# CE326 Machine Learning - Lab 1 Handout

In this lab, you will explore basic Python concepts and machine learning using Scikit-learn. You will learn how to use NumPy for numerical operations, Pandas for data manipulation, and Matplotlib for plotting. You will also learn to create functions, use global and local variables, and apply control flow structures such as `if-else` and loops. Finally, we will train a simple machine learning model using Scikit-learn.

After completing this lab, you will be able to:

1. Understand Python basics, including NumPy, Pandas, and plotting.
2. Create functions and work with global and local variables.
3. Use flow control structures like `if-else` and `for` loops.
4. Use Google Colab to run Python scripts.
5. Train a basic machine learning model using Scikit-learn.

To get started, make sure you:

- Have a Google account to access Google Colab.

You can open Google Colab by visiting https://colab.research.google.com/ .

## Task 1: Introduction to Python Basics

### NumPy Basics:

NumPy is a powerful library for numerical operations in Python.

1. Import NumPy and create an array.
2. Perform operations on arrays.

```python
import numpy as np

# Create a NumPy array
arr = np.array([1, 2, 3, 4, 5])

# Access elements of the array
print(arr[0])  # First element
print(arr[1:3])  # Elements from index 1 to 2

```

```python
# Array operations
arr = arr + 5
print(arr)

# Create a 2D array
arr_2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr_2d)

```

### Pandas Basics:

Pandas is a powerful library for data manipulation.

1. Import Pandas and create a DataFrame.

2. Perform basic data manipulation.

```python
import pandas as pd

# Create a DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie'], 'Age': [25, 30, 35]}
df = pd.DataFrame(data)

# Access elements in DataFrame
print(df.head())  # First few rows
print(df['Name'])  # Accessing a specific column

```

```python
# Add a new column
df['City'] = ['New York', 'Los Angeles', 'Chicago']
print(df)

# Filtering data
df_filtered = df[df['Age'] > 28]
print(df_filtered)

```

**Creating Functions:**

Functions in Python allow you to organize your code into reusable blocks.

Define a simple function.

```python
# Create a function that adds two numbers
def add_numbers(a, b):
    return a + b

# Call the function
result = add_numbers(10, 5)
print(result)

```

**Global and Local Variables**

Understanding variable scope is crucial in programming.

Define global and local variables.

```python
# Global variable
x = 10

def change_x():
    # Local variable (within function)
    x = 5
    print(f"Inside function: x = {x}")

# Call the function and print global variable
change_x()
print(f"Outside function: x = {x}")

```

**Flow Control (If-Else Statements):**

Flow control allows you to execute different code based on conditions.

Use `if-else` statements.

```python
age = 20

# If-else statement
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")

```

**Loops (For Loops):**

Loops help you iterate over a sequence of values.

Use a `for` loop to iterate over a list.

```python
# List of names
names = ['Alice', 'Bob', 'Charlie']

# For loop
for name in names:
    print(name)

```

**Plotting with Matplotlib:**

Matplotlib is a popular library for creating visualizations.

Plot a simple graph.

```python
import matplotlib.pyplot as plt

# Create data
x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 35]

# Create a plot
plt.plot(x, y)
plt.title("Simple Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()

```

## Task 2: Introduction to Scikit-learn

Scikit-learn is a popular library for machine learning in Python. You will use it to load datasets, split data, and train machine learning models. First, Install Scikit-learn (available in Colab by default, but you can install it if needed).

**Key Components of Scikit-learn**

1. **Datasets**: Scikit-learn provides easy access to several standard datasets, including the Iris dataset, Boston housing data, and handwritten digits. We will use the Iris dataset, a simple dataset of 150 iris flowers, classified into three species.

Example: Loading the Iris dataset.

```python
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target

```

2. **Model Selection**: Scikit-learn offers functions to split data into training and testing sets, and it provides tools like cross-validation to validate models effectively.

```python
from sklearn.model_selection import train_test_split

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

```

3. **Preprocessing**: Preprocessing ensures your data is in the right format for machine learning models. Common preprocessing tasks include scaling data, normalizing features, and encoding categorical variables.

   - **Standardization**: Scaling features to have zero mean and unit variance.
   - **Encoding categorical features**: Convert text labels into numerical labels.

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

```

```python
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
y_train_encoded = encoder.fit_transform(y_train)

```

4. **Model Training and Evaluation**: Scikit-learn provides many algorithms for classification, regression, and clustering. These algorithms follow a simple workflow:

   - **Fit the model**: Train the model on the training data.
   - **Predict**: Make predictions on new data.
   - **Evaluate**: Measure the model's performance using metrics like accuracy for classification or mean squared error for regression.

**Example: Classification with Scikit-learn**

We will train an SVM classifier on the Iris dataset using the following steps.

1. Load and split the data.
2. Preprocess the data.
3. Train the model.
4. Evaluate the model.

```python
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

```

```python
from sklearn.svm import SVC

svm_model = SVC()
svm_model.fit(X_train_scaled, y_train)

```

```python
from sklearn.metrics import accuracy_score

y_pred = svm_model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

```

### Cross-Validation and Model Tuning

Use **Cross-validation** and **GridSearchCV** to optimize model performance.

```python
from sklearn.model_selection import cross_val_score, GridSearchCV

# Perform 5-fold cross-validation
cv_scores = cross_val_score(SVC(), X_train_scaled, y_train, cv=5)
print(f"Cross-Validation Accuracy: {cv_scores.mean() * 100:.2f}%")

# Grid search for hyperparameter tuning
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
grid_search.fit(X_train_scaled, y_train)
best_model = grid_search.best_estimator_

```

### Pipelines for Streamlined Workflow

Use **Pipelines** to streamline preprocessing and training.

```python
from sklearn.pipeline import Pipeline

pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC()) ])

```

```
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%")

```

## Practice Tasks

To reinforce your learning, complete the following practice tasks. These tasks build on the concepts you've learned today.

1. Array Operations (NumPy):
   I. Create a 3x3 matrix using NumPy, and perform addition, subtraction, and element-wise multiplication with another matrix of the same size.
   II. Extract the second row and the last column of the matrix using slicing.

2. DataFrame Manipulation (Pandas):
   I. Create a DataFrame with three columns: 'Name', 'Age', and 'Score'. Add a new column that categorizes the score into "Pass" or "Fail" based on a threshold.
   II. Filter the DataFrame to show only the rows where the score is above 70.

3. Function Creation:
   Write a Python function that calculates the factorial of a given number.

4. Machine Learning Challenge:
   I. Train a Decision Tree model on the Iris dataset instead of the SVM. Use *sklearn.tree.DecisionTreeClassifier* and compare its accuracy to the SVM model.
   II. Try visualizing the decision tree using Matplotlib and the *plot_tree* function from Scikit-learn.