Logistic Regression with 'Adult' data

By Rustamova Malakkhanim | 604.19E

Instructions

Logistic regression is used in statistical software to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using a logistic regression equation. Logistic regression is easier to implement, interpret and very efficient to train.

Today, I am going to solve a classification problem with the help of scikit-learn and import some libraries for visualization of a confusion matrix, data pre-processing, ignoring warnings and classification.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

Then, I import adult data with the help of a pandas and do some modifications on it.

```
#uploading data to our code

data = pd.read_csv(r"C:\Users\user\Desktop\adult data.txt", sep=",", header=None)
```

```
data.head(3)
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	class
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K

```
data.columns
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14], dtype='int64')
```

```
#changing column names
data.columns=['age','workclass','fnlwgt','education','education-num','marital-status','occupation','relationship','race','sex','
```

Dropped some columns because they have numerical values, in classification problem we need only categoric variables.

```
#droping unwanted columns(unwanted-> columns with numerical values,because it's classification problem)
data.drop(data.columns[[0,2,4,6,10,11,12]], axis=1, inplace=True)
```

```
data
```

	workclass	education	marital-status	relationship	race	sex	native-country	class
0	State-gov	Bachelors	Never-married	Not-in-family	White	Male	United-States	<=50K
1	Self-emp-not-inc	Bachelors	Married-civ-spouse	Husband	White	Male	United-States	<=50K
2	Private	HS-grad	Divorced	Not-in-family	White	Male	United-States	<=50K
3	Private	11th	Married-civ-spouse	Husband	Black	Male	United-States	<=50K
4	Private	Bachelors	Married-civ-spouse	Wife	Black	Female	Cuba	<=50K
...	...	...	...	...	...	...	...	...
32556	Private	Assoc-acdm	Married-civ-spouse	Wife	White	Female	United-States	<=50K
32557	Private	HS-grad	Married-civ-spouse	Husband	White	Male	United-States	>50K
32558	Private	HS-grad	Widowed	Unmarried	White	Female	United-States	<=50K
32559	Private	HS-grad	Never-married	Own-child	White	Male	United-States	<=50K
32560	Self-emp-inc	HS-grad	Married-civ-spouse	Wife	White	Female	United-States	>50K

32561 rows × 8 columns

I create dummy table, generally 'get_dummies' function is used for data manipulation and it converts categorical data into dummy or indicator variables. Then, I dropped some columns like sex(because when sex is 1 it means person is man and woman-0), and columns with ? (blank) sign.

```
dummy = pd.get_dummies(data, columns =['workclass','education','marital-status','relationship','race','sex','native-country'])
```

```
dummy
```

	class	workclass_?	workclass_Federal-gov	workclass_Local-gov	workclass_Never-worked	workclass_Private	workclass_Self-emp-inc	workclass_Self-emp-not-inc	workclass_State-gov	workclass_Without-pay	...	native-country_Portugal	native-country_Puerto-Rico	native-country_Scotland
0	<=50K	0	0	0	0	0	0	0	1	0	...	0	0	0
1	<=50K	0	0	0	0	0	0	1	0	0	...	0	0	0
2	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
3	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
4	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32556	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
32557	>50K	0	0	0	0	1	0	0	0	0	...	0	0	0
32558	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
32559	<=50K	0	0	0	0	1	0	0	0	0	...	0	0	0
32560	>50K	0	0	0	0	0	1	0	0	0	...	0	0	0

32561 rows × 88 columns

```
dummy.drop(dummy.columns[[1,44,46]], axis=1, inplace=True)
```

I take Y as target value(class) and X as features.

```
: dummy
```

	class	workclass_ ?	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	workclass_ Private	workclass_ Self-emp-inc	workclass_ Self-emp-not-inc	workclass_ State-gov	workclass_ Without-pay	...	native-country_ Portugal	native country_ Puerto Ric
0	<=50K	0	0	0	0	0	0	0	1	0	...	0	
1	<=50K	0	0	0	0	0	0	1	0	0	...	0	
2	<=50K	0	0	0	0	1	0	0	0	0	...	0	
3	<=50K	0	0	0	0	1	0	0	0	0	...	0	
4	<=50K	0	0	0	0	1	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
32556	<=50K	0	0	0	0	1	0	0	0	0	...	0	
32557	>50K	0	0	0	0	1	0	0	0	0	...	0	
32558	<=50K	0	0	0	0	1	0	0	0	0	...	0	
32559	<=50K	0	0	0	0	1	0	0	0	0	...	0	
32560	>50K	0	0	0	0	0	1	0	0	0	...	0	

32561 rows × 88 columns

```
: X = dummy.iloc[:,1:]
```

```
: Y = dummy.iloc[:,0]
```

```
: Y.head()
```

I split data into 2 parts, train and test data with the help of 'train_test_split'. Predicted y is the predicted values of our target value(class).

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,random_state=0)
```

```
classifier = LogisticRegression()
```

```
classifier.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
predicted_y = classifier.predict(X_test)
```

```
predicted_y
```

```
array([' <=50K', ' <=50K', ' >50K', ..., ' <=50K', ' >50K', ' <=50K'],
      dtype=object)
```

And finally its accuracy will be 0.823… approximately 82%.

```
print("Accuracy:",metrics.accuracy_score(Y_test,  predicted_y))
```

```
Accuracy: 0.8238545633214592
```

Confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with it. It has TN, TP, FN, FP values.

```python
class_names=['>50k','<50k'] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="BuPu",fmt='g' )
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 257.44, 'Predicted label')
```