Logistic Regression

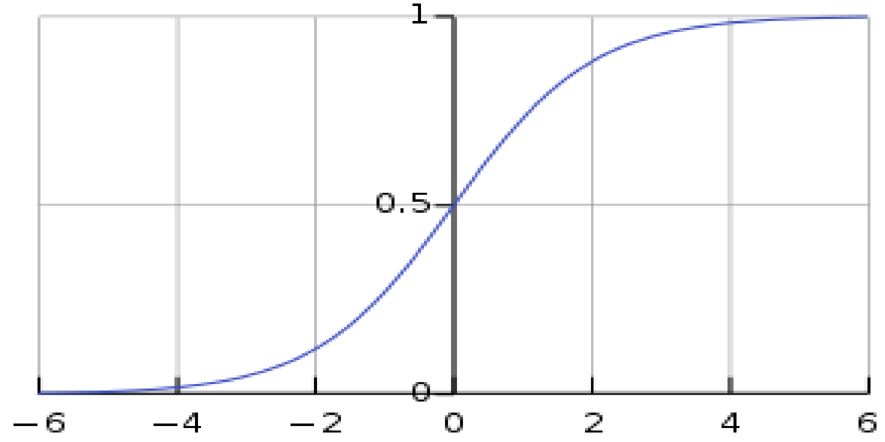
Logistic Regression

- This algorithm is used for classification type problems
- Types of Logistic Regression:
  - Binary
  - Multinomial
  - Ordinal
- We are going to cover Binary Logistic Regression

Logistic Response Function

- Standard logistic function on 2-dimensional plane is given by the following expression given on the right.
- From the graph, it is evident that the value of the f(x) ranges between 0 and 1.
- This function is also called **sigmoid** function and has a wide usage in various other algorithms such as neural network.

$$y = f(x) = \frac{1}{1 + e^{-x}}$$



Sane's STATS Academy of Statistics

Logistic Response Function

- The same function in the m-dimensional space can be written in the following way:

$$y = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}}$$

Where

$\beta_0, \beta_1, \beta_2, \ldots \beta_m$: Coefficients of the variables in m-dimensional space

- For any values of $\beta_0, \beta_1, \beta_2, \ldots \beta_m$ and $x_1, x_2, \ldots x_m$, the value of y always between 0 and 1.
- We can denote y by probability p.

Odds

$$p = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}}$$

$$1 - p = \frac{e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}}$$

$$\frac{p}{1-p} = e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}$$

- The ratio $\frac{p}{1-p}$ is called odds. For any values of $\beta_0, \beta_1, \beta_2, \dots \beta_m$ and $x_1, x_2, \dots x_m$, odds always ranges from 0 to ∞.

Sane's STATS Academy of Statistics

Interpreting Logistic Function

- In our binary classification, let us consider 0 and 1 as two possible outcomes, with 0 as non-occurrence of a particular event and 1 as occurrence of the particular event.
- p in our expression, is considered as probability of occurrence of the event and $1 - p$ as non-occurrence of the event
- Hence, the ratio $\frac{p}{1-p}$ is ratio of probability of occurrence to the probability of non-occurrence of the event.

Logit Function

$$\frac{p}{1-p} = e^{(\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_m x_m)}$$

$$\log(odds) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

- The ratio $\log\left(\frac{p}{1-p}\right)$ is called logit function. For any values of $\beta_0, \beta_1, \beta_2, \ldots \beta_m$ and $x_1, x_2, \ldots x_m$, log(odds) always range from -∞ to ∞.

Sane's STATS Academy of Statistics

Parameter Calculation

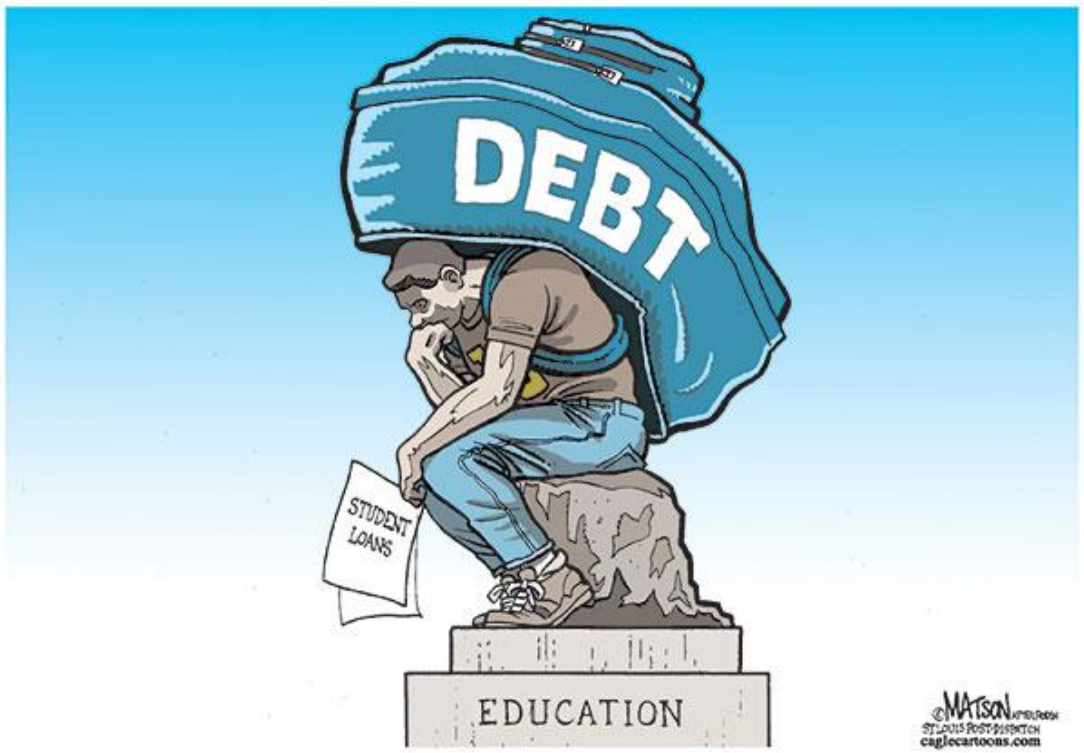
- Parameters $\beta_0, \beta_1, \beta_2, \ldots \beta_m$ are calculated with the help of maximum likelihood method.
- In Python, we make use of the function LogisticRegression() from sklearn.linear_model

Assumptions

- Logistic regression does **not** make many of the key assumptions of linear regression and general linear models - particularly regarding
  - Linearity
  - Normality
  - Homoscedasticity
- So we can apply logistic regression to any data for which we have categorical response and mixture of categorical and numerical predictors

Example: Loan Defaulter

- Consider the data on loan defaulters.
- For analysis, we have taken non-defaulter + defaulters
- Data has been recorded in the file Default.csv



Sane's STATS Academy of Statistics

Program & Output – With only categorical variable

```
## Consider Student variable
X=pd.DataFrame(dum_Default['student_Yes'])
y = dum_Default.iloc[:,2]

# Import the necessary modules
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

logreg = LogisticRegression()

# Fit the classifier to the training data
logreg.fit(X,y)

### Predicting categorical values
trialvals = np.array([0,1])
X_test = pd.DataFrame(trialvals)
logreg.predict_proba(X_test)
```

```
In [181]: logreg.coef_
Out[181]: array([[0.38256903]])

In [182]: logreg.intercept_
Out[182]: array([-3.48496241])
```

- The equation in this case, the following will be the equation,

$$P(\text{"Default"}) = \frac{1}{1+e^{-3.48496241+0.38256903*student\_Yes}}$$

Program & Output – With only numeric variable

```
In [183]: X=pd.DataFrame(dum_Default['income'])
     ...: y = dum_Default.iloc[:,2]
     ...:
     ...: # Import the necessary modules
     ...:
     ...: logreg = LogisticRegression()
     ...:
     ...: # Fit the classifier to the training data
     ...: logreg.fit(X,y)
     ...:
     ...: ### Predicting numeric values
     ...: trialvals = np.array([10000,30000,50000,70000])
     ...: X_test = pd.DataFrame(trialvals)
     ...: logreg.predict_proba(X_test)
Out[183]:
array([[7.49814563e-01, 2.50185437e-01],
       [9.64183419e-01, 3.58165812e-02],
       [9.95881411e-01, 4.11858858e-03],
       [9.99539789e-01, 4.60211203e-04]])
```

```
In [185]: logreg.coef_
Out[185]: array([[-0.00010976]])

In [186]: logreg.intercept_
Out[186]: array([-6.49168482e-09])
```

- The equation in this case, the following will be the equation,

$$P(\text{"Default"}) = \frac{1}{1+e^{-6.492e-9-0.000109*Income}}$$

Logistic Regression with Supervised

```
In [185]: logreg.coef_
Out[185]: array([[-0.00010976]])

In [186]: logreg.intercept_
Out[186]: array([-6.49168482e-09])

In [187]: X = dum_Default.iloc[:,[0,1,3]]
     ...: y = dum_Default.iloc[:,2]
     ...: # Create training and test sets
     ...: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4,
     ...:                                                     random_state=42)
     ...:
     ...: # Create the classifier: logreg
     ...: logreg = LogisticRegression()
     ...: # Fit the classifier to the training data
     ...: logreg.fit(X_train,y_train)
     ...: # Predict the labels of the test set: y_pred
     ...: y_pred = logreg.predict(X_test)
```

```
In [189]: print(confusion_matrix(y_test, y_pred))
[[3862    3]
 [ 135    0]]

In [190]: print(classification_report(y_test, y_pred))
             precision    recall  f1-score   support

          0       0.97      1.00      0.98      3865
          1       0.00      0.00      0.00       135

avg / total       0.93      0.97      0.95      4000
```

Questions?