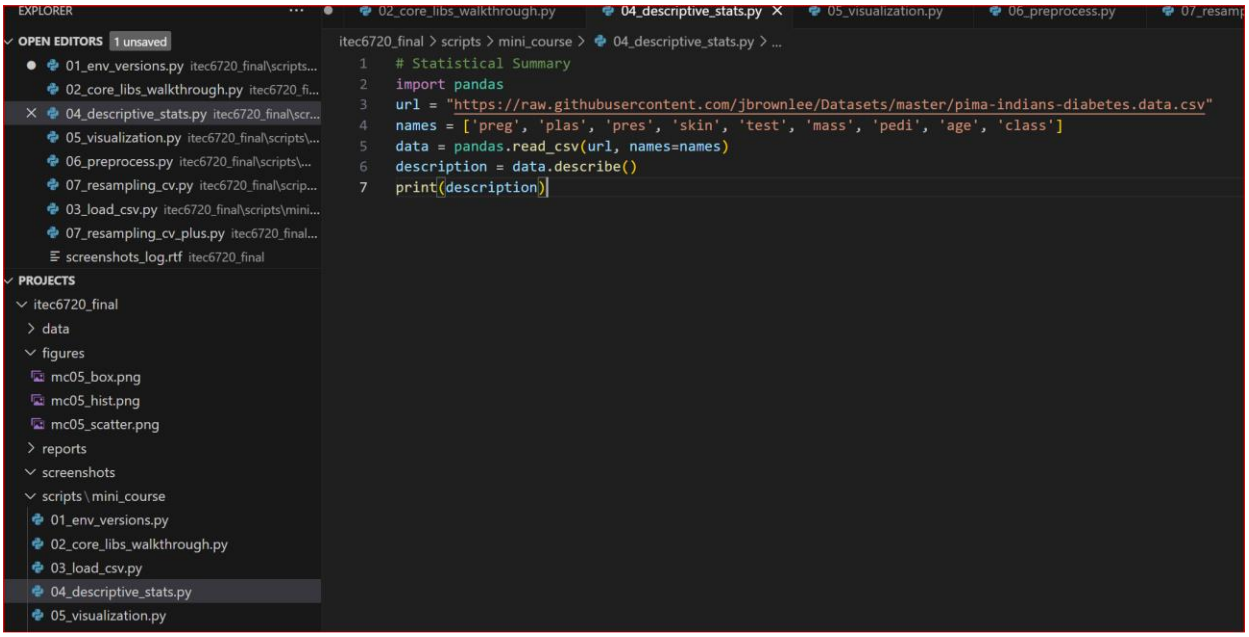
Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\03_load_csv.py
(768, 9)
```

MC-04 -- Descriptive Statistics

Caption: loaded the Pima Indians Diabetes dataset from the URL with pandas.read_csv and printed the dataset's summary using data.describe().

Code



Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\04_descriptive_stats.py
             preg        plas        pres        skin        test        mass        pedi         age       class
count  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000
mean     3.845052  120.894531   69.105469   20.536458   79.799479   31.992578    0.471876   33.240885    0.348958
std      3.369578   31.972618   19.355807   15.952218  115.244002    7.884160    0.331329   11.760232    0.476951
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.078000   21.000000    0.000000
25%      1.000000   99.000000   62.000000    0.000000    0.000000   27.300000    0.243750   24.000000    0.000000
50%      3.000000  117.000000   72.000000   23.000000   30.500000   32.000000    0.372500   29.000000    0.000000
75%      6.000000  140.250000   80.000000   32.000000  127.250000   36.600000    0.626250   41.000000    1.000000
max     17.000000  199.000000  122.000000   99.000000  846.000000   67.100000    2.420000   81.000000    1.000000
```

MC-05 -- Visualization

Caption: MC-05 —loaded the Pima dataset and produced a **scatter-plot matrix** of all attributes using pandas.plotting.scatter_matrix (displayed with Matplotlib).

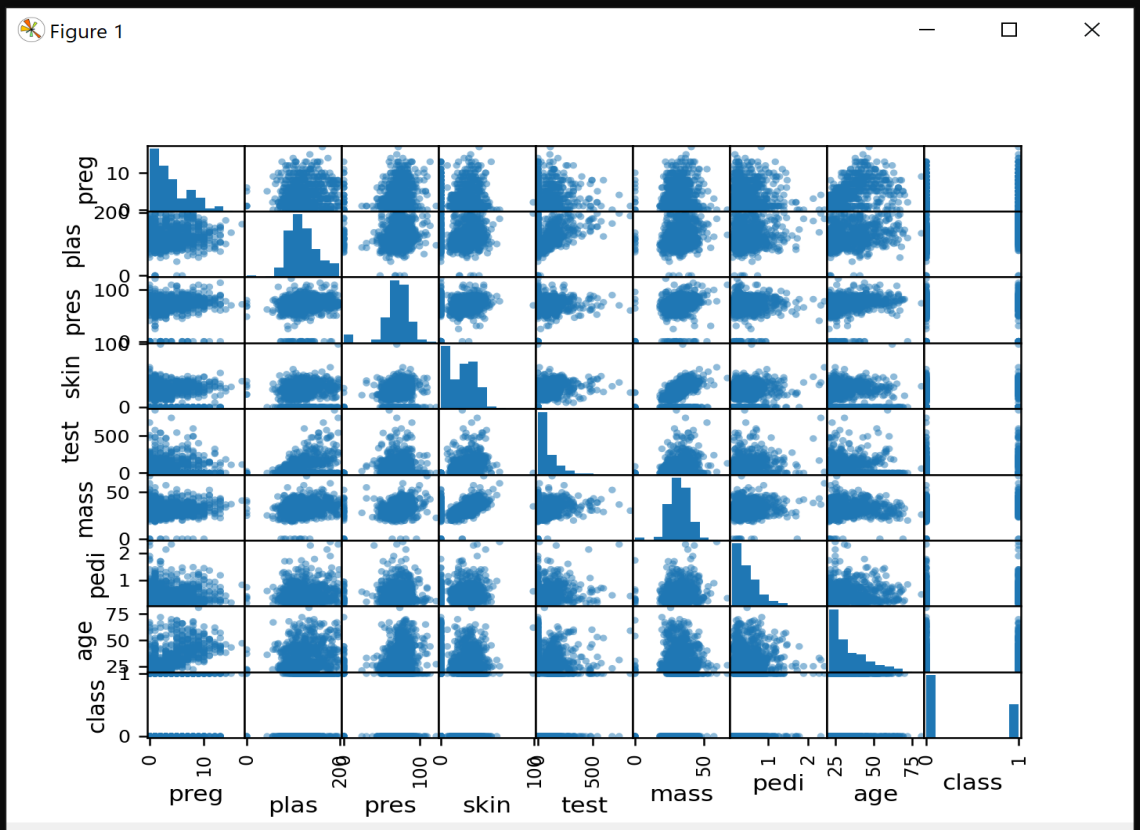
Code

itec6720_final > scripts > mini_course > 05_visualization.py > ...

```python
# Scatter Plot Matrix
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pandas.read_csv(url, names=names)
scatter_matrix(data)
plt.show()
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\05_visualization.py
```



MC-06 -- Preprocessing

Caption:MC-06 —standardized the Pima features X with scikit-learn's StandardScaler (mean 0, std 1) and printed the first 5 rows of the rescaled data.

Code

itec6720_final > scripts > mini_course > 06_preprocess.py > ...

```python
# Standardize data (0 mean, 1 stdev)
from sklearn.preprocessing import StandardScaler
import pandas
import numpy
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
# separate array into input and output components
X = array[:,0:8]
Y = array[:,8]
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
# summarize transformed data
numpy.set_printoptions(precision=3)
print(rescaledX[0:5,:])
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\06_preprocess.py
[[ 0.64   0.848  0.15   0.907 -0.693  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161  0.531 -0.693 -0.684 -0.365 -0.191]
 [ 1.234  1.944 -0.264 -1.288 -0.693 -1.103  0.604 -0.106]
 [-0.845 -0.998 -0.161  0.155  0.123 -0.494 -0.921 -1.042]
 [-1.142  0.504 -1.505  0.907  0.766  1.41   5.485 -0.02 ]]
```

MC-07 -- Resampling

Caption: MC-07 — evaluated Logistic Regression on the Pima dataset using 10-fold cross-validation and reported mean accuracy ± standard deviation.

Code

itec6720_final > scripts > mini_course > 07_resampling_cv.py > ...

```python
# MC-07 (basic): Evaluate using 10-fold CV (matches the mini-course)
from pandas import read_csv
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg','plas','pres','skin','test','mass','pedi','age','class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X, y = array[:, 0:8], array[:, 8]

kfold = KFold(n_splits=10, random_state=7, shuffle=True)
model = LogisticRegression(solver="liblinear", max_iter=1000, random_state=42)
results = cross_val_score(model, X, y, cv=kfold, scoring="accuracy")

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\07_resampling_cv.py
Accuracy: 77.086% (5.091%)
```

MC-08 -- Metrics

Caption: MC-08 — Accuracy via 10-fold CV and LogLoss (lower is better) for Logistic Regression on the Pima dataset; KFold uses shuffle=True, random_state=7.

Code

itec6720_final > scripts > mini_course > 08_metrics.py > ...

```python
# MC-08: Algorithm evaluation metrics (accuracy + log loss)
from pandas import read_csv
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LogisticRegression
import numpy as np

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

df = read_csv(url, names=names)
X = df.iloc[:, 0:8].values
y = df.iloc[:, 8].values

# FIX: enable shuffling if you want to use random_state
kfold = KFold(n_splits=10, shuffle=True, random_state=7)

model = LogisticRegression(solver='liblinear', max_iter=1000)

# Accuracy (higher is better)
acc = cross_val_score(model, X, y, cv=kfold, scoring="accuracy")
print("Accuracy: %.3f%% (%.3f%%)" % (acc.mean() * 100.0, acc.std() * 100.0))

# Log loss (lower is better) -> cross_val_score returns *negative* log loss
neg_ll = cross_val_score(model, X, y, cv=kfold, scoring="neg_log_loss")
print("LogLoss:  %.3f (±%.3f)" % (-neg_ll.mean(), neg_ll.std()))

```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\08_metrics.py
Accuracy: 77.086% (5.091%)
LogLoss:  0.494 (±0.042)
```

MC-09 -- Spot-Check Algorithms

Caption: MC-09 — Spot-check: KNN regressor on Boston Housing with 10-fold CV; report mean ± std RMSE.

Code

itec6720_final > scripts > mini_course > 09_spot_check.py > ...

```python
from pandas import read_csv
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsRegressor
import numpy as np

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.data"
names = ['CRIM','ZN','INDUS','CHAS','NOX','RM','AGE','DIS','RAD','TAX','PTRATIO','B','LSTAT','MEDV']
df = read_csv(url, sep=r"\s+", names=names)

X, y = df.iloc[:, :13].values, df.iloc[:, 13].values
cv = KFold(n_splits=10, shuffle=True, random_state=7)
model = KNeighborsRegressor()

scores = cross_val_score(model, X, y, cv=cv, scoring='neg_mean_squared_error')
rmse = np.sqrt(-scores)
print(f"RMSE: {rmse.mean():.3f} ({rmse.std():.3f})")

```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\09_spot_check.py
RMSE: 6.130 (1.131)
```

MC-10 -- Model Comparison

Caption: MC-10 -- Comparing Logistic Regression vs. LDA on the Pima Diabetes dataset using 10-fold cross-validation and reporting mean ± std accuracy to choose the more reliable model.

Code

itec6720_final > scripts > mini_course > 10_model_compare.py > ...

```python
# Compare Algorithms
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
# load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
# prepare models
models = []
models.append(('LR', LogisticRegression(solver='liblinear')))
models.append(('LDA', LinearDiscriminantAnalysis()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:

    cv = KFold(n_splits=10, shuffle=True, random_state=7)
    cv_results = cross_val_score(model, X, Y, cv=cv, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\10_model_compare.py
LR: 0.770865 (0.050905)
LDA: 0.766969 (0.047966)
```

MC-11 -- Tuning

Caption: Tuning model hyperparameters with GridSearchCV: try a set of alpha values for Ridge using cross-validation, report the best CV score, and select the alpha that generalizes best.

Code

```python

# Grid Search for Algorithm Tuning
from pandas import read_csv
import numpy
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
alphas = numpy.array([1,0.1,0.01,0.001,0.0001,0])
param_grid = dict(alpha=alphas)
model = Ridge()
grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3)
grid.fit(X, Y)
print(grid.best_score_)
print(grid.best_estimator_.alpha)

```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\11_tuning.py
0.27961755931297233
1.0
```

MC-12 -- Ensembles

Caption: MC-12 -- Evaluate a 100-tree RandomForestClassifier on the Pima Diabetes dataset using shuffled 10-fold cross-validation (max_features=3) and print the mean CV accuracy.

Code

itec6720_final > scripts > mini_course > 12_ensembles.py > ...

```python
# Random Forest Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
num_trees = 100
max_features = 3
kfold = KFold(n_splits=10, shuffle=True, random_state=7)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\12_ensembles.py
0.7695488721804511
```

MC-13 -- Finalize and Save

Caption: Hold out 33% for testing, train Logistic Regression on the 67% training split, pickle the fitted model to disk, reload it later, and report test accuracy—your "finalized" model workflow.

Code

itec6720_final > scripts > mini_course > 13_finalize_save.py > ...

```python
# Save Model Using Pickle
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pickle
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=test_size, random_state=seed)
# Fit the model on 67%
model = LogisticRegression(solver='liblinear')
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
loaded_model = pickle.load(open(filename, 'rb'))
result = loaded_model.score(X_test, Y_test)
print(result)
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\13_finalize_save.py
0.7559055118110236
```

MC-14 – Hello World End-to-End Project

1.Understand data (EDA)

Caption: "L14 – EDA code: load iris, print shape/classes/summary."

Code

itec6720_final > scripts > mini_course > 14_hello_world_project.py > ...

```python
from sklearn.datasets import load_iris
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 accuracy_score, classification_report, confusion_matrix
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

# ----------------------------
# 1) Load & EDA (minimal)
# ----------------------------
iris = load_iris(as_frame=True)
df = iris.frame  # features + target
df['target'] = iris.target
print("Shape:", df.shape)
print("Classes:", df['target'].value_counts().sort_index().to_dict())
print(df.describe().T[['mean','std','min','max']])
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\mini_course\14_hello_world_project.py
Shape: (150, 5)
Classes: {0: 50, 1: 50, 2: 50}
                     mean       std  min  max
sepal length (cm)  5.843333  0.828066  4.3  7.9
sepal width (cm)   3.057333  0.435866  2.0  4.4
petal length (cm)  3.758000  1.765298  1.0  6.9
petal width (cm)   1.199333  0.762238  0.1  2.5
target             1.000000  0.819232  0.0  2.0
```

2.Preprocess / Train–test split

Caption: "L14 – Split code: stratified 75/25 train/test."

```
# ----------------------------
# 2) Train/Test split
# ----------------------------
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, stratify=y, random_state=7
)

print("Train/Test shapes:", X_train.shape, X_test.shape)
print("Train class counts:", np.bincount(y_train))
print("Test class counts:",  np.bincount(y_test))
```

Output L14 – Split output: shapes and class balance

```
Train/Test shapes: (112, 4) (38, 4)
Train class counts: [37 37 38]
Test class counts: [13 13 12]
```

3.Spot-check algorithms (10-fold CV)

Caption: "L14 – Spot-check code: LR, LDA, KNN, CART, NB, SVM with 10-fold CV."

```
# ----------------------------
# 3) Spot-check algorithms
#    (scale where it matters via Pipeline)
# ----------------------------
candidates = [

    ("LR", Pipeline([
    ('scaler', StandardScaler()),
    ('clf', LogisticRegression(solver='lbfgs', max_iter=2000))])),  # <- no multi_class



    ("LDA", LDA := LinearDiscriminantAnalysis()),
    ("KNN", Pipeline([('scaler', StandardScaler()),
                      ('clf', KNeighborsClassifier())])),
    ("CART", DecisionTreeClassifier(random_state=7)),
    ("NB",  GaussianNB()),
    ("SVM", Pipeline([('scaler', StandardScaler()),
                      ('clf', SVC(kernel='rbf'))]))
]

cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
print("\nSpot-check (10-fold CV accuracy):")
scores_table = []
for name, model in candidates:
    scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
    scores_table.append((name, scores.mean(), scores.std()))
    print(f"{name:>4}: {scores.mean():.3f} (+/- {scores.std():.3f})")
```

Output: the CV accuracy table

```
Spot-check (10-fold CV accuracy):
  LR: 0.948 (+/- 0.057)
 LDA: 0.973 (+/- 0.041)
 KNN: 0.957 (+/- 0.057)
CART: 0.948 (+/- 0.057)
  NB: 0.939 (+/- 0.068)
 SVM: 0.948 (+/- 0.057)
```

4.Tune best model (GridSearchCV)

Caption: "L14 – Tuning code: SVM GridSearchCV (kernel, C, gamma)."

```
# ----------------------------
# 4) Tune the top model(s)
#    Example: SVM (often strong on Iris)
# ----------------------------
svm_pipe = Pipeline([('scaler', StandardScaler()),
                     ('clf', SVC())])

param_grid = {
    'clf__kernel': ['rbf', 'linear'],
    'clf__C': [0.1, 1, 10, 100],
    'clf__gamma': ['scale', 0.1, 0.01, 0.001]  # gamma used when rbf
}
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:", grid.best_score_)
```

Output: "L14 – Tuning results: best params & CV score."

```
Best SVM params: {'clf__C': 1, 'clf__gamma': 'scale', 'clf__kernel': 'linear'}
Best CV accuracy: 0.9818181818181818
```

5.Try an ensemble

Caption: "L14 – Ensemble code: RF & GB CV evaluation."

```
# ----------------------------
# 5) Try an ensemble (comparison)
# ----------------------------
rf = RandomForestClassifier(n_estimators=200, random_state=7)
gb = GradientBoostingClassifier(random_state=7)
for name, model in [("RF", rf), ("GB", gb)]:
    scores = cross_val_score(model, X_train, y_train, cv=cv, scoring='accuracy')
    print(f"{name} CV accuracy: {scores.mean():.3f} (+/- {scores.std():.3f})")
```

OutPut

```
RF CV accuracy: 0.965 (+/- 0.043)
GB CV accuracy: 0.965 (+/- 0.043)
```

```
# ----------------------------
# 6) Finalize, evaluate on test, save/load
#    (Pick the best: grid.best_estimator_ here)
# ----------------------------
best_model = grid.best_estimator_
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

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

6.Finalize, evaluate, save & reload

Caption: "L14 – Finalize code: fit best model, test metrics, save & reload."

```
# ----------------------------
# 6) Finalize, evaluate on test, save/load
#    (Pick the best: grid.best_estimator_ here)
# ----------------------------
best_model = grid.best_estimator_
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

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

dump(best_model, "iris_best_model.joblib")
loaded = load("iris_best_model.joblib")
print("Loaded model test accuracy (sanity check):",
      accuracy_score(y_test, loaded.predict(X_test)))
```

Output: test accuracy, confusion matrix, classification report, and "Loaded model test accuracy".

```
TEST accuracy: 0.9736842105263158
Confusion matrix:
 [[13  0  0]
 [ 0 12  1]
 [ 0  0 12]]
Classification report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      0.92      0.96        13
   virginica       0.92      1.00      0.96        12

    accuracy                           0.97        38
   macro avg       0.97      0.97      0.97        38
weighted avg       0.98      0.97      0.97        38

Loaded model test accuracy (sanity check): 0.9736842105263158
```

2) Iris Step-by-Step Project
Step 1–6: code + output/plot per step.

Step_1. Install / versions

itec6720_final > scripts > step_by_step > 01_env_versions.py

```python
# Check the versions of libraries

# Python version
import sys
print('Python: {}'.format(sys.version))
# scipy
import scipy
print('scipy: {}'.format(scipy.__version__))
# numpy
import numpy
print('numpy: {}'.format(numpy.__version__))
# matplotlib
import matplotlib
print('matplotlib: {}'.format(matplotlib.__version__))
# pandas
import pandas
print('pandas: {}'.format(pandas.__version__))
# scikit-learn
import sklearn
print('sklearn: {}'.format(sklearn.__version__))
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\step_by_step\01_env_versions.py
Python: 3.11.13 | packaged by Anaconda, Inc. | (main, Jun  5 2025, 13:03:15) [MSC v.1929 64 bit (AMD64)]
scipy: 1.16.1
numpy: 2.3.3
matplotlib: 3.10.1
pandas: 2.3.2
sklearn: 1.7.2
```

Step_2. Load the dataset

itec6720_final > scripts > step_by_step > 02_load_csv.py > ...

```python
# Load libraries
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC


# Load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)

```

Step_3. Summarize the dataset

itec6720_final > scripts > step_by_step > 03_descriptive_stats.py > ...

```python

# summarize the data
from pandas import read_csv
# Load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)
# shape
print(dataset.shape)
# head
print(dataset.head(20))
# descriptions
print(dataset.describe())
# class distribution
print(dataset.groupby('class').size())
```

Output

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\step_by_step\03_descriptive_stats.py
(150, 5)
    sepal-length  sepal-width  petal-length  petal-width        class
0            5.1          3.5           1.4          0.2  Iris-setosa
1            4.9          3.0           1.4          0.2  Iris-setosa
2            4.7          3.2           1.3          0.2  Iris-setosa
3            4.6          3.1           1.5          0.2  Iris-setosa
4            5.0          3.6           1.4          0.2  Iris-setosa
5            5.4          3.9           1.7          0.4  Iris-setosa
6            4.6          3.4           1.4          0.3  Iris-setosa
7            5.0          3.4           1.5          0.2  Iris-setosa
8            4.4          2.9           1.4          0.2  Iris-setosa
9            4.9          3.1           1.5          0.1  Iris-setosa
10           5.4          3.7           1.5          0.2  Iris-setosa
11           4.8          3.4           1.6          0.2  Iris-setosa
12           4.8          3.0           1.4          0.1  Iris-setosa
13           4.3          3.0           1.1          0.1  Iris-setosa
14           5.8          4.0           1.2          0.2  Iris-setosa
15           5.7          4.4           1.5          0.4  Iris-setosa
16           5.4          3.9           1.3          0.4  Iris-setosa
17           5.1          3.5           1.4          0.3  Iris-setosa
18           5.7          3.8           1.7          0.3  Iris-setosa
19           5.1          3.8           1.5          0.3  Iris-setosa
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

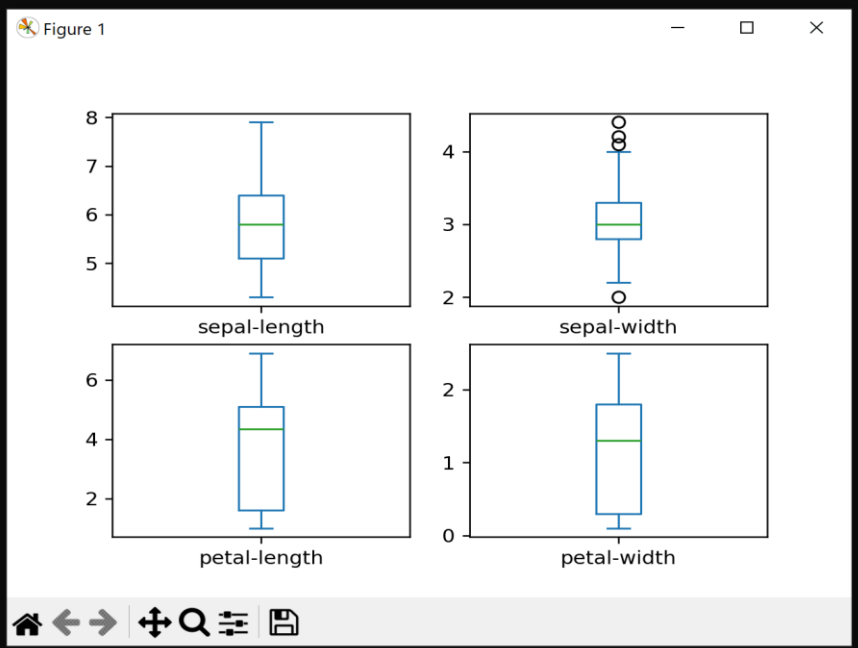
Step_4. Visualize the data

itec6720_final > scripts > step_by_step > 04_visualization.py > ...

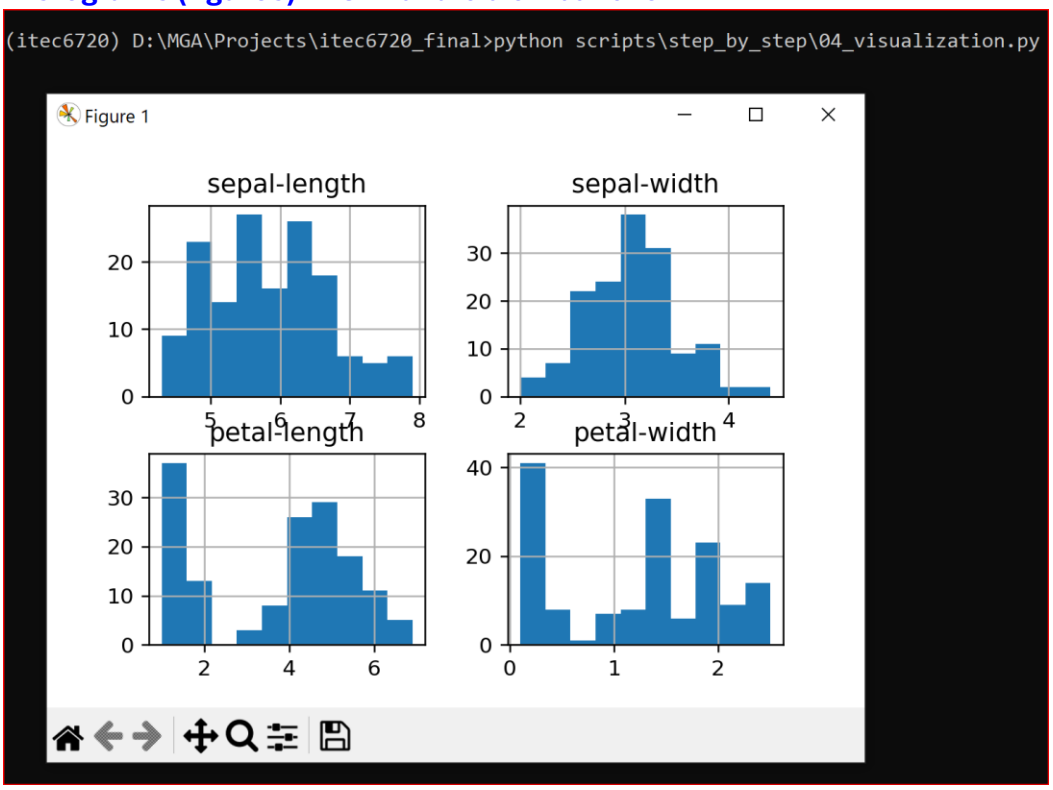
```python

# visualize the data
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot as plt
# Load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)
# box and whisker plots
dataset.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
# histograms
dataset.hist()
plt.show()
# scatter plot matrix
scatter_matrix(dataset)
plt.show()
```

Boxplots (figures) — Box-and-whisker plots for the four Iris features (cm).

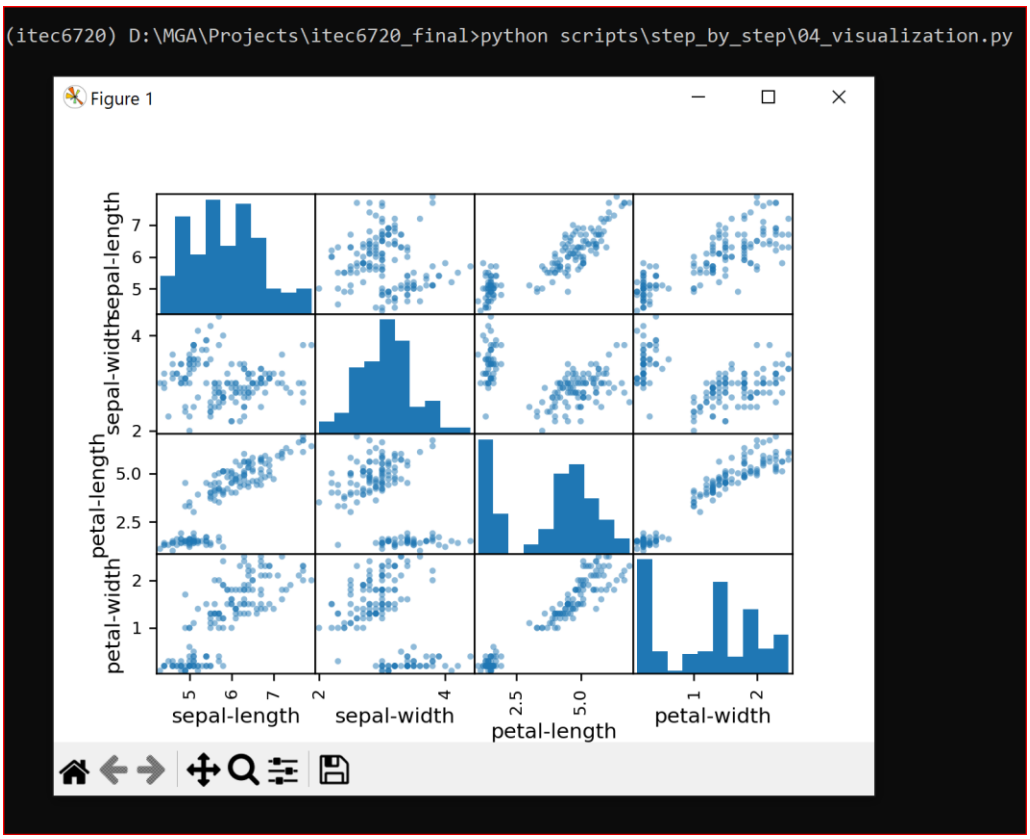
```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\step_by_step\04_visualization.py
```



Histograms (figures) — Univariate distributions.



Scatter-matrix (figures) — Pairwise relationships.



Step_5.Evaluate some algorithms (Spot-check).

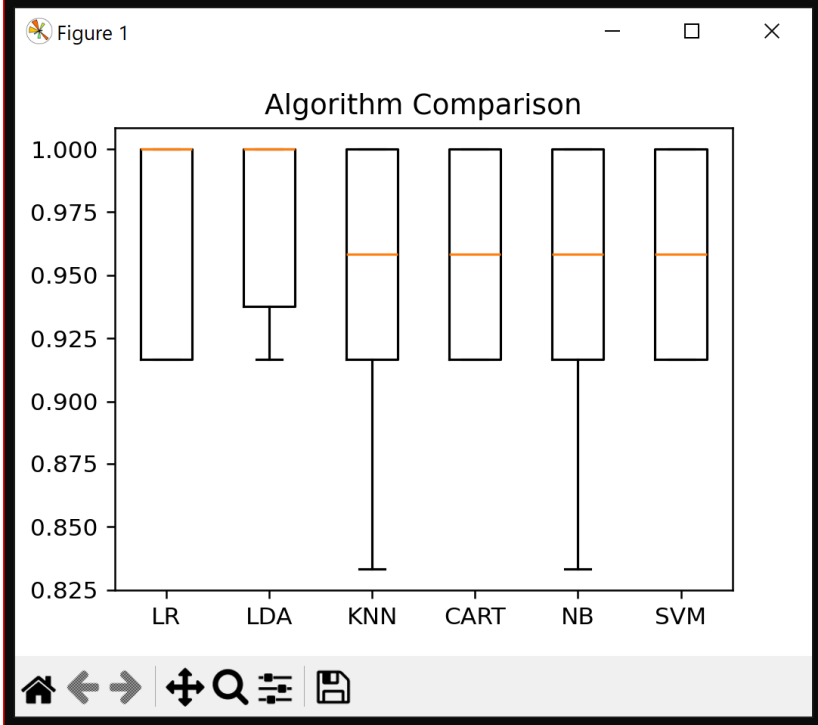
Step-by-Step — 05 Evaluate Algorithms: 10-fold stratified CV accuracy for LR, LDA, KNN, CART, NB, SVM (with pipelines to scale KNN/SVM); prints mean ± std and compares models with a boxplot.

scripts > step_by_step > 05_evaluate_algorithms.py > ...

```python
from pandas import read_csv
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length','sepal-width','petal-length','petal-width','class']
dataset = read_csv(url, names=names)

# Split-out validation dataset
array = dataset.values
X, y = array[:,0:4], array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(
    X, y, test_size=0.20, random_state=1, shuffle=True
)

# Spot Check Algorithms (updated)
models = [
    ('LR',  make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))),  # lbfgs by default; no deprecation
    ('LDA', LinearDiscriminantAnalysis()),
    ('KNN', make_pipeline(StandardScaler(), KNeighborsClassifier())),
    ('CART', DecisionTreeClassifier(random_state=1)),
    ('NB',  GaussianNB()),
    ('SVM', make_pipeline(StandardScaler(), SVC()))  # default gamma='scale'
]

# evaluate each model in turn
results, names = [], []
cv = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
for name, model in models:
    cv_results = cross_val_score(model, X_train, Y_train, cv=cv, scoring='accuracy')
    results.append(cv_results); names.append(name)
    print(f'{name}: {cv_results.mean():.3f} ({cv_results.std():.3f})')

plt.boxplot(results, tick_labels=names)
plt.title('Algorithm Comparison')
```

Mean ± std accuracy for LR, LDA, KNN, CART, NB, SVM printed at top; boxplot shows SVM/LDA near the top with tight spread ⇒ strong, consistent performance.

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\step_by_step\05_evaluate_algorithms.py
LR: 0.967 (0.041)
LDA: 0.975 (0.038)
KNN: 0.950 (0.055)
CART: 0.958 (0.042)
NB: 0.950 (0.055)
SVM: 0.958 (0.042)
```



Step_6. Make predictions

Train an SVC on the 80% training split, predict the 20% validation set, then print accuracy, the confusion matrix, and a full classification report.

itec6720_final > scripts > step_by_step > 06_predictions.py > ...

```python
# make predictions
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
# Load dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
dataset = read_csv(url, names=names)
# Split-out validation dataset
array = dataset.values
X = array[:,0:4]
y = array[:,4]
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20, random_state=1)
# Make predictions on validation dataset
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
# Evaluate predictions
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

Accuracy ≈ 0.967 with 29/30 correct (one *versicolor* predicted as *virginica*). Macro F1 ≈ 0.96; per-class scores are strong across all three classes.

```
(itec6720) D:\MGA\Projects\itec6720_final>python scripts\step_by_step\06_predictions.py
0.9666666666666667
[[11  0  0]
 [ 0 12  1]
 [ 0  0  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      0.92      0.96        13
 Iris-virginica       0.86      1.00      0.92         6

       accuracy                           0.97        30
      macro avg       0.95      0.97      0.96        30
   weighted avg       0.97      0.97      0.97        30
```