Module 2: Scikit-Learn Assignment (100 pts)

For this assignment, we will analyze the dataset Iris using scikit-learn library.

The main purpose of this assignment is to get familiar with Scikit-Learn. Later, we will learn how to use these tools better.

Problem #1 (30 pts)

Import the following:

- `numpy as np`
- `pandas as pd`
- `seaborn as sns`

You should be familiar with the first two libraries. Please, read about the Seaborn library **HERE**.

```
import numpy as np
import pandas as pd
import seaborn as sns
```

Problem #2 (30 pts)

Here we will import `load_iris` function from the datasets module using scikit-learn. Please, complete the code by printing the dataset using `print()`

```
#Here learn how to use the scikit-learn library with datasets
from sklearn.datasets import load_iris
iris = load_iris()
```

Question #1: (10 pts)

What are the ***target names*** of the Iris dataset?

✓ 0s completed at 2:08 PM

Answer:

```
#Below print the Iris dataset and answer the questions (10 pts)

print(iris)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
```

Question #2: (10 pts)

What is the name of the ***feature matrix*** and ***target array*** in the Iris dataset?

Answer:

- Feature matrix: Flowers with n_features: Sepal length and width, Petal length and width.
- target array: Flower Species

Problem #3 (20 pts)

STEP 1: (5 pts)

Now we load the Iris dataset using pandas to compare to problem #2 as follows:

- Download the `Iris.csv` dataset in `Files` on Canvas.
- Upload this dataset in Colab.

STEP 2: (5 pts)

Use `.read_csv()` to load the Iris dataset:

```
#Solution:
#Don't forget to copy the path of the uploaded data file

df = pd.read_csv('Iris.csv')
df
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa

1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

STEP 3: (5 pts)

Use `.head()` to see part of the dataset.

```
#Solution:

df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

STEP 4: (5 pts)

Use `.info()` to know more about the dataframe.

```
#Solution

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

Problem #4 (20 pts)

Question 1: (10 pts)

What differences do you see between Problem 2 and Problem 3? Name at least 3 of them.

Answer:

1. With the print() function we only see the individual data in numpy arrays.
2. Using pandas we can line up the target array with the feature matrix
3. In the DataFrame structure we can get information about the data types along with a null count.

Question 2: (10 pts)

Modeling with scikit-learn using Problem 3 as follows:

- Use `.drop` to create the feature matrix and store it in the variable `X`. You use the following code:

```
X = data.drop(['Id', 'Species'], axis=1)
```

- Create the target array and store it in the variable `y` as follows:

```
y = data['Species']
```

```
#Make sure to copy the code here to run X and y
```

```
#Make sure to copy the code here to run X and y

x = df.drop(['Id', 'Species'], axis=1)
y = df['Species']
```

Use `print()` and `.shape()` to know the shape of the `y` variable.

```
 print(y.shape)
```

Now use `print()` and `.shape()` to know the shape of the `X` variable.

```
#Print the shape of X

print(x.shape)

     (150, 4)
```