Module 2 Lab Exercise: Tools Used in Machine Learning

Learning Objectives

By the end of this lab, you will be able to:

- Set up and navigate Jupyter Notebook, Google Colab, and VS Code environments
- Install and import essential Python libraries for machine learning
- Create and format professional documentation using Markdown
- Initialize a GitHub repository for your ML projects
- Understand the basic workflow of data science tools

Prerequisites

- Basic understanding of what machine learning is (Module 1)
- Access to internet for downloading tools and datasets
- A Google account (for Colab) or local Python installation

Part 1: Environment Setup and Tool Overview

What are the main tools we'll use in this course?

Jupyter Notebook/Google Colab: Interactive computing environments where you can write code, see results immediately, and document your work with text and visualizations.

Python Libraries: Pre-written code packages that make machine learning tasks easier:

- **Pandas**: For working with data (like Excel, but more powerful)
- **NumPy**: For mathematical operations on arrays of numbers
- **Matplotlib**: For creating charts and graphs
- **Scikit-learn**: The main library for machine learning algorithms

GitHub: A platform to store, share, and collaborate on code projects

VS Code: A powerful text editor for writing and debugging code

Let's start by setting up our environment!

Environment Setup Instructions

Option 1: Google Colab (Recommended for Beginners)

1. Go to colab.research.google.com
2. Sign in with your Google account
3. Click "New Notebook"
4. You're ready to go! Libraries are pre-installed.

Option 2: Local Jupyter Notebook

1. Install Python from python.org
2. Open terminal/command prompt
3. Run: `pip install jupyter pandas numpy matplotlib scikit-learn`
4. Run: `jupyter notebook`
5. Create a new notebook

Option 3: VS Code

1. Download VS Code from code.visualstudio.com
2. Install Python extension
3. Install Jupyter extension
4. Create a new .ipynb file

For this lab, we recommend starting with Google Colab as it requires no installation.

Import Libraries

This cell installs (if needed) and imports core libraries. It suppresses warnings for a clean output and verifies versions to ensure compatibility.

```
# Install required libraries (uncomment if needed; in Colab, most are pre-installed)
# !pip install pandas numpy matplotlib scikit-learn

# Import libraries with standard aliases for data manipulation, numerical operations, plotting, and ML datasets
import pandas as pd             # For data handling in DataFrame format
import numpy as np              # For numerical computations and arrays
import matplotlib.pyplot as plt # For creating visualizations
from sklearn import datasets    # For loading built-in datasets like Iris

# Import warnings to suppress unnecessary output for cleaner notebook display
import warnings
warnings.filterwarnings('ignore')  # Hide warning messages for cleaner output

# Print success messages and versions to verify imports
print("✅ All libraries imported successfully!")
print(f"Pandas version: {pd.__version__}")
print(f"NumPy version: {np.__version__}")
```

```
✅ All libraries imported successfully!
Pandas version: 2.2.2
NumPy version: 2.0.2
```

Part 2: Loading and Exploring Your First Dataset

We'll use the famous Iris dataset - a classic dataset for beginners. It contains measurements of iris flowers from three different species.

This section introduces the Iris dataset (measurements of iris flowers from three species) as a beginner-friendly example. Prepares the user for data loading and exploration of Iris dataset using scikit-learn's built-in function and prints basic info like shape (150 samples, 4 features) and names.

```
# Load a simple dataset (Iris flowers - a classic beginner dataset)
# Import the load_iris function from scikit-learn to access the built-in Iris dataset
from sklearn.datasets import load_iris

# Load the Iris dataset into a Bunch object (contains data, target, feature names, etc.)
iris = load_iris()
print("Dataset loaded successfully!")
print(f"Dataset shape: {iris.data.shape}")
print(f"Features: {iris.feature_names}")
print(f"Target classes: {iris.target_names}")
```

```
Dataset loaded successfully!
Dataset shape: (150, 4)
Features: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target classes: ['setosa' 'versicolor' 'virginica']
```

Converts the raw dataset to a Pandas DataFrame, adds a species column, and displays the head (first 5 rows) and info (structure) for initial exploration.

```
# Convert the Iris data (numpy array) to a Pandas DataFrame for easier manipulation and analysis
df = pd.DataFrame(iris.data, columns=iris.feature_names)  # Create DataFrame with feature columns
df['species'] = iris.target_names[iris.target]            # Add a new column for species names by mapping numerical targets t

# Display the first few rows of the DataFrame to inspect the data
print("First 5 rows of our dataset:")
print(df.head())

# Print DataFrame info: data types, non-null counts, memory usage
print("\nDataset info:")
print(df.info())
```

```
First 5 rows of our dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
```

```
Dataset info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

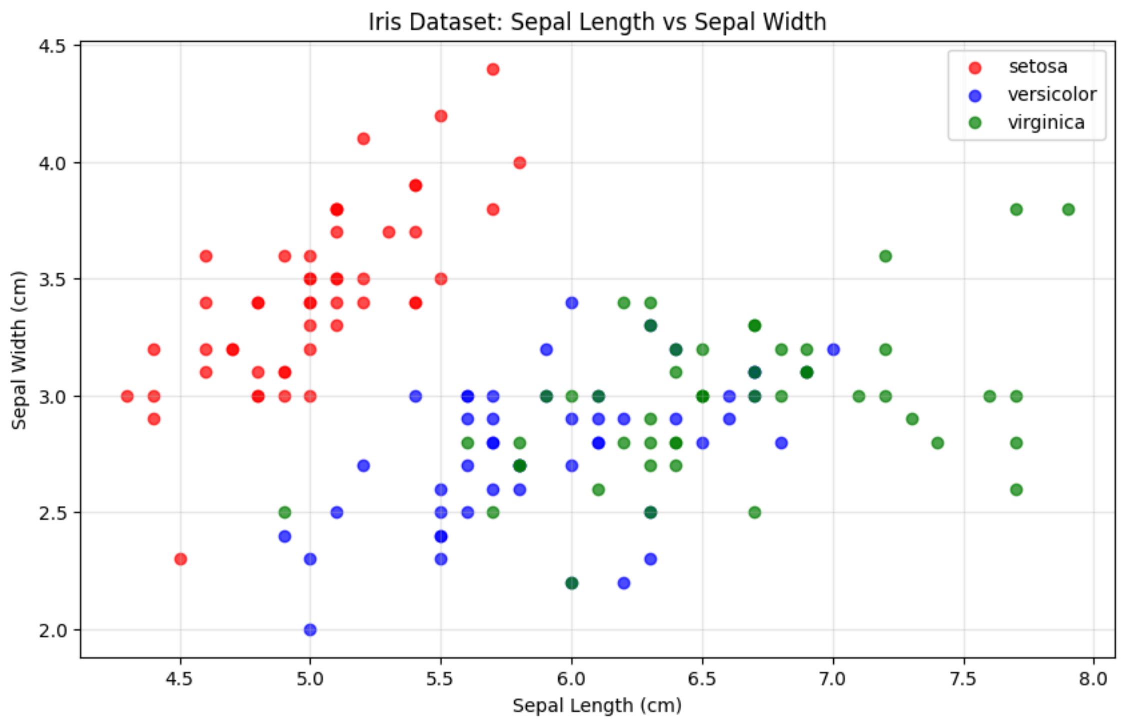
Part 3: Creating Your First Visualization

Data visualization is crucial in machine learning. Let's create a simple plot to understand our data.

Emphasizes the importance of data visualization in ML and introduces creating a simple plot to understand the data.

This cell creates a scatter plot of sepal dimensions, colored by species, to visualize separations between classes. It uses Matplotlib for plotting.

```python
# Task 1: Create a scatter plot of sepal length vs sepal width, colored by species
# Create a figure with specified size for the plot
plt.figure(figsize=(10, 6))

# Plot sepal length vs sepal width, colored by species
species_colors = {'setosa': 'red', 'versicolor': 'blue', 'virginica': 'green'}

# Scatter plot: group by species and plot points with corresponding colors
for species in df['species'].unique():
    species_data = df[df['species'] == species]
    plt.scatter(species_data['sepal length (cm)'],
                species_data['sepal width (cm)'],
                c=species_colors[species],
                label=species,
                alpha=0.7)    # alpha for transparency

# Add title and labels to the plot
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.title('Iris Dataset: Sepal Length vs Sepal Width')
plt.legend()                # Add legend to distinguish species
plt.grid(True, alpha=0.3)
plt.show()                  # Display the plot

# Print success message
print("🎉 Congratulations! You've created your first data visualization!")
```



```
🎉 Congratulations! You've created your first data visualization!
```

Part 4: Practice with Basic Data Operations

Let's practice some basic data analysis operations that you'll use throughout the course.

This sections counts species occurrences and plots a bar chart to show balance (50 each). It helps identify if the dataset is balanced. It extends the exploration of the Iris dataset by adding quantitative insights to the visual analysis. The code uses the same DataFrame (df) created before, where the Iris dataset was converted to a Pandas DataFrame with features and species columns.

```
# Basic statistical analysis
print("Basic Statistics for Iris Dataset:")   # Print a header to clearly separate this section's output
print("=" * 40)                               # Print 40 equal signs as a visual separator

# Calculate mean values for each feature, grouped by species
# groupby('species') groups the data by species, and mean() computes the average for each numeric column
species_means = df.groupby('species').mean()
print("\nMean values by species:")
print(species_means)             # Display the mean values in a table format (DataFrame)

# Count the number of samples per species using value_counts()
# This returns a Series with species names as indices and their respective counts
species_counts = df['species'].value_counts()
print("\nSamples per species:")
print(species_counts)            # Display the counts for each species
```

```
Basic Statistics for Iris Dataset:
========================================

Mean values by species:
            sepal length (cm)  sepal width (cm)  petal length (cm)  \
species
setosa                  5.006             3.428              1.462
versicolor              5.936             2.770              4.260
virginica               6.588             2.974              5.552

            petal width (cm)
species
setosa                 0.246
versicolor             1.326
virginica              2.026

Samples per species:
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

This section performs basic statistical analysis by computing mean feature values per species and confirming sample counts. It uses Pandas' groupby and value_counts to summarize the data, providing insights into feature distributions and class balance, which are critical for machine learning tasks.

Part 5: GitHub and Documentation Best Practices

Why GitHub for Machine Learning?

- **Version Control**: Track changes to your code and data
- **Collaboration**: Work with others on projects
- **Portfolio**: Showcase your work to potential employers
- **Backup**: Never lose your work

Basic GitHub Workflow:

1. **Create Repository**: A folder for your project
2. **Clone/Download**: Get the project on your computer
3. **Add Files**: Put your notebooks and data
4. **Commit**: Save a snapshot of your changes
5. **Push**: Upload changes to GitHub

For This Course:

- Create a repository named "ITAI-1371-ML-Labs"
- Upload each lab notebook as you complete it
- Include a README.md file describing your projects

Action Item: After this lab, create your GitHub account and repository.

Assessment: Tool Familiarity Check

Complete the following tasks to demonstrate your understanding of the tools.

Task 2a calculates basic descriptive statistics (mean and standard deviation) for a single feature (sepal length) using NumPy, emphasizing numerical analysis. The assert statements verify the outputs are numeric types, ensuring correctness. This is a foundational step in exploratory data analysis (EDA) to summarize data distributions. For the Iris dataset, the mean sepal length is approximately 5.84 cm, and the standard deviation is 0.83 cm, indicating moderate variability across all samples.

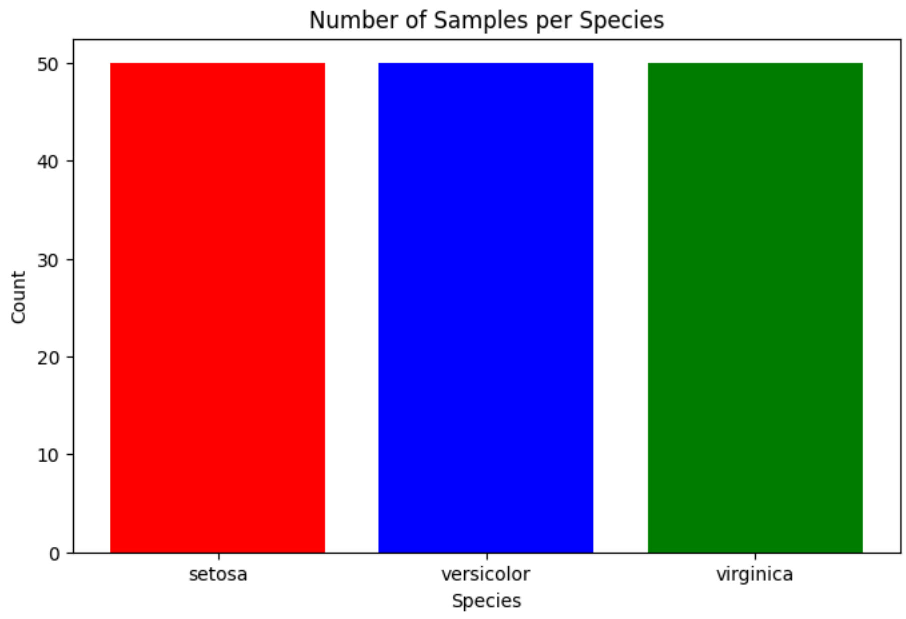
Task 2b is for creating a bar chart to visualize the distribution of samples across the three Iris species using Matplotlib. It uses Pandas' value_counts() to count occurrences per species (50 each, confirming balance), then plots a bar chart for quick insight into class distribution. This helps identify if the dataset is balanced, which is important for fair machine learning models. The colors distinguish species, and the dictionary print provides a textual summary. It is a straightforward extension of EDA to make data interpretable visually.

```
# Task 2a: Calculate the mean and standard deviation of sepal length

# Extract the sepal length column from the DataFrame as a Pandas Series (or NumPy array)
sepal_lengths = df['sepal length (cm)']

# Your code here: Compute mean (average) using NumPy's mean function
mean_sepal_length = np.mean(sepal_lengths)

# Compute standard deviation (population std, ddof=0 by default) using NumPy's std function
std_sepal_length = np.std(sepal_lengths)

# Print results formatted to 2 decimal places for readability
print(f"Mean sepal length: {mean_sepal_length:.2f} cm")
print(f"Standard deviation: {std_sepal_length:.2f} cm")

# Verification (don't modify): Check that results are floating-point numbers
assert isinstance(mean_sepal_length, (float, np.floating)), "Mean should be a number"
assert isinstance(std_sepal_length, (float, np.floating)), "Std should be a number"
print("✅ Task 1 completed successfully!")
```

```
Mean sepal length: 5.84 cm
Standard deviation: 0.83 cm
✅ Task 1 completed successfully!
```

```
# Task 2b: Create a simple bar chart showing species counts

# Count occurrences of each species using value_counts(), returning a Series with species as index and counts as values
species_counts = df['species'].value_counts()

# Create a new figure with dimensions 8x5 inches for the plot
plt.figure(figsize=(8, 5))
```

```
# Plot vertical bars: x-axis uses species names (index), y-axis uses counts (values), with custom colors for each bar
plt.bar(species_counts.index, species_counts.values, color=['red', 'blue', 'green'])

# Add a descriptive title to the plot
plt.title('Number of Samples per Species')

# Label the x-axis (species names)
plt.xlabel('Species')

# Label the y-axis (sample counts)
plt.ylabel('Count')

# Display the plot in the notebook
plt.show()

# Print the counts as a dictionary for textual reference
print(f"Species distribution: {dict(species_counts)}")

# Confirmation message
print("✅ Task 2 completed successfully!")
```



```
Species distribution: {'setosa': np.int64(50), 'versicolor': np.int64(50), 'virginica': np.int64(50)}
✅ Task 2 completed successfully!
```

Your Analysis and Reflection

Instructions: Complete the analysis below by editing this markdown cell.

My Observations About the Iris Dataset

Dataset Overview:

- Number of samples: 150 (from iris.data.shape which is (150, 4)).
- Number of features: 4 (sepal length, sepal width, petal length, petal width).
- Number of classes: 3 (setosa, versicolor, virginica).

Key Findings from the Visualization:

1. The scatter plot shows clear separation between the setosa species (clustered with higher sepal width and lower sepal length) and the other two species (versicolor and virginica), which overlap more with each other. This suggests sepal measurements could be useful for distinguishing setosa but less so for the others.
2. The bar chart indicates a balanced dataset with exactly 50 samples per species, which is ideal for machine learning as it avoids class imbalance issues during model training.
3. From the scatter plot, versicolor and virginica have longer sepals on average compared to setosa, hinting that petal measurements (not shown here) might provide better separation for all classes.

Questions for Further Investigation:

- How do petal length and width contribute to better species separation in a similar scatter plot?
- Would adding more visualizations, like histograms for each feature, reveal any outliers or skewed distributions in the data?

Reflection: *In 2-3 sentences, describe what you learned about using these tools.*

- I learned that Jupyter Notebooks make it easy to combine code, visualizations, and text for interactive data exploration, which is crucial for understanding datasets before building models. Tools like Pandas and Matplotlib simplify data handling and plotting, allowing quick insights without complex setups. Overall, this setup emphasizes the importance of documentation and reproducibility in ML workflows.

Note: This is practice for documenting your machine learning projects professionally.

Lab Summary and Next Steps

What You've Accomplished:

✅ Set up your machine learning development environment
✅ Imported and used essential Python libraries
✅ Loaded and explored your first dataset
✅ Created your first data visualization
✅ Practiced professional documentation with Markdown
✅ Learned about GitHub for project management

Preparation for Module 3:

In the next lab, you'll:

- Learn about different types of machine learning
- Build your first simple classifier
- Understand the complete ML workflow
- Work with more complex datasets

Action Items:

1. **Create your GitHub account** and repository
2. **Upload this completed notebook** to your repository
3. **Experiment** with different visualizations using the Iris dataset
4. **Practice** Markdown formatting in a new notebook