CYBR 486 - Lab #5 – Perceptron

This lab will have us using a new dataset imported from scikit-learn. This dataset contains information on breast cancer cases with various features used to determine 2 labels, whether a tissue sample is malignant or benign. We will be using this dataset to create a perceptron binary classifier.

Imports:

```
from sklearn.linear_model import Perceptron
# https://scikit-learn.org/dev/modules/generated/sklearn.linear_model.Perceptron.html
from sklearn.datasets import load_breast_cancer
# https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
# https://scikit-learn.org/stable/modules/model_evaluation.html
from sklearn.model_selection import train_test_split
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
```

Dataset creation:

```
data_X, data_y = load_breast_cancer(return_X_y=True, as_frame=True)
```

Tasks to complete for this lab:

1.) **Display the information about the dataset, including the data types, and determine whether or not there are any null entries in the dataset that might inhibit model training.**

```
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
dtype: int64

First 5 rows of the dataset:
   mean radius  mean texture  ...  worst symmetry  worst fractal dimension
0        17.99         10.38  ...          0.4601                  0.11890
1        20.57         17.77  ...          0.2750                  0.08902
2        19.69         21.25  ...          0.3613                  0.08758
3        11.42         20.38  ...          0.6638                  0.17300
4        20.29         14.34  ...          0.2364                  0.07678

[5 rows x 30 columns]
```

```
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
None

Are there any null values in the dataset?
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
```

```python
# Import necessary libraries
from sklearn.datasets import load_breast_cancer
import pandas as pd

# Load the breast cancer dataset
data_X, data_y = load_breast_cancer(return_X_y=True, as_frame=True)

# Display dataset information (data types, number of entries, etc.)
print("Dataset information:")
print(data_X.info())

# Check if there are any null values
print("\nAre there any null values in the dataset?")
print(data_X.isnull().sum())  # This will return the count of null values per column

# Display the first few rows of the dataset to understand its structure
print("\nFirst 5 rows of the dataset:")
print(data_X.head())
```

```
C:\Users\Bhavy\PycharmProjects\lab33\venv\Scripts\python.exe C:/Users/Bhavy/PycharmProjects/lab33/5a.py
Dataset information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
```

2.) Split the dataset into 80% training 20% test as we've done previously

```python
# Import necessary libraries
from sklearn.model_selection import train_test_split

# Split the dataset into 80% training and 20% testing sets
X_train, X_test, y_train, y_test = train_test_split(data_X, data_y, test_size=0.2, random_state=42)

# Print the shapes of the resulting datasets to confirm the split
print("Training data shape (X):", X_train.shape)
print("Testing data shape (X):", X_test.shape)
print("Training data shape (y):", y_train.shape)
print("Testing data shape (y):", y_test.shape)
```

```
First 5 rows of the dataset:
   mean radius  mean texture  ...  worst symmetry  worst fractal dimension
0        17.99         10.38  ...          0.4601                  0.11890
1        20.57         17.77  ...          0.2750                  0.08902
2        19.69         21.25  ...          0.3613                  0.08758
3        11.42         20.38  ...          0.6638                  0.17300
4        20.29         14.34  ...          0.2364                  0.07678

[5 rows x 30 columns]
Training data shape (X): (455, 30)
Testing data shape (X): (114, 30)
Training data shape (y): (455,)
Testing data shape (y): (114,)
```

3.) Build and train the Perceptron model object using the training set

a. You can check the documentation for the perceptron, there are various arguments you can provide it but for this case you don’t really need to you can just create the perceptron object with the default settings.
b. Training the model works just like the Linear Regression model, you call .fit() on the model object with the X training subset, and the y training subset.

```python
# Import necessary libraries
from sklearn.linear_model import Perceptron

# Step 3a: Create the Perceptron model with default settings
perceptron_model = Perceptron()

# Step 3b: Train the Perceptron model on the training data
perceptron_model.fit(X_train, y_train)

# Output to confirm the model has been trained
print("Perceptron model has been trained.")
```

4.) Make predictions based on the test set, then evaluate the performance of the model. You must create and display the <u>confusion matrix, accuracy score, precision score and finally the recall score</u>.

a. Predictions work like how they did for the linear regression model as well, you call create a prediction variable for the function to return, then call .predict() on the perceptron object with the X test subset

b. The evaluation methods are their own imported functions, you can just call them with the 2 evaluation values, the y test set and the predictions generated from the perceptron model

```
[5 rows x 30 columns]
Training data shape (X): (455, 30)
Testing data shape (X): (114, 30)
Training data shape (y): (455,)
Testing data shape (y): (114,)
Perceptron model has been trained.
Accuracy: 0.9473684210526315
Precision: 0.9710144927536232
Recall: 0.9436619718309859
Confusion Matrix:
 [[41  2]
 [ 4 67]]
```

```python
# Import necessary evaluation functions
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

# Step 4a: Make predictions on the test data
y_pred = perceptron_model.predict(X_test)

# Step 4b: Evaluate the model using accuracy, precision, recall, and confusion matrix

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate precision
precision = precision_score(y_test, y_pred)

# Calculate recall
recall = recall_score(y_test, y_pred)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Display the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("Confusion Matrix:\n", conf_matrix)
```

Take a screenshot and upload or attach it to this document after performing each major step of the lab.