EXPLORER

MACHINELEARNING

- .venv
- .vscode
  - settings.json
- HW / WilliamsBenjamin_HW1
  - Exercise1
    - exercise1.py U
    - image.png U
    - wine.data.csv U
  - Exercise2
    - breast-cancer-wisconsin.data.csv U
    - breast-cancer-wisconsin.names U
    - exercise2.py U
    - image.png U
  - Exercise3
    - exercise3.py U
    - image.png U
  - HW1.pdf U
  - WilliamsBenjamin_HW1.zip U
- Lab 1
  - WilliamsBenjamin_Lab1
  - WilliamsBenjamin_Lab1.pdf
  - WilliamsBenjamin_Lab1.zip
- Lab 2 / WilliamsBenjamin_Lab2
  - Exercise1
    - exercise1.py U
    - image.png U
  - Exercise2
    - exercise2.py U
    - image.png U
- .gitignore

exercise1.py .../WilliamsBenjamin_Lab2/... U | image.png .../Exercise1 U | exercise2.py U

Lab 2 > WilliamsBenjamin_Lab2 > Exercise1 > exercise1.py > ...

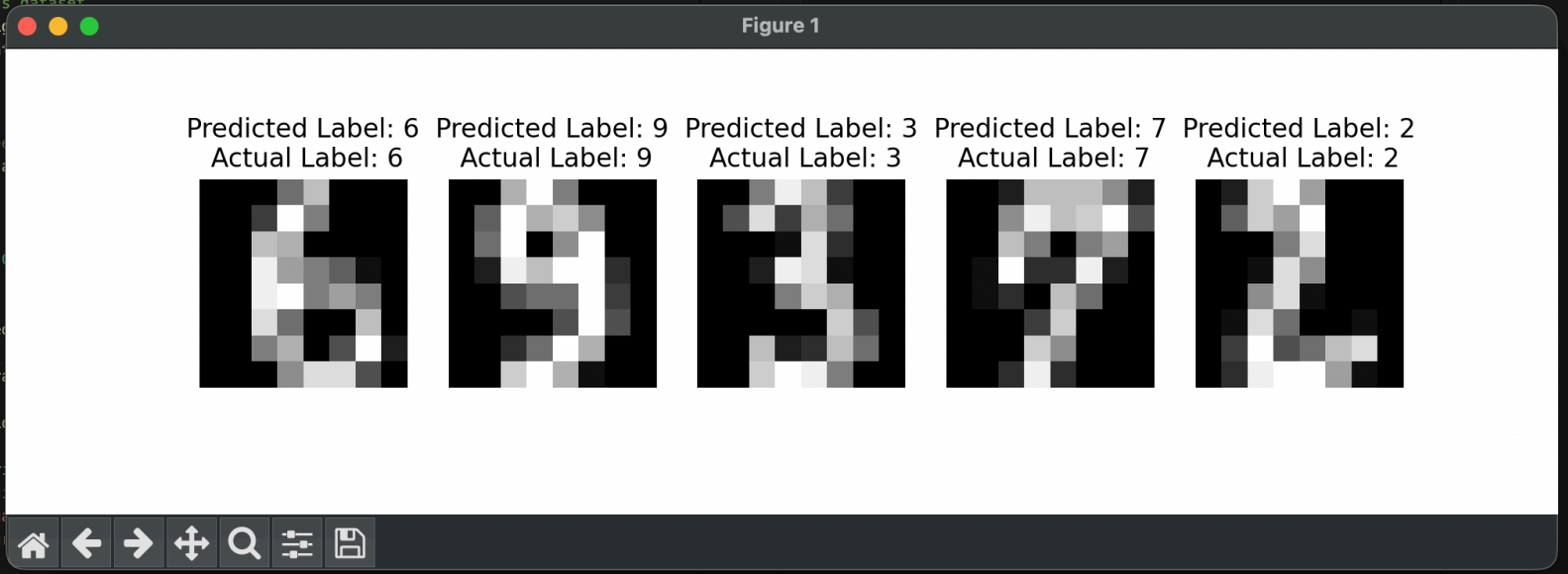
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns

# The handwritten digits dataset contains 1797 images where each image is 8x8
# Thus, we have 64 features (8x8)
# X: features (64)
# y: label (0-9)
# Load the digits dataset
digits = load_dig
X, y = digits.da
print(f'Shape X:
print(f'Shape y:

# I wanted to ke
index_list = np.a

x_train, x_test,

knn = KNeighbors
knn.fit(x_train,

y_pred = knn.pred

accuracy = accura

matrix = confusi

sns.heatmap(matr
plt.xlabel("Pred
plt.ylabel("Actu
plt.title(f"Accu
plt.show()

# Visualize some samples
fig, axes = plt.subplots(1, 5, figsize=(10, 3))
for ax, idx in zip(axes, range(5)):
    ax.imshow(digits.images[i_test[idx]], cmap='gray')
    ax.set_title(f'Predicted Label: {y_pred[idx]}\n Actual Label: {y_test[idx]}')
    ax.axis('off')
plt.show()
```

CHAT TERMINAL

```
source /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/activate
/Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/python "/Users/benwilliams/Documents/Development/MachineLearning/Lab 2/WilliamsBenjamin_Lab2/Exercise1/exercise1.py"
benwilliams@Mac MachineLearning % source /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/activate
(.venv) benwilliams@Mac MachineLearning % /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/python "/Users/benwilliams/Documents/Development/MachineLearning/Lab 2/WilliamsBenjamin_Lab2/Exercise1/exercise1.py"
Shape X: (1797, 64)
Shape y: (1797,)
```

zsh
zsh William...
Python

Figure 1



MachineLearning

exercise1.py .../WilliamsBenjamin_Lab2/... U | exercise2.py U | image.png U | exerci

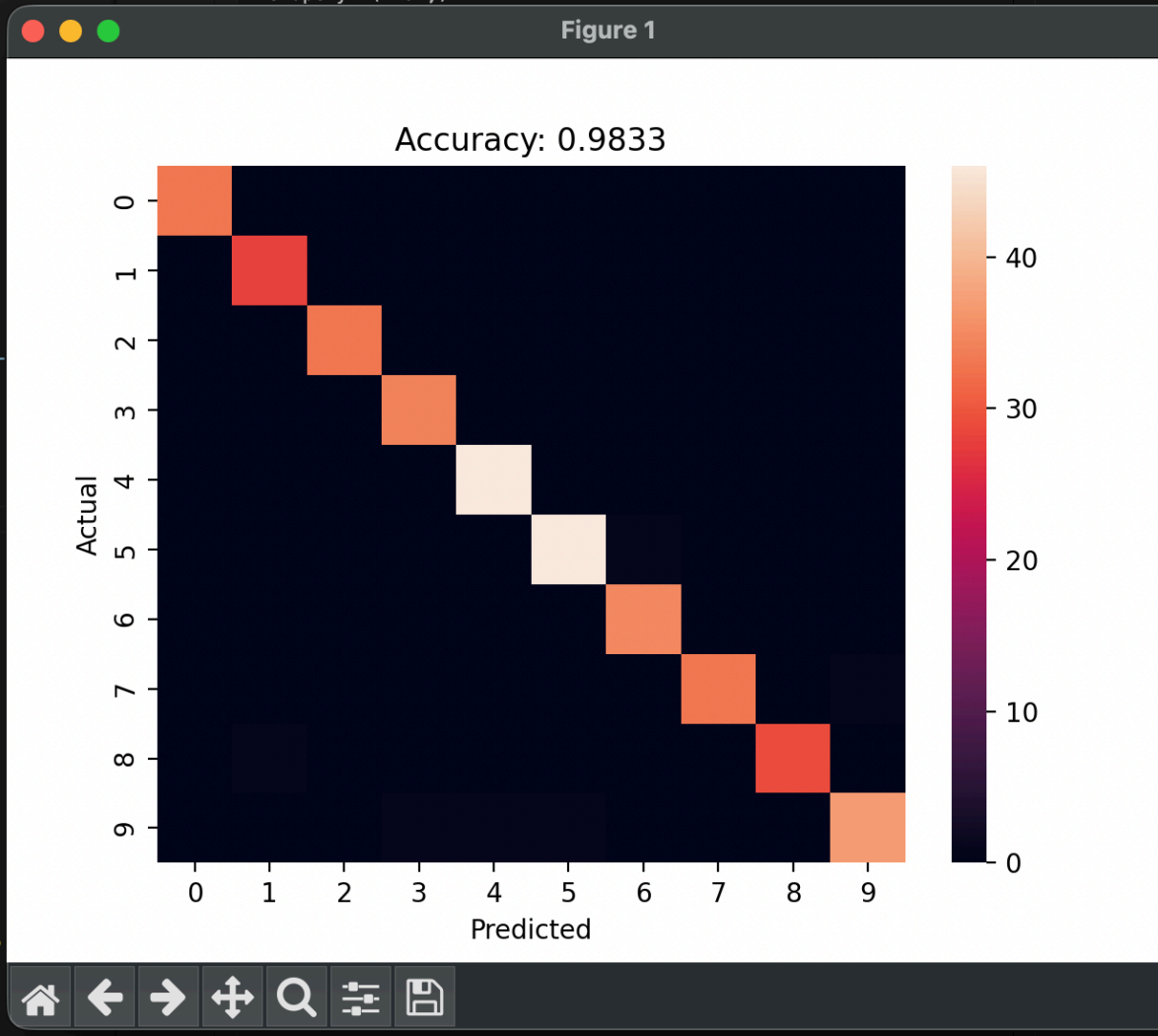
Lab 2 > WilliamsBenjamin_Lab2 > Exercise1 > exercise1.py > ...

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns

# The handwritten digits dataset contains 1797 images where each image is 8x8
# Thus, we have 64 features (8x8)
# X: features (64)
# y: label (0-9)
# Load the digits dataset
digits = load_digits()
X, y = digits.data, digits.target
print(f'Shape X: {X.shape}')
print(f'Shape y: {y.shape}')

# I wanted to keep track of the test indicies for later
index_list = np.arange(X.shape[0])

x_train, x_test, y_train, y_test, i_train, i_test = train_test_split(X, y, index_

knn = KNeighborsClassifier(3)
knn.fit(x_train, y_train)

y_pred = knn.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

matrix = confusion_matrix(y_test, y_pred)

sns.heatmap(matrix)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title(f"Accuracy: {accuracy:.4f}")
plt.show()

# Visualize some samples
fig, axes = plt.subplots(1, 5, figsize=(10, 3))
for ax, idx in zip(axes, range(5)):
    ax.imshow(digits.images[i_test[idx]], cmap='gray')
    ax.set_title(f'Predicted Label: {y_pred[idx]}\n Actual Label: {y_test[idx]}')
    ax.axis('off')
plt.show()
```

CHAT TERMINAL

```
source /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/activate
/Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/python "/Users/benwilliams/Documents/Development/MachineLearning/Lab 2/WilliamsBenjamin_Lab2/Exercise1/exercise1.py"
benwilliams@Mac MachineLearning % source /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/activate
(.venv) benwilliams@Mac MachineLearning % /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/python "/Users/benwilliams/Documents/Development/MachineLearning/Lab 2/WilliamsBenjamin_Lab2/Exercise1/exercise1.py"
Shape X: (1797, 64)
Shape y: (1797,)
```

zsh
zsh Will
Python



msBenjamin_Lab2/... U ML_L2Ex1_sample.py U exercise2.py U exercise1.py .../Williar

Lab 2 > WilliamsBenjamin_Lab2 > Exercise2 > exercise2.py > ...

```python
data = np.array([[1, 5],[3, 2],[8, 4],[7, 14]], dtype=float)

def my_mean(col):
    s = 0.0
    n = len(col)
    for v in col:
        s += v
    return s / n

def my_std(col):
    mu = my_mean(col)
    s = 0.0
    n = len(col)
    for v in col:
        s += (v - mu) ** 2
    return (s / n) ** 0.5

def standardize(X):
    rows, cols = X.shape
    means = [my_mean(X[:, j]) for j in range(cols)]
    stds  = [my_std(X[:, j])  for j in range(cols)]

    Z = np.zeros_like(X, dtype=float)
    for i in range(rows):
        for j in range(cols):
            Z[i, j] = (X[i, j] - means[j]) / stds[j]

    return Z, np.array(means), np.array(stds)

def inverse_standardize(Z, means, stds):
    rows, cols = Z.shape
    X_rec = np.zeros_like(Z, dtype=float)
    for i in range(rows):
        for j in range(cols):
            X_rec[i, j] = Z[i, j] * stds[j] + means[j]
    return X_rec

Z_my, means_my, stds_my = standardize(data)
data_back_my = inverse_standardize(Z_my, means_my, stds_my)
scaler = StandardScaler(with_mean=True, with_std=True)
Z_sk = scaler.fit_transform(data)
data_back_sk = scaler.inverse_transform(Z_sk)

print("Original data:\n", data)
print("\nMy standardized:\n", Z_my)
print("\nMy inverse standardized:\n", data_back_my)
print("\nSklearn standardized:\n", Z_sk)
print("\nSklearn inverse standardized:\n", data_back_sk)
print("\nMax abs diff (standardized):", np.max(np.abs(Z_my - Z_sk)))
print("Max abs diff (inverse):      ", np.max(np.abs(data_back_my - data_back_sk)))
```

CHAT TERMINAL

```
 [ 7. 14.]]

My standardized:
 [[-1.31055608 -0.27156272]
 [-0.61159284 -0.92331326]
 [ 1.13581527 -0.4888129 ]
 [ 0.78633365  1.68368888]]

My inverse standardized:
 [[ 1.  5.]
 [ 3.  2.]
 [ 8.  4.]
 [ 7. 14.]]

Sklearn standardized:
 [[-1.31055608 -0.27156272]
 [-0.61159284 -0.92331326]
 [ 1.13581527 -0.4888129 ]
 [ 0.78633365  1.68368888]]

Sklearn inverse standardized:
 [[ 1.  5.]
 [ 3.  2.]
 [ 8.  4.]
 [ 7. 14.]]

Max abs diff (standardized): 0.0
Max abs diff (inverse):      0.0
(.venv) benwilliams@Mac MachineLearning % /Users/benwilliams/Documents/Development/MachineLearning/.venv/bin/python "/Users/benwilliams/Documents/Development/MachineLearning/Lab 2/WilliamsBenjamin_Lab2/Exercise2/exercise2.py"
Original data:
 [[ 1.  5.]
 [ 3.  2.]
 [ 8.  4.]
 [ 7. 14.]]

My standardized:
 [[-1.31055608 -0.27156272]
 [-0.61159284 -0.92331326]
 [ 1.13581527 -0.4888129 ]
 [ 0.78633365  1.68368888]]

My inverse standardized:
 [[ 1.  5.]
 [ 3.  2.]
 [ 8.  4.]
 [ 7. 14.]]

Sklearn standardized:
 [[-1.31055608 -0.27156272]
 [-0.61159284 -0.92331326]
 [ 1.13581527 -0.4888129 ]
 [ 0.78633365  1.68368888]]

Sklearn inverse standardized:
 [[ 1.  5.]
 [ 3.  2.]
 [ 8.  4.]
 [ 7. 14.]]

Max abs diff (standardized): 0.0
Max abs diff (inverse):      0.0
(.venv) benwilliams@Mac MachineLearning %
```