# Week 2: Fundamentals and ML Specific Concepts

## Assignment 1: Exploring Data with Pandas and Seaborn

Task:

- Load a dataset using Pandas.
- Explore the dataset (e.g., check for missing values, data types, summary statistics).
- Visualize some important features using Seaborn.

*

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Iris dataset
url = "https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv"
df = pd.read_csv(url)

# Explore the dataset
print("First few rows of the dataframe:")
print(df.head())

print("\nSummary statistics:")
print(df.describe())

print("\nMissing values:")
print(df.isnull().sum())

print("\nData types:")
print(df.dtypes)

# Visualize important features
# Scatter plot of sepal length vs sepal width
```

```
sns.scatterplot(x='sepal_length', y='sepal_width', data=df, hue='species')
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.show()

# Box plot of petal length by species
sns.boxplot(x='species', y='petal_length', data=df)
plt.title('Box Plot of Petal Length by Species')
plt.show()
```

~$ python3 week_2-1.py

First few rows of the dataframe:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |

Summary statistics:

| | 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 |

Missing values:

| | |
|---|---|
| sepal_length | 0 |
| sepal_width | 0 |
| petal_length | 0 |
| petal_width | 0 |

species 0

dtype: int64

Data types:

sepal_length float64

sepal_width float64

petal_length float64

petal_width float64

species object

dtype: object

---

## Assignment 2: Implementing Logistic Regression

Task:

- Load a dataset suitable for classification.
- Split the dataset into training and testing sets using train_test_split.
- Train a Logistic Regression model on the training data.
- Evaluate the model using accuracy and F1-score.

*

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, f1_score

# Step 1: Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Step 2: Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Step 3: Train a Logistic Regression model on the training data
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 4: Evaluate the model using accuracy and F1-score
# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Calculate F1-score
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1-score:", f1)
```

```
~$ python3 week_2-2.py
Accuracy: 1.0
F1-score: 1.0
```

---

## Assignment 3: NLP Task with NLTK

Task:

- Preprocess a text dataset using NLTK.
- Perform stemming and lemmatization.
- Tokenize the text using regexp tokenizer.

*

---

```
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import PorterStemmer, WordNetLemmatizer
```

```
from nltk.corpus import stopwords

# Sample list of documents
documents = [
    "This is a sample sentence.",
    "Another example sentence.",
    "Yet another example for demonstration."
]

# Step 1: Convert text to lowercase
documents = [doc.lower() for doc in documents]

# Step 2: Remove punctuation and tokenize the text
tokenizer = RegexpTokenizer(r'\w+')
documents = [tokenizer.tokenize(doc) for doc in documents]

# Step 3: Remove stopwords
stop_words = set(stopwords.words('english'))
documents = [[word for word in doc if word not in stop_words] for doc in documents]

# Step 4: Perform stemming
porter_stemmer = PorterStemmer()
stemmed_documents = [[porter_stemmer.stem(word) for word in doc] for doc in documents]

# Step 5: Perform lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_documents = [[lemmatizer.lemmatize(word) for word in doc] for doc in documents]

# Step 6: Tokenize the text using regexp tokenizer
tokenizer = RegexpTokenizer(r'\w+')
tokenized_documents = [tokenizer.tokenize(' '.join(doc)) for doc in documents]

# Print the results
```

```
print("Original Documents:")
for doc in documents:
    print(doc)

print("\nStemmed Documents:")
for doc in stemmed_documents:
    print(doc)

print("\nLemmatized Documents:")
for doc in lemmatized_documents:
    print(doc)

print("\nTokenized Documents:")
for doc in tokenized_documents:
    print(doc)
```

```
$ python3 week_2-3.py
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/nagakeerthana_123/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/nagakeerthana_123/nltk_data...
Original Documents:
['sample', 'sentence']
['another', 'example', 'sentence']
['yet', 'another', 'example', 'demonstration']

Stemmed Documents:
['sampl', 'sentenc']
['anoth', 'exampl', 'sentenc']
['yet', 'anoth', 'exampl', 'demonstr']

Lemmatized_documents:
```

['sample', 'sentence']

['another', 'example', 'sentence']

['yet', 'another', 'example', 'demonstration']

Tokenized Documents:

['sample', 'sentence']

['another', 'example', 'sentence']

['yet', 'another', 'example', 'demonstration']