Logistic Regression Assignment Report

The written program works as a Salary Classifier . It is based on basic logistic regression. The program starts by loading the given dataset into the pandas data-frame.

Once , the data is loaded into the data frame , some of the input parameters in the data are modified for convenience. This includes:-

A. The Date of Birth (DOB) column is modified to Age. Age is calculated from the given DOB, by subtracting the birth year from the current year. Thus the DOB column is replaced by Age (in years).
B. We have certain categorical values which cannot be directly used to apply Logistic Regression . Thus we need to map such categorical values to integers. Thus first we find all possible entries corresponding to a specific categorical parameter.
C. "College State" - All possible values throughout the column are first found using the following command:- **`myData['CollegeState'].value_counts()`**. Then these values are mapped to integers . Similarly , this was done for mostly all Categorical input parameters.
D. Some of the parameters like "10board" and "12board" were simply removed from the experimentation procedure .These factors are simply irrelevant towards the calculation.
E. Then we start experimentation. We try different combinations of input parameters which provide different accuracy levels. We try to maximise accuracy and simultaneously remove unwanted parameters.
F. Some of the input parameters like "ID","CollegeID","CollegeCityTier","CollegeState","12graduation", are simple irrelevant in determining the salary level, hence such parameters are dropped. As a positive impact , removing these parameters increases the accuracy by a great deal .

G. If we exclude the following columns- “Gender”,”Degree","12board"

Accuracy:- 67.81545302946081 %

H. Further a strange result of the experimentation is that if we exclude “Degree”, the overall accuracy increases. However as we all know Degree is a very important determinant of such experiments , hence in real life it cannot be excluded . Thus we drew the following results from this:

(Without including “Degree”) - The way it is done in the code submitted

The output of the accuracy section is:-

Accuracy:- 71.37240886347391 %

The output of the confusion matrix section is:-

```
Confusion Matrix :-

 [[ 868  450]
 [ 351 1129]]
```

The output of the class wise accuracies section is:-

```
Positive class accuracy is :-  76.28378378378379 %
Negative class accuracy is :-  65.85735963581185 %
```

(Including “Degree”)

The output of the accuracy section is:-

Accuracy:- 71.12223016440315 %

The output of the confusion matrix section is:-

```
Confusion Matrix :-

 [[ 859  459]
 [ 349 1131]]
```

The output of the class wise accuracies section is:-

```
Positive class accuracy is :-  76.41891891891892 %
Negative class accuracy is :-  65.17450682852808 %
```

Accuracies when the following input parameters were ignore and only the others were taken into account:

['ElectronicsAndSemicon', 'Gender', 'English', 'Logical', 'ComputerProgramming', 'collegeGPA', 'Quant']
Accuracy:- 58.57755539671193 % *//A fairly low accuracy*
(Test to train ratio= 0.30)

['10percentage', 'Logical', 'Quant', 'ElectricalEngg', 'English']
Accuracy:- 58.79899916597164] % *//A fairly low accuracy*
(Test to train ratio= 0.40)

['Quant', 'agreeableness', 'Logical', 'English', 'Gender', 'CollegeID']
Accuracy:- 61.80408738548273 %

['Age', 'MechanicalEngg', '12graduation', 'extraversion', 'Specialization', 'GraduationYear']
Accuracy:- 62.93164200140944 %

['Quant', 'Domain', 'Logical', 'ComputerProgramming', 'CollegeCityID']
Accuracy:- 65.6800563777308 %

['ElectronicsAndSemicon', 'Logical', 'English', 'conscientiousness']
Accuracy:- 66.83341670835418 %

['agreeableness', 'TelecomEngg', 'CollegeState', 'openess_to_experience', 'ID']
Accuracy:- 70.43521760880441 %
(Test to train ratio= 0.29)

['extraversion', 'GraduationYear', 'Age', 'ID', 'Quant']
Accuracy:- 72.04502814258912 %

I. Another striking feature is that some factors like "College City ID", which are completely irrelevant in the determination. However, given the problem statement, such factors have been included in the classification of salary levels, with the only objective of increasing the accuracy.

J. Further some parameters like , ComputerScience, MechanicalEngg, ElectricalEngg, TelecomEngg , CivilEngg , which though important , do not have data pertaining to all entries and simply have a "-1", however these have still been included as they simply increase the efficiency. These can be removed for a more realistic system for classification.

K. A factor like "College City" poses another striking feature. If we remove this from consideration, the accuracy level remains unaffected . Thus we can simply remove such a factor so as to improve simplicity and time efficiency of the code. This was verified by experimentation.

L. The steps that were followed for experimentation were:-

1. Consider all the parameters initially .
2. Map some of the categorical inputs into numerics so that they can be used.
3. However, at the same time scrape out the Categorical parameters which are completely useless
4. Now , when there is a dataset of only numerics and that too important ones, calculate the accuracy with all numerical columns . Let that accuracy be x.
5. Now, starting with the first column , drop all columns one by one. If by dropping a particular column, the efficiency increases, drop it and move ahead . However if that column is important for examination in the real life, then report it as an anomaly but remove it for accuracy purposes.
6. This functionality was done manually.
7. Once we have a particular level of accuracy and after many hit and trials , it is not improving then we start changing the ratio of training and testing data. We try several combinations and come up with a result which provides maximum efficiency.
8. In the end we calculate the confusion matrix and class wise accuracies for the data set which is left.
9. In order to determine various possibilities , other than intuition and hit and trial the following Code block was run with the code thousands of times in order to give the experimentation a little bit more accuracy. The below code block is a basic random function which generates combinations of input parameters to be ignored . It was really helpful.

```
parameters=pd.DataFrame(myData.iloc[:,:-1].select_dtypes(
                            include=['int64','float64']))
a=random.sample(parameters.columns.values.tolist(),6)
parameters.drop(a,axis=1,inplace = True)
```

K. Without removing any input parameters and considering all numerical ones, which includes even the silly input parameters like “ID”, the data was as follows:

The output of the accuracy section is:-

Accuracy:- 69.44245889921372 %

The output of the confusion matrix section is:-

Confusion Matrix :-

```
 [[ 830  488]
 [ 367 1113]]
```

The output of the class wise accuracies section is:-

Positive class accuracy is :- 75.20270270270271 %

Negative class accuracy is :- 62.9742033383915 %

ANALYSIS:-

The final accuracy achieved of the system is:-

Excluded members:-

```
[“ID”,"CollegeID","CollegeCityTier","CollegeState","12gra
duation","Degree"]
```

Accuracy:- 71.37240886347391 %

(Test to train ratio= 0.29)

This level of 71 percent accuracy is quite decent as we are able to predict 71 percent of the data correctly .

Confusion matrix :-

```
[[ 830  488]
 [ 367 1113]]
```

The number of correct outputs is 830 + 1113. (i.e number of True positives + False negatives). The number of incorrect outputs is 367 + 488.

Testing to training ratio :-

Currently in the code the ratio is 30:70. However a slight change of the ratio to 29:71 changes the accuracy from **71.37240886347391 % to 71.42353770260746 %**.

(Apart from the combination used , the above accuracy is also found using the below combination:-
['CivilEngg', 'Age', 'ID', 'CollegeID', 'TelecomEngg'])

Thus there is quite an increase in this accuracy, just by changing the ratio by 1 unit. The result was arrived at after many combinations like 40:60, 50:50, 20:80 and then checking with values around these ranges.

The percentage of the training data is naturally higher so that our system can have an increased learning dataset. Also every dataset can show a different variation with the ratios to different limits. For the given dataset and problem statement , 0.29 is a great ratio.

Various cases tried :-
Excluded data for the following:-
(The one submitted in the code)
`["ID","CollegeID","CollegeCityTier","CollegeState","12graduation","Degree"]`

(Testing to training Ratio)
40:60
Accuracy:- 70.68390325271059 %

20:80
Accuracy:- 69.82489055659788 %

35:65
Accuracy:- 70.82371054657429 %

10:90
Accuracy:- 68.8715953307393 %

One such example with ratio as 60:40 gave the following results:['extraversion', 'GraduationYear', 'Age', 'ID', 'Quant']

Accuracy = 72.04502814258912 %

(This is so far one of the highest received however this was not submitted in the code , this is because, in this the ratio being 60:40, the level of training is less. Which according to my interpretation should ideally be more , in order that the system learns more)

Confusion Matrix :-

```
[[502 242]
 [205 650]]
```

Positive class accuracy is :- 76.0233918128655 %

Negative class accuracy is :- 67.47311827956989 %