Machine Learning : Using Scikit - Learn

Objective

The Titanic dataset, which you have seen previously, is an open competition on Kaggle. The objective of this exercise is to apply Sci-kit Learn to build a basic predictive model for the titanic dataset. Once you have built your model you can upload the final predicted results to Kaggle in order to gauge the accuracy.

The following is a list of the features in the dataset. The class we are trying to predict is Survived.

VARIABLE DESCRIPTIONS:

Survived	Survival (0 = No; 1 = Yes)
Pclass	Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
Name	Name
Sex	Sex
Age	Age
SibSp	Number of Siblings/Spouses Aboard
Parch	Number of Parents/Children Aboard
Ticket	Ticket Number
Fare	Passenger Fare
Cabin	Cabin
Embarked	Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

Part A: Generating a Baseline Result

1. You will notice that the exercise folder on Blackboard contains two files. The first is train.csv and the second is test.csv. The files contain different instances of data. The main difference is that the data in train.csv is labelled (that is, it includes the 'Survived' class) for all data instances. However, the test.csv data does not include the label for each instance (this is your unseen data). Your objective is build a model using the data in train.csv and then use it to make predictions from train.csv.

2. In order to submit you results you will need to create an account at Kaggle.

3. Read the two dataset into separate Pandas dataframe (using read_csv method).

4. [**Remove Features**] Some of the features may not contribute to the prediction of the final class.

    a. For this exercise we will remove the Name, Cabin and Ticket features. Drop them entirely from the dataset (you should do this for both the training and test data).

5. [**Missing Values**] You will notice that in the Titanic dataset there are a number of missing values for some features. Determine the number of missing value in each column.

    a. You will notice the **Age** column has the highest number of missing values in both the train and test dataset. Use Scikit's Imputer class to impute the missing value using a strategy of mean.
    b. The **Fare** feature in the test dataset is missing one value. You should again impute for this value using mean.
    c. The **Embarked** feature in the train dataset is missing two values. Delete these specific rows.

6. **[Encoding Categorical Variables]** Because Scikit-learn classification and regression algorithms accept as input as a NumPy array containing continuous values we must convert the category based features to numeric values.
    a. Use the map function in Pandas to map the values for Sex (female and male) to numerical values 0 and 1.
    b. Use the map function to map the values for the Embarked feature to numeric values ({'S': 0, 'C': 1, 'Q': 2}).

7. Next separate the training data into feature training data and class label data (the Survived column).

8. Drop the passenger ID from both the test and training dataset. However, you should store the passenger ID from the test dataset as a series object and will need this data for submitting your model.

9. Next you should use Scikit Learn's Nearest Neighbour class (KNeighborsClassifier) to predict the Survived value for test dataset.

10. Finally, before submitting your results you should merge the results from Step 7 above with the PassengerID feature from step 8 into a single dataframe (see code below). Write the result of the merge to a CSV file. Once you have this done you can submit the results online at Kaggle and see the accuracy of your model.

```
# results below is the NumPy array of predicted results returned from our classifier

resultSeries = pd.Series(data = results, name = 'Survived', dtype='int64')


# create a data frame with just the PassengerID feature from the test dataset and the results

df = pd.DataFrame({"PassengerId":passengerIDSeries, "Survived":resultSeries})


# write the results to a CSV file (you should then upload this file)

 df.to_csv("submission.csv", index=False, header=True)
```

Part B: Normalize the Feature Vectors

1. **[Normalize Data].** Use the MinMaxScaler class to standardize the following features: ["Age", "SibSp", "Parch", "Fare", Pclass]. You would typically normalize the data after you have encoded the categorical variables (after Step 5 in part 1).

2. Resubmit you code to determine the accuracy.