AT - Lesson 80 - 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 following lessons:

1. Logistic Regression - Univariate Classification I
2. Logistic Regression - Decision Boundary
3. Logistic Regression - Multiclass Classification I

Getting Started:

1. Click on START Project on the panel and follow the instructions given below.
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_Project80** format.
4. Now, write your code in the prescribed code cells.

0.0.2 Problem Statement

In this project, you are going to create your own synthetic data for multiclass classification and create a Logistic Regression model to classify the data using Machine Learning.

0.0.3 List of Activities

Activity 1: Create the Dummy Dataset

Activity 2: Dataset Inspection

Activity 3: Train-Test Split

Activity 4: Logistic Regression - Model Training

Activity 5: Model Prediction and Evaluation - Training Set

Activity 6: Model Prediction and Evaluation - Testing Set

Activity 1: Create Dummy Dataset In this activity, you have to create a dummy dataset for multiclass classification.

The steps to be followed are as follows:

1. Create a dummy dataset having two columns representing two independent variables and a third column representing the target.

The number of records should be divided into 6 random groups like `[200, 4270, 7930, 21, 3331, 2721]` such that the target columns has 6 different labels `[0, 1, 2, 3, 4, 5]`.

Recall:

To create a dummy data-frame, use the `make_blob()` function of the `sklearn.datasets` module which will return two arrays `feature_array` and the `target_array`. The syntax for the `make_blob()` function is as follows:

Syntax: `make_blobs(n_samples, centers, n_features, random_state, cluster_std)`

```
[ ]: # Create two arrays using the 'make_blobs()' function and store them in the␣
      ↪'features_array' and 'target_array' variables.
```

Hint:

In the `make_blobs()` function use `n_samples=[200, 4270, 7930,21,3331,2721]` and `center=None` for the division of target label into seven groups.

---

2. Print the object-type of the arrays created by the `make_blob()` function and also print the number of rows and columns in them.

```
[ ]: # Find out the object-type of the arrays created by the 'make_blob()' function␣
      ↪and the number of rows and columns in them.

     # Print the type of 'features_array' and 'target_array'


     # Print the number of rows and column of 'features_array'

     # Print the number of rows and column of 'target_array'
```

Q: How many rows are created in the feature and target columns?

A:

---

3. Create a DataFrame from the two arrays using a Python dictionary.

Steps: (Learnt in "Logistic Regression - Decision Boundary" lesson) - Create a dummy dicitonary.

- Add the feature columns as keys `col 1`, `col 2` and target column as `target`.
- Add the values from the feature and target columns one by one respectively in the dictionary using List Comprehension.
- Convert the dictionary into a DataFrame
- Print first five rows of the DataFrame.

[ ]:
```
# Create a Pandas DataFrame containing the items from the 'features_array' and␣
↪'target_array' arrays.

# Import the module


# Create a dummy dictionary


# Convert the dictionary into DataFrame

# Print first five rows of the DataFrame
```

Hint:

Use function `from_dict()` to convert Python Dictionary to DataFrame.

Syntax: `pd.DataFrame.from_dict(some_dictionary)`

After this activity, the DataFrame should be created with two independent features columns and one dependent target column.

Activity 2: Dataset Inspection In this activity, you have look into the distribution of the labels in the `target` column of the DataFrame.

1. Print the number of occurences of each label in `target` column.

[ ]:
```
# Display the number of occurrences of each label in the 'target' column.
```

2. Print the percentage of the samples for each label in `target` column.

[ ]:
```
# Get the percentage of count of each label samples in the dataset.
```

Q: How many unique labels are present in the DataFrame? What are they?

A:

Q: Is the DataFrame balanced?

A:

---

3. Create a scatter plot between the columns `col 1` and `col 2` for all the labels to visualize the clusters of every class (or points).

```
[ ]: # Create a scatter plot between 'col 1' and 'col 2' columns separately for all␣
      ↪the classes in the same plot.

     # Import the module


     # Define the size of the graph

     # Create a for loop executing for every unique class in `target` column.

       # Plot the scatter plot for 'col 1' and 'col 2' where 'target ==i"


     # Plot the x and y lables

     # Display the legends and the graph
```

Hint: Revise the lesson "Logistic Regression - Decision Boundary".

After this activity, the labels to be predicted i.e the target variables and their distribution should be known.

Activity 3: Train-Test Split We need to predict the value of the `target` variable, using other variables. Thus, `target` is the dependent variable and other columns are the independent variables.

1. Split the dataset into the training set and test set such that the training set contains 70% of the instances and the remaining instances will become the test set.

2. Set `random_state = 42`.

```
[ ]: # Import 'train_test_split' module


     # Create the features data frame holding all the columns except the last column
     # and print first five rows of this dataframe

     # Create the target series that holds last column 'target'
     # and print first five rows of this series

     # Split the train and test sets using the 'train_test_split()' function.
```

3. Print the number of rows and columns in the training and testing set.

```
[ ]: # Print the shape of all the four variables i.e. 'x_train', 'x_test', 'y_train'␣
      ↪and 'y_test'
```

After this activity, the features and target data should be splitted into training and testing data.

Activity 4: Logistic Regression - Model Training Implement Logistic Regression Classification using `sklearn` module 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 the score using the `score()` function.

```
[ ]: # Build a logistic regression model using the 'sklearn' module.


     # 1. Create the Logistic Regression object

     # 2. Call the 'fit()' function with training set as inputs.


     # 3. Call the 'score()' function with training set as inputs to check the␣
      ↪accuracy score of the model.
```

Note: Ignore the warnings if any for now.

After this activity, a Logistic Regression model object should be trained for multiclass classification.

Activity 5: Model Prediction and Evaluation - Training Set **1.** Predict the values for training set by calling the `predict()` function on the Logistic Regression object.

2. Print the unique labels predicted using Logistic Regression on training features.

3. Print the distribution of the labels predicted in the predicted target series for the training features.

```
[ ]: # Predict the values of 'target' by the logistic regression model on the train␣
      ↪set.

     # Predict the target for the training features data

     # Convert the predicted array into series

     # Print the unique labels in the predicted series for training features
```

```
# Print the distribution labels in the predicted series for training features
```

Q: Are all the label values predicted for the training features data?

A:

Q: Which labels are predicted and not predicted by the Logistic Regression model?

A:

After this activity, values of the labels should be predicted for the target columns using training features set and the model should be evaluated for the same.

Activity 6: Model Prediction and Evaluation - Test Set **1.** Predict the values for the test set by calling the `predict()` function on the Logistic Regression object.

2. Print the unique labels predicted using Logistic Regression on test features.

3. Print the distribution of the labels predicted in the predicted target series for the test features.

[ ]:
```
# Predict the values of 'target' by the logistic regression model on the test␣
↪set.

# Predict the target for the test features data


# Convert the predicted array into series

# Print the unique labels in the predicted series for test features

# Print the distribution labels in the predicted series for test features
```

Q: Are all the labels predicted for the test features data?

A:

Q: What are labels predicted and not predicted using Logistic Regression model object?

A:

---

4. Display the confusion matrix for the test set:

[ ]:
```
# Print the confusion matrix for the actual and predicted data of the test set␣
↪(if required)
```

5. Display the classification report for the test set:

```
[ ]: # Print the classification report for the actual and predicted data of the␣
      ↪testing set (if required)
```

After this activity, labels should be predicted for the target columns using test features set and the model should be evaluated for the same.

Write your interpretation of the results here.

- Interpretation 1:
- Interpretation 2:
- Interpretation 3:

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_Project80**) 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_Project80** in the URL box and then click on the **Submit** button.