Readme

Assignment submitted by Group_70

Run the code :

Usage : python mlba_assignment1.py --files <full_testdata_filename.csv**>**
<full_traindata_filename**>**

Example : python mlba_assignment1.py --files ./cop/train.csv ./cop/test.csv

Aim: Classification of a given protein sequence into positive and negative proteins based on the dataset given

Provided data: test, train datasets, and sample submission file

Our methodology: We have tried using various machine learning techniques like SVM, Logistic regression, random forest, etc, on the given train dataset. The maximum performance was obtained for the random forest classifier, which has given us an accuracy of around ~92 percent.

With the data we first encode the data using a custom encoder. Then, we apply smote to adjust class imbalances. We have also used random and grid searches to find the best hyperparameters and avoid overfitting.

Steps:

- **Data importing and preprocessing**

We had the train.csv, which had two columns, Sequence and ID. There were over 1000 rows in our dataset. We used the pandas library to read the CSV file and then encoded each sequence into code from 0 to 20 for each protein letter code (20). We then divided them into two lists, sequence and label.

After that, we had two lists
Sequences: That contained the encoded sequences
Label: List containing all the labels(0,1)

```python
#Importing all the required libraries
import subprocess
packages_to_install = ['pandas', 'numpy','sklearn','imbalanced-learn']
for package in packages_to_install:
  subprocess.check_call(['pip', 'install', package])

import argparse
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from imblearn.over_sampling import SMOTE
import collections
import csv
from sklearn.svm import SVC

#Parsing the data from command line
parser=argparse.ArgumentParser()
parser.add_argument("--files",nargs='+',help="Train and test files")
args = parser.parse_args()
args=args.files
train_csv=args[0]
test_csv=args[1]
#Declaring custom encoder
label_encoder = LabelEncoder()
amino_acid_mapping = {char: i  for i, char in enumerate('ACDEFGHIKLMNPQRSTVWY')}

#Loading training data
train_data=pd.read_csv(train_csv)
```

- **Test Train Split**

We then applied the test train split function to split the dataset into test and train set. We have set the hyperparameter test_size=0.3 so that 70 percent would be training and 30 percent of the data would be testing. Inorder to account for the class imbalances, we have use dthe smote classifier. SMOTE is a technique used to address class imbalance in machine learning datasets, where one class is significantly underrepresented compared to the other. It helps by generating synthetic samples of the minority class to balance the dataset.

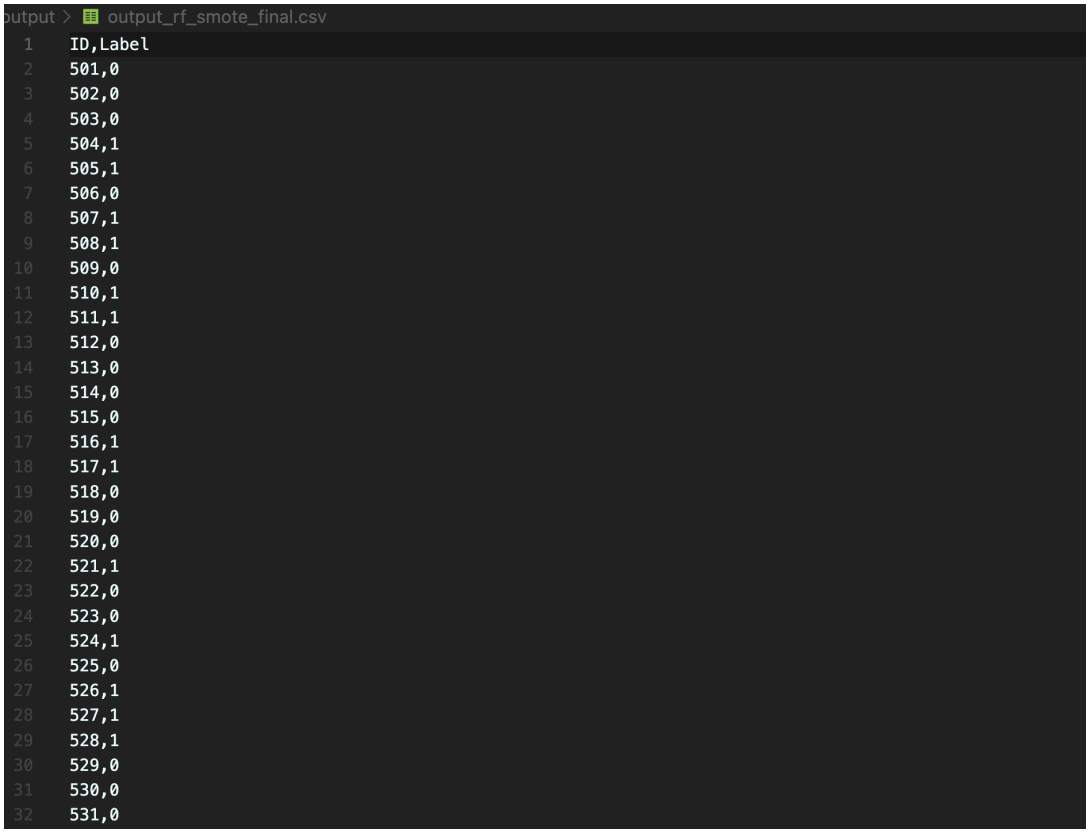
```
[5] print(Sequences)

[[71  4 30 ...  1  9  0]
 [48  8 48 ...  7 27  0]
 [29  0 40 ... 10 18  0]
 ...
 [ 8  4  5 ...  2  2  0]
 [ 3  5  4 ...  0  2  0]
 [12  5  8 ...  0  2  0]]

print(Labels)

[1 1 1 ... 0 0 0]
```

- **Creating the model**

We then applied SVM first to the model, which gave an accuracy of around 86 percent on the training dataset and got around 79 percent accuracy on the Kaggle model.
We then used a RandomForest classifier with hyperparameters set to n_estimators set to 600, max_depth=10, min_samples_split=5. Even though we got an accuracy of 92 percent on the given dataset, the kaggle public dataset gave us only 82.8 percent. We then used grid search and random search and finally arrived at the optimal hyper parameters as n_estimators set to 60, max_depth=None, min_samples_split=2. This gave an accuracy of 91.1 on given local dataset and on kaggle dataset we got the highest accuracy of 83.7 percent.

- **Final output**

We applied the rf model to the unlabelled test dataset and predicted the protein sequences' values. We saved the file in the output folder in the attached folder.



Directory structure: