**PROJECT TITLE: IRIS CLASSIFICATION**

**Class: 20BCS38/A Group No.: \_\_\_\_\_\_\_\_\_\_**

**Group Members Details**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Name** | **UID** |
| **1** | **Aditya Kumar** | **20BCS7555** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Problem Statement**

This problem is centered around predicting the type of an iris flower (setosa, versicolor, or virginica) based on the flower's parameters such as petal length, petal width, etc.

To solve this problem, we build an ML model that takes as inputs 4 parameters:

* petal length
* petal width
* sepal length
* sepal width

and predicts which iris type the flower belongs to:

* setosa
* versicolor
* virginica

To be precise, the model will return probabilities for the flower to belong to each type.

**Key Features/Benefits**

To distinguish between different species of the Iris flower based on four measurements (features): sepal length, sepal width, petal length, and petal width.

**List of Software Used**

* .NET Core 3.1
* ML.NET

**Deliverables**

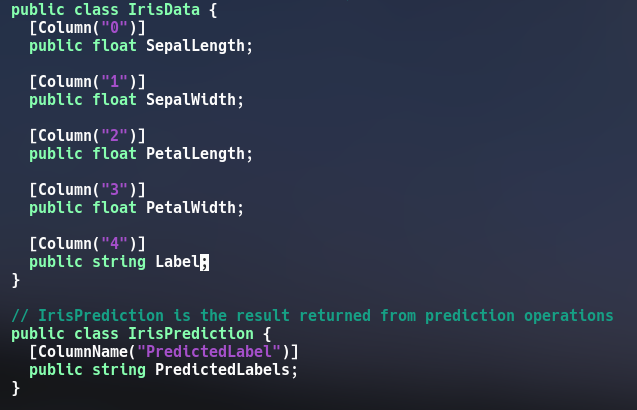
* *Deliverable 1*
* *Deliverable 2*
* *Deliverable 3*
* *Deliverable 4*

**(*Kindly provide the link where all deliverables are uploaded*)**

**Workflow**

**Step – 1: Class Classes that map to the Dataset**

Here we create two classes that represent our dataseet. The first class represents the features and the second class represents the result.



**Step – 2: Create a Pipeline and Load the Data**

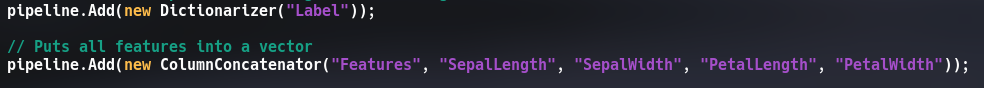
A pipeline is a set of classes in ML.NET that allows us to load external data, transform it to format suitable for training and train the model

.

**Step – 3: Transform the Data**

Data needs to be transformed from string data to numeric data since training can only be carried out on numeric data. So the column labels are assigned numbers. This is achieved by using a Dictionarizer object.

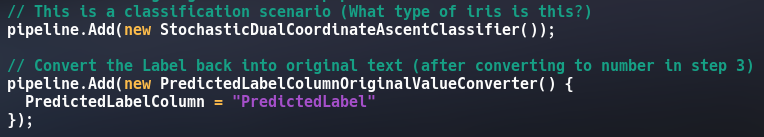
Next, we use a ColumnConcatenator to merge all the features into one single string.



**Step - 4: Add a Learning Algorithm**

Here we add a learning algorithm that can train a classification model. This algorithm is StochasticDualCoordinateClassifier object.

Then we convert back the numeric label into the original text using PredictedLabelColumnOriginalValueConverter object.



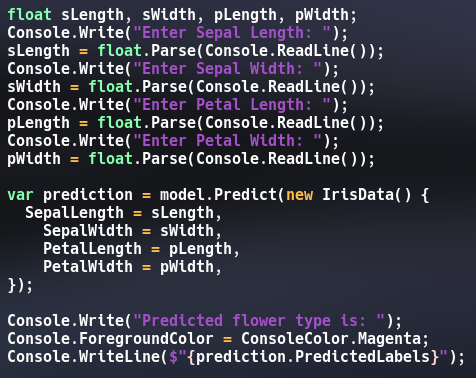
**Step - 5: Train the Model**

In this step, we simply call the Train() method of the pipeline object to train our model.



**Step - 6: Use the Model to make Prediction**

Now we can test the model by passing new set of features and then allow the model to predict the class it belongs.



**Result**

