AT - Lesson 78 - Project_Question Copy

March 21, 2023

0.0.1 Instructions

Goal of the Project This project is designed for you to practice and solve the activities that are based on the concepts covered in the Logistic Regression Lessons

---

Getting Started:

1. Click on START Project on the panel and follow the instructions given below.r
2. Create a duplicate copy of the Colab file as described below.

- Click on the **File menu**. A new drop-down list will appear.
- Click on the **Save a copy in Drive** option. A duplicate copy will get created. It will open up in the new tab on your web browser.

3. After creating the duplicate copy of the notebook, please rename it in the **YYYY-MM-DD__StudentName__Project78** format.
4. Now, write your code in the prescribed code cells.

0.0.2 Problem Statement

Nowadays everyone possesses one or more email accounts for digital communication. You may have observed that your mailbox classifies your relevant emails as primary emails and other irrelevant emails as spam.

Spam emails are junk emails or unrequired emails. They may consist of advertisements, updated, and unwanted messages which are sent to the receivers without their permission. Hence, spam detection is one of the important problem statement in email and messaging servicing.

In this project, you will create a model to classify the emails as **spam** or **not spam** using Logistic Regression.

0.0.3 List of Activities

Activity 1: Analysing the Dataset

Activity 2: Train-Test Split

Activity 3: Normalisation of the Features

Activity 4: Logistic Regression - Model Training

Activity 5: Logistic Regression - Model Prediction and Evaluation

---

Activity 1: Analysing the Dataset Create a Pandas DataFrame for **Spambase** dataset using the below link. This dataset consists of the following 58 columns. Most of the columns represent frequencies of a particular word or character in the email:

Column #	Attribute	Description
0 - 47	word_freq_WORD	The first 48 columns are the percentage of the frequencies of the particular word one column each
48 - 53	word_freq_CHAR	The next 6 columns are the percentage of the frequencies of the particular special character like semi-colon (;), exclamation (!) one column each
54	capital_run_length_average	average length of uninterrupted sequences of capital letters
55	capital_run_length_longest	length of longest uninterrupted sequence of capital letters
56	capital_run_length_total	sum of the length of uninterrupted sequences of capital letters
57	email_type	denotes whether the email is spam (1) or not (0)

- **Dataset Link:** https://student-datasets-bucket.s3.ap-south-1.amazonaws.com/whitehat-ds-datasets/spambase.csv

- **Source:** UCI Library https://archive.ics.uci.edu/ml/datasets/Spambase

Print the first five rows of the dataset. Check for null values and treat them accordingly (if any).

```
[ ]: # Import modules

     # Load the dataset

     # Print the first five rows of the DataFrame
```

Rename the last column of the DataFrame as target.

```
[ ]: # Rename the last column as 'target'
```

Print the information of the DataFrame to verify the above update.

```
[ ]: # Print the dataset information
```

Q: Are there any missing values?

A:

Activity 2: Train-Test Split You have to determine the effect of all the features on the 'target' variable. Thus, every column other than the `target` is the feature variable and `target` column is the target variable.

Steps:

1. Create a list of all the features.
2. Split the "DataFrame" into features and target arrays using the features list.
3. Split the dataset into a training set and test set such that the training set contains 70% of the instances and the remaining instances will become the test set.
4. Reshape the target variable arrays into two-dimensional arrays by using `reshape(-1, 1)` function of the `numpy` module.

```
[ ]: # Split the DataFrame into the train and test sets.

     # Import the module


     # Create a features list

     # Split the DataFrame into the train and test sets such that test set has 30%␣
      ↪of the values.

     # Reshape target arrays to 2-dimensional array.
```

Activity 3: Normalisation of the Features Get a descriptive analysis of the feature set and decide whether any normalisation is needed.

Describe the features for training data.

```
[ ]: # Get the descriptive statistics for 'X_train'.
```

Describe the features for the testing data.

```
[ ]: # Get the descriptive statistics for 'X_test'.
```

Q: Does the data needs normalisation? Why?

A:

If the answer to the above question is **yes**, Normalise the data by calculating their Z-scores (or standard scaler) in the following code sections.

Define the Standard Normalisation function.

```
[ ]: # Define the 'standard_scalar()' function for calculating Z-scores
```

Hint Z-score for each value can be calculated by the following expression:

$$Z = \frac{X - \mu}{\sigma}$$

Where,

- X is an observation
- μ is the population mean
- σ is the population standard deviation

Apply the normalisation function to the features of the training data.

```
[ ]: # Apply the 'standard_scalar()' on X_train using apply() function and get the␣
     ↪descriptive statistics of the normalised X_train
```

Apply the normalisation function to the features of the test data.

```
[ ]: # Apply the 'standard_scalar()' on X_test and get the descriptive statistics of␣
     ↪the normalised X_test
```

Activity 4: Logistic Regression - Model Training Implement Logistic Regression Classification using `sklearn` module to estimate the values of β coefficients in the following way:

1. Deploy the model by importing the `LogisticRegression` class and create an object of this class.
2. Call the `fit()` function on the Logistic Regression object and print score using the `score()` function.

3. Print the β coefficient values.

```
[ ]: # Deploy the 'LogisticRegression' model using the 'fit()' function.
```

Get the beta coefficients for the features using the model object trained in the above code.

```
[ ]: # Print the beta coefficient values
```

Activity 5: Logistic Regression - Model Prediction and Evaluation Predict the values for both training and test sets by calling the `predict()` function on the Logistic Regression object.

```
[ ]: # Make predictions on the test dataset by using the 'predict()' function.
```

Also, display the confusion matrix.

```
[ ]: # Display the results of confusion_matrix
```

Q: What is the positive outcome out of both the labels?

A:

Q: Write the count of True Positives and True Negatives?

A:

Print the classification report values to evaluate the accuracy of your model.

```
[ ]: # Display the results of classification_report
```

Q Write the f1-score of both labels?

A:

0.0.4 Submitting the Project

1. After finishing the project, click on the **Share** button on the top right corner of the notebook. A new dialog box will appear.
2. In the dialog box, make sure that '**Anyone on the Internet with this link can view**' option is selected and then click on the **Copy link** button.
3. The link of the duplicate copy (named as **YYYY-MM-DD_StudentName_Project77**) of the notebook will get copied.
4. Go to your dashboard and click on the **My Projects** option.
5. Click on the **View Project** button for the project you want to submit.
6. Click on the **Submit Project Here** button.

7. Paste the link to the project file named as **YYYY-MM-DD__StudentName__Project77** in the URL box and then click on the **Submit** button.