BrainDead

Problem Statement 1 : Analyze Placement Data

we are going to discuss how to predict the placement status of a student based on various student attributes using Logistic regression algorithm.

Placements hold great importance for students and educational institutions. It helps a student to build a strong foundation for the professional career ahead as well as a good placement record gives a competitive edge to a college/university in the education market.

This study focuses on a system that predicts if a student would be placed or not based on the student's qualifications, historical data, and experience. This predictor uses a machine-learning algorithm to give the result.

The algorithm used is logistic regression. Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable(or output), y, can take only discrete values for given set of features(or inputs), X. Talking about the dataset, it contains the secondary school percentage, higher secondary school percentage, degree percentage, degree, and work experience of students. After predicting the result its efficiency is also calculated based on the dataset. The dataset used here is in .csv format.

Below is the step-by-step Approach:

Step 1: Import the required modules.

```
# import modules

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

Step 2: Now to read the dataset that we are going to use for the analysis and then checking the dataset.

```
# reading the file

dataset = pd.read_csv('Placement_Data_Full_Class.csv')

dataset
```

Output-

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

215 rows × 15 columns

Step 3: Now we will drop the columns that are not needed.

```
# dropping the serial no and salary col

dataset = dataset.drop('sl_no', axis=1)

dataset = dataset.drop('salary', axis=1)
```

Step 4: Now before moving forward we need to pre-process and transform our data. For that, we will use astype() method on some columns and change the datatype to category.

```
# catgorising col for further labelling

dataset["gender"] = dataset["gender"].astype('category')

dataset["ssc_b"] = dataset["ssc_b"].astype('category')

dataset["hsc_b"] = dataset["hsc_b"].astype('category')

dataset["degree_t"] = dataset["degree_t"].astype('category')

dataset["workex"] = dataset["workex"].astype('category')

dataset["specialisation"] = dataset["specialisation"].astype('category')

dataset["status"] = dataset["status"].astype('category')
```

```
dataset["hsc_s"] = dataset["hsc_s"].astype('category')

dataset.dtypes
```

Output-

```
gender            category
ssc_p              float64
ssc_b             category
hsc_p              float64
hsc_b             category
hsc_s             category
degree_p           float64
degree_t          category
workex            category
etest_p            float64
specialisation    category
mba_p              float64
status            category
dtype: object
```

Step 5: Now we will apply codes on some of these columns to convert their text values to numerical values.

```
# labelling the columns

dataset["gender"] = dataset["gender"].cat.codes

dataset["ssc_b"] = dataset["ssc_b"].cat.codes

dataset["hsc_b"] = dataset["hsc_b"].cat.codes

dataset["degree_t"] = dataset["degree_t"].cat.codes

dataset["workex"] = dataset["workex"].cat.codes

dataset["specialisation"] = dataset["specialisation"].cat.codes

dataset["status"] = dataset["status"].cat.codes

dataset["hsc_s"] = dataset["hsc_s"].cat.codes


# display dataset
```

dataset

Output:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
0	1	67.00	1	91.00	1	1	58.00	2	0	55.0	1	58.80	1
1	1	79.33	0	78.33	1	2	77.48	2	1	86.5	0	66.28	1
2	1	65.00	0	68.00	0	0	64.00	0	0	75.0	0	57.80	1
3	1	56.00	0	52.00	0	2	52.00	2	0	66.0	1	59.43	0
4	1	85.80	0	73.60	0	1	73.30	0	0	96.8	0	55.50	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	1	80.60	1	82.00	1	1	77.60	0	0	91.0	0	74.49	1
211	1	58.00	1	60.00	1	2	72.00	2	0	74.0	0	53.62	1
212	1	67.00	1	67.00	1	1	73.00	0	1	59.0	0	69.72	1
213	0	74.00	1	66.00	1	1	58.00	0	0	70.0	1	60.23	1
214	1	62.00	0	58.00	1	2	53.00	0	0	89.0	1	60.22	0

Step 6: Now to split the dataset into features and values using iloc() function:

```
# selecting the features and labels
X = dataset.iloc[:, :-1].values
Y = dataset.iloc[:, -1].values

# display dependent variables
Y
```

Output:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
0	1	67.00	1	91.00	1	1	58.00	2	0	55.0	1	58.80	1
1	1	79.33	0	78.33	1	2	77.48	2	1	86.5	0	66.28	1
2	1	65.00	0	68.00	0	0	64.00	0	0	75.0	0	57.80	1
3	1	56.00	0	52.00	0	2	52.00	2	0	66.0	1	59.43	0
4	1	85.80	0	73.60	0	1	73.30	0	0	96.8	0	55.50	1

Step 7: Now we will split the dataset into train and test data which will be used to check the efficiency later.

```
# dividing the data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.2)
```

```
# display dataset
dataset.head()
```

Output:

	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status
0	1	67.00	1	91.00	1	1	58.00	2	0	55.0	1	58.80	1
1	1	79.33	0	78.33	1	2	77.48	2	1	86.5	0	66.28	1
2	1	65.00	0	68.00	0	0	64.00	0	0	75.0	0	57.80	1
3	1	56.00	0	52.00	0	2	52.00	2	0	66.0	1	59.43	0
4	1	85.80	0	73.60	0	1	73.30	0	0	96.8	0	55.50	1

Step 8: Now we need to train our model for which we will need to import a file, and then we will create a classifier using sklearn module. Then we will check the accuracy of the model.

```
# creating a classifier using sklearn
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(random_state=0, solver='lbfgs',
                         max_iter=1000).fit(X_train,
                                            Y_train)
# printing the acc
```

clf.score(X_test, Y_test)

Output:

```
0.9069767441860465
```

Step 9: Once we have trained the model, we will check it giving some random values:

predicting for random value

clf.predict([[0, 87, 0, 95, 0, 2, 78, 2, 0, 0, 1, 0]])

Output:

```
array([1], dtype=int8)
```

Step 10: To gain a more nuanced understanding of our model's performance we need to make a confusion matrix. A confusion matrix is a table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives.

To get the confusion matrix it takes in two arguments: The actual labels of your test set y_test and predicted labels. The predicted labels of the classifier are stored in y_pred as follows:

creating a Y_pred for test data

Y_pred = clf.predict(X_test)

display predicted values

Y_pred

Output:

```
array([1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1],
      dtype=int8)
```

Step 11: Finally, we have y_pred, so we can generate the confusion matrix:

```
# evaluation of the classifier

from sklearn.metrics import confusion_matrix, accuracy_score


# display confusion matrix

print(confusion_matrix(Y_test, Y_pred))


# display accuracy

print(accuracy_score(Y_test, Y_pred))
```

Output:

```
[[ 9  1]
 [ 3 30]]

0.9069767441860465
```