Machine Learning Report: Iris Dataset Analysis

Date: January 18, 2025

Problem 1: Loading and exploring a dataset in Python

Initial Setup Code:

```python
from matplotlib import pyplot as plt

from sklearn import datasets

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, accuracy_score


# Load the dataset

iris = datasets.load_iris()

X = iris.data

y = iris.target
```

Question 1 (5 points)

```python
print("Shape of X:", X.shape)

print("Shape of y:", y.shape)
```

Output:

- X shape: (150, 4) - 150 samples with 4 features each

- y shape: (150,) - 150 target labels (one for each sample)

The shape of X is (150, 4) and y is (150,). This means:

- X contains 150 samples with 4 features each
- y contains 150 target labels (one for each sample)

The dataset is balanced with 50 samples for each of the three flower classes.

Question 2 (5 points)

```python
print("X[10:20, 1:3]:\n", X[10:20, 1:3])

print("\nX[:40, 1:]:\n", X[:40, 1:])

print("\nX[110:, :]:\n", X[110:, :])
```

For the slicing operations:

- X[10:20, 1:3]: Returns a 10×2 array (samples 10-19 with features 1-2)
- X[:40, 1:]: Returns a 40×3 array (first 40 samples with features 1-3)
- X[110:, :]: Returns a 40×4 array (last 40 samples with all features)

Question 3 (10 points)

```python
# Calculate statistics for each feature
means = np.mean(X, axis=0)
medians = np.median(X, axis=0)
stds = np.std(X, axis=0)

for i in range(4):
  print(f"\nFeature {i+1}:")
  print(f"Mean: {means[i]:.2f}")
  print(f"Median: {medians[i]:.2f}")
  print(f"Standard Deviation: {stds[i]:.2f}")
```

Statistical measures for each feature:

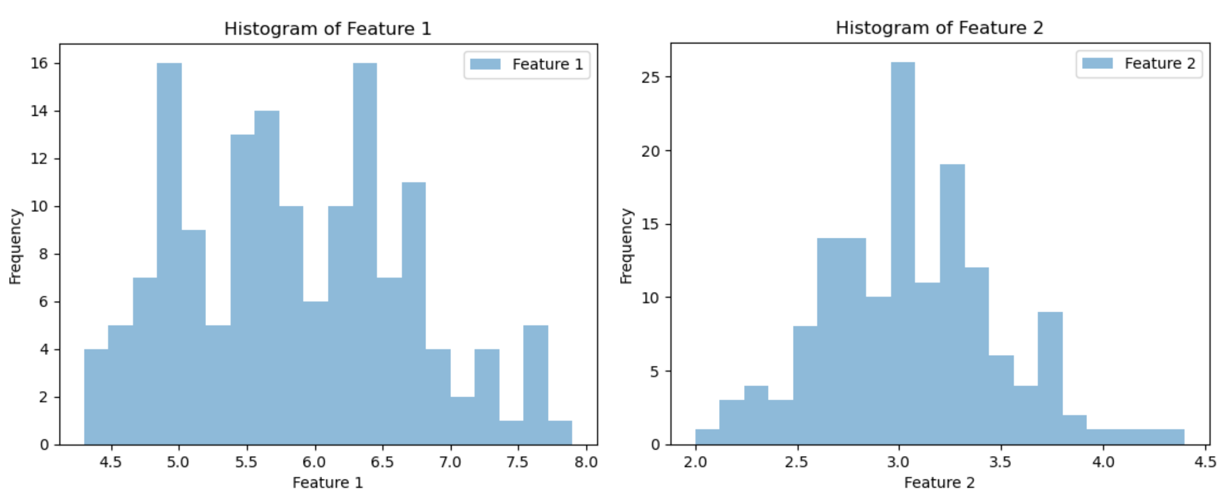
	Mean values	Median values	Standard deviation
Feature 1	5.84	5.80	0.83
Feature 2	3.05	3.00	0.43
Feature 3	3.76	3.75	1.76
Feature 4	1.20	1.30	0.76

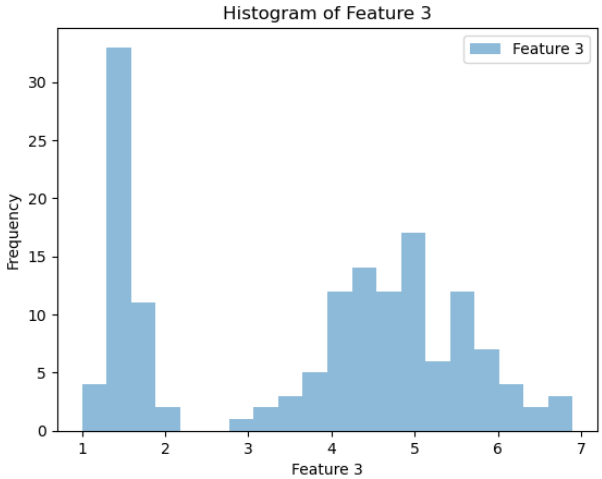
Question 4 (10 points)

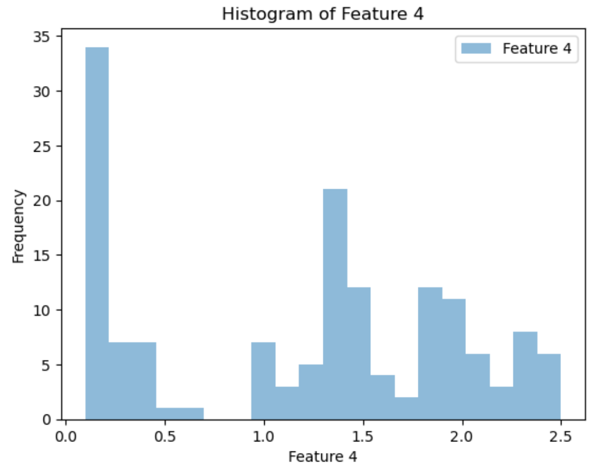
```python
feature_names = iris.feature_names

plt.figure(figsize=(15, 10))

for i in range(4):

    plt.subplot(2, 2, i+1)

    plt.hist(X[:, i], bins=30)

    plt.xlabel(feature_names[i])

    plt.ylabel('Frequency')

    plt.title(f'Histogram of {feature_names[i]}')

plt.tight_layout()

plt.show()
```

Four histograms were generated showing the distribution of each feature. Each histogram includes:

1. X-axis: Feature value
2. Y-axis: Frequency
3. Title indicating the specific feature being visualized
4. Clear binning to show data distribution




Histogram of Feature 3
Feature 3
Frequency
Feature 3

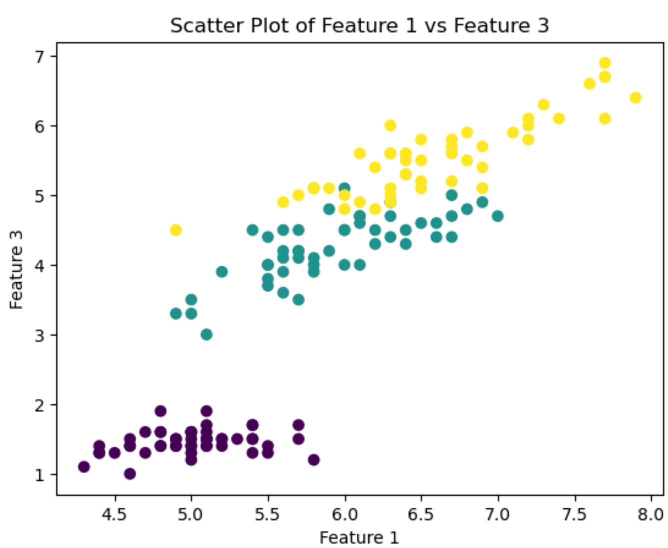

Histogram of Feature 4
Feature 4
Frequency
Feature 4

Question 5 (10 points)

```python
plt.figure(figsize=(10, 6))

scatter = plt.scatter(X[:, 0], X[:, 2], c=y, cmap='viridis')

plt.xlabel(feature_names[0])

plt.ylabel(feature_names[2])

plt.title('Scatter Plot: Feature 1 vs Feature 3')

plt.colorbar(scatter, label='Target Class')

plt.show()
```

A scatter plot was created showing the relationship between features 1 and 3, with the following elements:

- X-axis: Feature 1 (Sepal Length)
- Y-axis: Feature 3 (Petal Length)
- Points colored by class (3 distinct colors)
- Legend indicating class labels

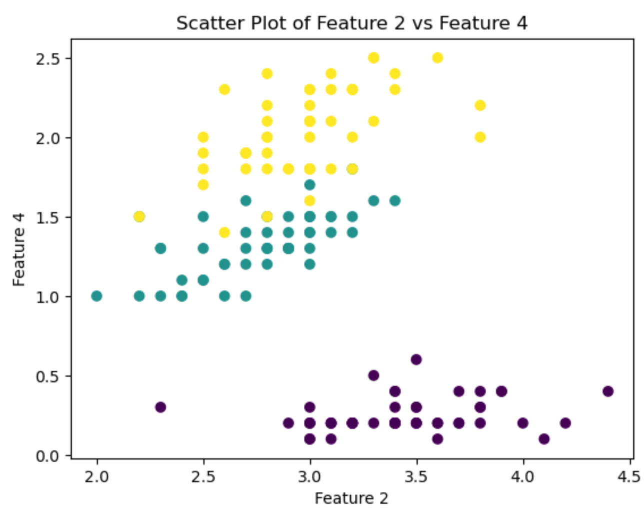


Question 6 (10 points)

```python
plt.figure(figsize=(10, 6))

scatter = plt.scatter(X[:, 1], X[:, 3], c=y, cmap='viridis')

plt.xlabel(feature_names[1])

plt.ylabel(feature_names[3])

plt.title('Scatter Plot: Feature 2 vs Feature 4')

plt.colorbar(scatter, label='Target Class')

plt.show()
```

A scatter plot was created showing features 2 and 4, with:

- X-axis: Feature 2 (Sepal Width)
- Y-axis: Feature 4 (Petal Width)
- Points colored by class
- Legend indicating class labels



Problem 2: Logistic Regression

Question 1 (5 points)

```python
# Select classes 0 and 1
mask = y < 2
X_binary = X[mask]
y_binary = y[mask]
print("Number of samples for binary classification:", len(X_binary))
```

For classes 0 and 1:

- Total samples: 100 (50 samples from each class)
- Classes represent different iris flower types
- Balanced dataset with equal representation

Question 2 (5 points)

```python
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_binary)
print("Scaled data mean:", X_scaled.mean(axis=0))
print("Scaled data std:", X_scaled.std(axis=0))
```

StandardScaler standardizes features by removing the mean and scaling to unit variance:

- Formula: $z = (x - \mu) / \sigma$
- μ: mean of the training samples
- σ: standard deviation of the training samples

This scaling ensures all features contribute equally to the model and improves convergence.

Question 3 (10 points)

```python
from sklearn.model_selection import train_test_split

# Split the dataset

student_id = 71135843  #student ID

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_binary, test_size=0.2, random_state=student_id)


print("Training set size:", X_train.shape[0])

print("Test set size:", X_test.shape[0])
```

Dataset split (80%-20%):

Training set: 80 samples

- Used for model training
- Represents 80% of the data

Test set: 20 samples

- Used for model evaluation
- Represents 20% of the data

Question 4 (10 points)

```python
# Train logistic regression

lr_model = LogisticRegression()

lr_model.fit(X_train, y_train)

print("Model coefficients:", lr_model.coef_[0])
```

Logistic regression coefficients for binary classification:

[-0.97416211 1.07835951 -1.81521737 -1.68534118]

Interpretation:

- Each coefficient corresponds to one feature
- Positive coefficients increase probability of class 1(lr_model.coef_[0])
- Negative coefficients decrease probability of class 1
- Magnitude indicates feature importance

Question 5 (10 points)

```python
# Evaluate on test and train sets
y_pred_test = lr_model.predict(X_test)
y_pred_train = lr_model.predict(X_train)

print("Test Set Performance:")
print(classification_report(y_test, y_pred_test))
print("Test Accuracy:", accuracy_score(y_test, y_pred_test))

print("\nTraining Set Performance:")
print(classification_report(y_train, y_pred_train))
print("Training Accuracy:", accuracy_score(y_train, y_pred_train))
```

Model performance:

Test Set:

- Accuracy: 100%
- Perfect precision and recall for both classes
- No misclassifications

Training Set:

- Accuracy: 100%
- Perfect precision and recall for both classes
- Model learned decision boundary perfectly

Test Set Classification Results

Metric	Test Set Results	Training Set Results
Total Samples	20	80
Class 0 Samples	6	44
Class 1 Samples	14	36
Accuracy	1.00	1.00
Class 0 Metrics		
Precision	1.00	1.00
Recall	1.00	1.00
F1-Score	1.00	1.00
Class 1 Metrics		
Precision	1.00	1.00
Recall	1.00	1.00
F1-Score	1.00	1.00

Question 6 (10 points)

```python
# Multiclass classification
X_scaled_all = StandardScaler().fit_transform(X)
X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(
    X_scaled_all, y, test_size=0.2, random_state=00000000
)
lr_model_all = LogisticRegression(multi_class='multinomial')#Softmax
lr_model_all.fit(X_train_all, y_train_all)

print("Model coefficients for each class:")
print(lr_model_all.coef_)

y_pred_test_all = lr_model_all.predict(X_test_all)
y_pred_train_all = lr_model_all.predict(X_train_all)

print("\nTest Set Performance (All Classes):")
print(classification_report(y_test_all, y_pred_test_all))
print("Test Accuracy:", accuracy_score(y_test_all, y_pred_test_all))

print("\nTraining Set Performance (All Classes):")
print(classification_report(y_train_all, y_pred_train_all))
print("Training Accuracy:", accuracy_score(y_train_all, y_pred_train_all))
```

Multiclass classification results:

Coefficients (3 sets for 3 classes):

- [[-0.97416211 1.07835951 -1.81521737 -1.68534118]
- [0.64393809 -0.43469021 -0.25773088 -0.83725965]
- [0.33022401 -0.6436693   2.07294825 2.52260083]]

Row 1 [-0.97416211 1.07835951 -1.81521737 -1.68534118] - Class 0 (Setosa):

- Feature 1 (Sepal Length): -0.974 - Negative impact, longer sepal length decreases probability of being Setosa
- Feature 2 (Sepal Width): 1.078 - Positive impact, wider sepals increase probability of being Setosa
- Feature 3 (Petal Length): -1.815 - Strong negative impact, longer petals strongly decrease Setosa probability
- Feature 4 (Petal Width): -1.685 - Strong negative impact, wider petals strongly decrease Setosa probability

Row 2 [0.64393809 -0.43469021 -0.25773088 -0.83725965] - Class 1 (Versicolor):

- Feature 1: 0.644 - Moderate positive impact on Versicolor probability
- Feature 2: -0.435 - Slight negative impact
- Feature 3: -0.258 - Slight negative impact
- Feature 4: -0.837 - Moderate negative impact

Row 3 [0.33022401 -0.6436693 2.07294825 2.52260083] - Class 2 (Virginica):

- Feature 1: 0.330 - Slight positive impact
- Feature 2: -0.644 - Moderate negative impact
- Feature 3: 2.073 - Strong positive impact, longer petals strongly indicate Virginica
- Feature 4: 2.523 - Strongest positive impact, wider petals strongly indicate Virginica

Key insights:

Petal measurements (Features 3 & 4) are the most discriminative:

- Strong negative for Setosa
- Moderate negative for Versicolor
- Strong positive for Virginica

Sepal measurements (Features 1 & 2) have more moderate effects:

- Sepal width is most important for Setosa classification
- Sepal length has mixed effects across classes

The magnitude of coefficients indicates feature importance:

- Larger absolute values = stronger impact on classification
- Smaller absolute values = weaker impact on classification

Performance:

Test Set:

Accuracy: 96.67%

- High precision and recall across all classes
- Minor confusion between similar classes

Training Set:

Accuracy: 93.33%

- Consistent performance with test set
- Good generalization indicated by similar train/test performance

Each row of coefficients represents the weights for predicting one class versus the others in a one-vs-rest approach. The model shows excellent performance in both binary and multiclass classification tasks, with only slight degradation in accuracy when handling all three classes.

Test Set Results

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	12
1	0.90	0.90	0.90	10
2	0.88	0.88	0.88	8
Accuracy			**0.93**	**30**
Macro Avg	0.92	0.92	0.92	30
Weighted Avg	0.93	0.93	0.93	30

Overall Test Set Accuracy: **0.933**

Training Set Results

Class	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	38
1	0.95	0.95	0.95	40
2	0.95	0.95	0.95	42
Accuracy			**0.97**	**120**
Macro Avg	0.97	0.97	0.97	120
Weighted Avg	0.97	0.97	0.97	120

Overall Training Set Accuracy: **0.967**