Data Mining and Machine Learning

Lecture 11; Logistic Regression

Dr. Edgar Acuna
Departamento de Matematicas

Universidad de Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

Introduction

For now, we will consider that our dataset has only two clases.

The total number of instances is given by $n=n_1+n_2$, where n_1 instance belong to class C_1 and n_2 instances are from class C_2.

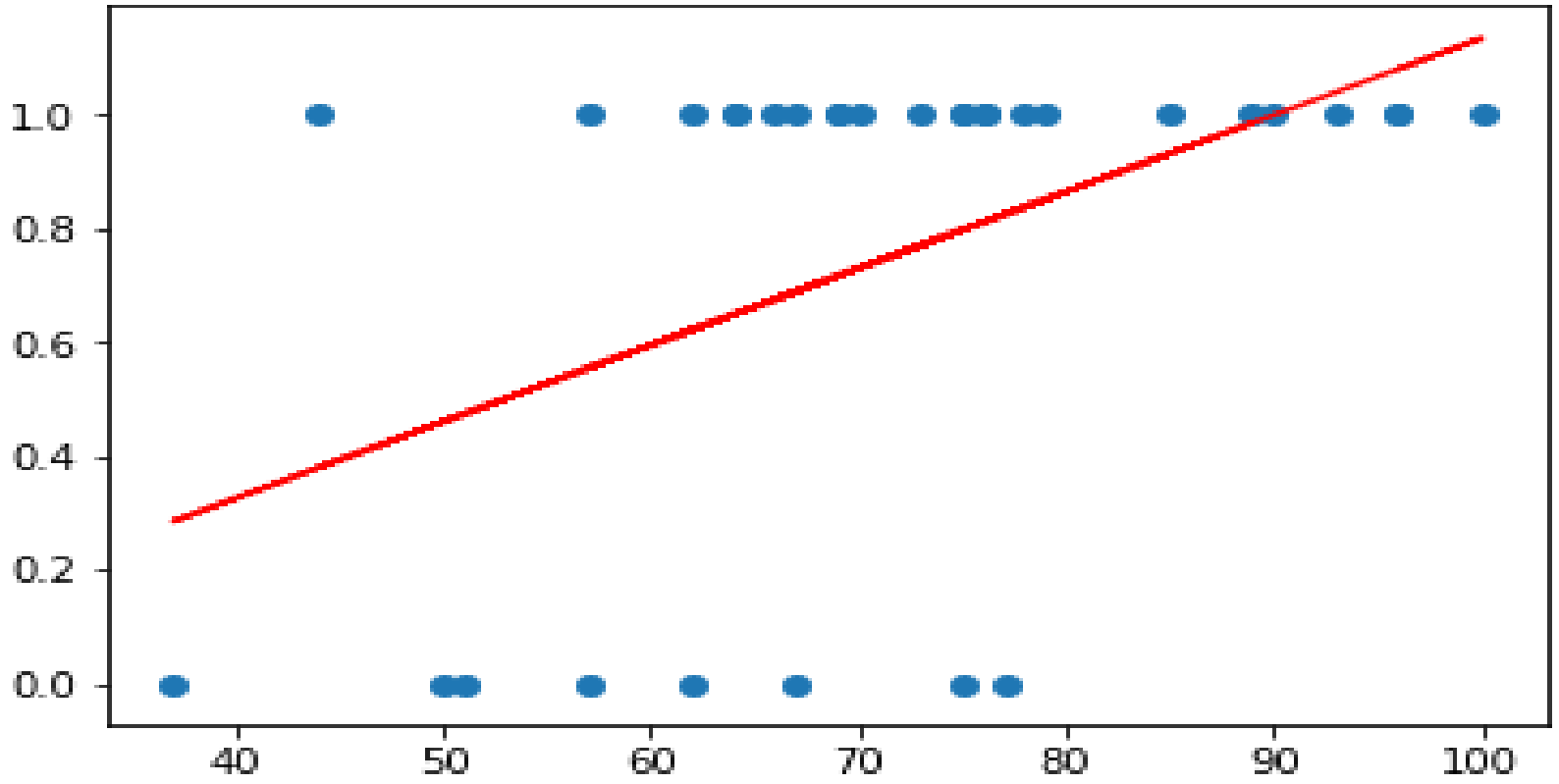
For each instance $\mathbf{x}_j$ a binary variable Y has value 1 if $\mathbf{x}_j$ belongs to class C1, otherwise its value is 0.

The apriori probability (relative frequency) that Y=1 is denoted by π_1.

EXAMPLE: PREDICTION OF FINAL GRADE BASED ONLY ON EX1

```
df=pd.read_csv("c://PW-PR/eje1dis.csv")
#Convirtiendo en matriz la tabla de predictoras y la columna de clases
y=df['Nota']
X=df.iloc[:,0:2]
#creando una columna "pass" numerica para representar las clases
lb_make = LabelEncoder()
df["pass"] = lb_make.fit_transform(df["Nota"])
databin=df[['E1','pass']]
print databin.head()
    E1 pass
0  96  1
1  96  1
2 100 1
3  93  1
4  90  1
```

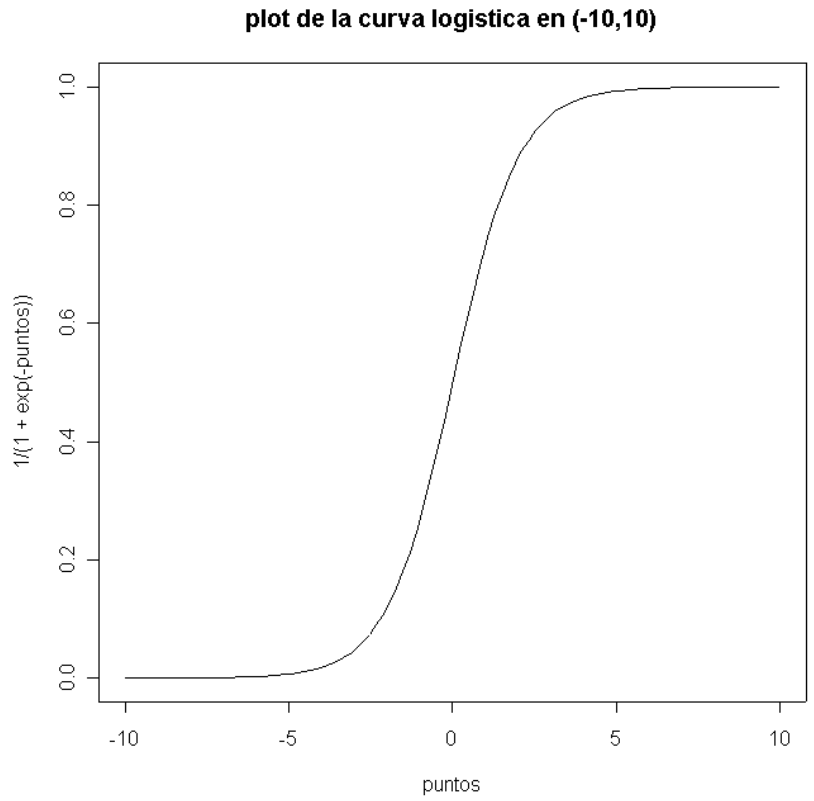
Regression Line for a binary response



The Logistic Curve

Logistic distribution function

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



Derivation of the Logistic regression model

In the logistic model (Berkson, 1944) it is assumed that

$$\log\left(\frac{f(x/C_1)}{f(x/C_2)}\right) = \alpha + \beta' x$$

where **x** is a p-dimensional vector of predictor variables, α is a constant and β is a vector of p parameters. The Multivariate Normal with equal covariances and the multinomial distribution satisfy the above condition.

Let p=P(Y=1/x) be the posterior probability that a object with observed measurements x belong to class Y=1. Then,

$$\frac{p}{1-p} = \frac{\frac{\pi_1 f(x/C_1)}{f(x)}}{\frac{\pi_2 f(x/C_2)}{f(x)}} = \frac{\pi_1 f(x/C_1)}{\pi_2 f(x/C_2)} \qquad (1)$$

Derivation of the Logistic regression model (cont)

where $\frac{p}{1-p}$

is called the ***odds ratio***. Taking logaritmos in both sides of the equation , we get

$$\log(\frac{p}{1-p}) = \log(\frac{\pi_1}{\pi_2}) + \log\frac{f(\mathbf{x}/C_1)}{f(\mathbf{x}/C_2)} \qquad (2)$$

Analisis discriminante lineal como un clasificador Bayesiano

For the Multivariate Normal distribution which is given by

$$\frac{1}{|\Sigma|^{1/2}(2\pi)^{p/2}}\exp(-1/2[(x-\mu)'\sum^{-1}(x-\mu)])$$

The equation 2 can be written as

$$\log(\frac{p}{1-p}) = \log(\frac{\pi_1}{\pi_2}) + \log\frac{\exp[-1/2(\mathbf{x}-\mathbf{u_1})\prime\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{u_1})}{\exp[-1/2(\mathbf{x}-\mathbf{u_2})\prime\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{u_2})}$$

Taking logarithms in both sides of the last equation we obtain

$$\log(\frac{p}{1-p}) = \log(\frac{\pi_1}{\pi_2}) - 1/2[(\mathbf{x}-\mathbf{u}_1)'\Sigma^{-1}(\mathbf{x}-\mathbf{u}_1) - (\mathbf{x}-\mathbf{u}_2)'\Sigma^{-1}(x-\mathbf{u}_2)]$$

Derivation of the Logistic regression model (cont)

The equation (2) can be written

$$\log\left(\frac{p}{1-p}\right) = \alpha + \beta' \mathbf{x}$$

or equivalently

$$p = \frac{\exp(\alpha + \beta' \mathbf{x})}{1 + \exp(\alpha + \beta' \mathbf{x})}$$

The last equation is called logistic regression, and

$$\log\left(\frac{p}{1-p}\right)$$

is called the ***logit*** transformation

Derivation of the Logistic regression model (cont)

The estimates $\tilde{\alpha}$ and $\tilde{\beta}$ are those that maximize the likelihood function (or its logarithm) corresponding to the model. For a sample of size 1 is given by

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)$$

For a sample of size n, the estimators are obtained by maximizing

$$L(\alpha, \beta) = \prod_{i=1}^{n_1} \frac{\exp(a + \mathbf{x_i}'\boldsymbol{\beta})}{1 + \exp(\alpha + \mathbf{x_i}'\boldsymbol{\beta})} \cdot \prod_{j=n_1+1}^{n} \frac{1}{1 + \exp(\alpha + \mathbf{x_j}'\boldsymbol{\beta})}$$

The solution of this maximization problem is found iteratively using methods such as Newton-Raphson, iterative reweighed least squares minimos cuadrados,etc . However, sometimes there are problems of convergence generated when there are separability between the clases.

Logistic Regression for classification

For classification purposes, the easiest way to discriminate between the two classes is to considerar that when $p>0.5$ the instance belongs to the class of interest. But many times, this discrimination is unfair, specially when one class is much less frequent that the other one.

Other alternative methods are:

A) Plot in the same graph, the percentage of instances having the event and that are classifed correctly(Sensitivity) versus different levels of probability, and the percentage of instances of the other class that are classfied correctly (specifity) versus the same levels of probability used before. The probability to be used to discriminate between the two clases is obtained intersecting both curves.

Logistic Regression for classification

B)Use the ROC (receiver operating characteristic) curve. In this case, one has to plot the sensitivity versus (1-specifity)100%, for several values of p. The optimal p is the one that is closer to the upper left corner which is (0,100).

In Python there are two main libraries to perform logistic regression: scikit-learn and h2o.

Example: Predicting final grade based on E1 and E2

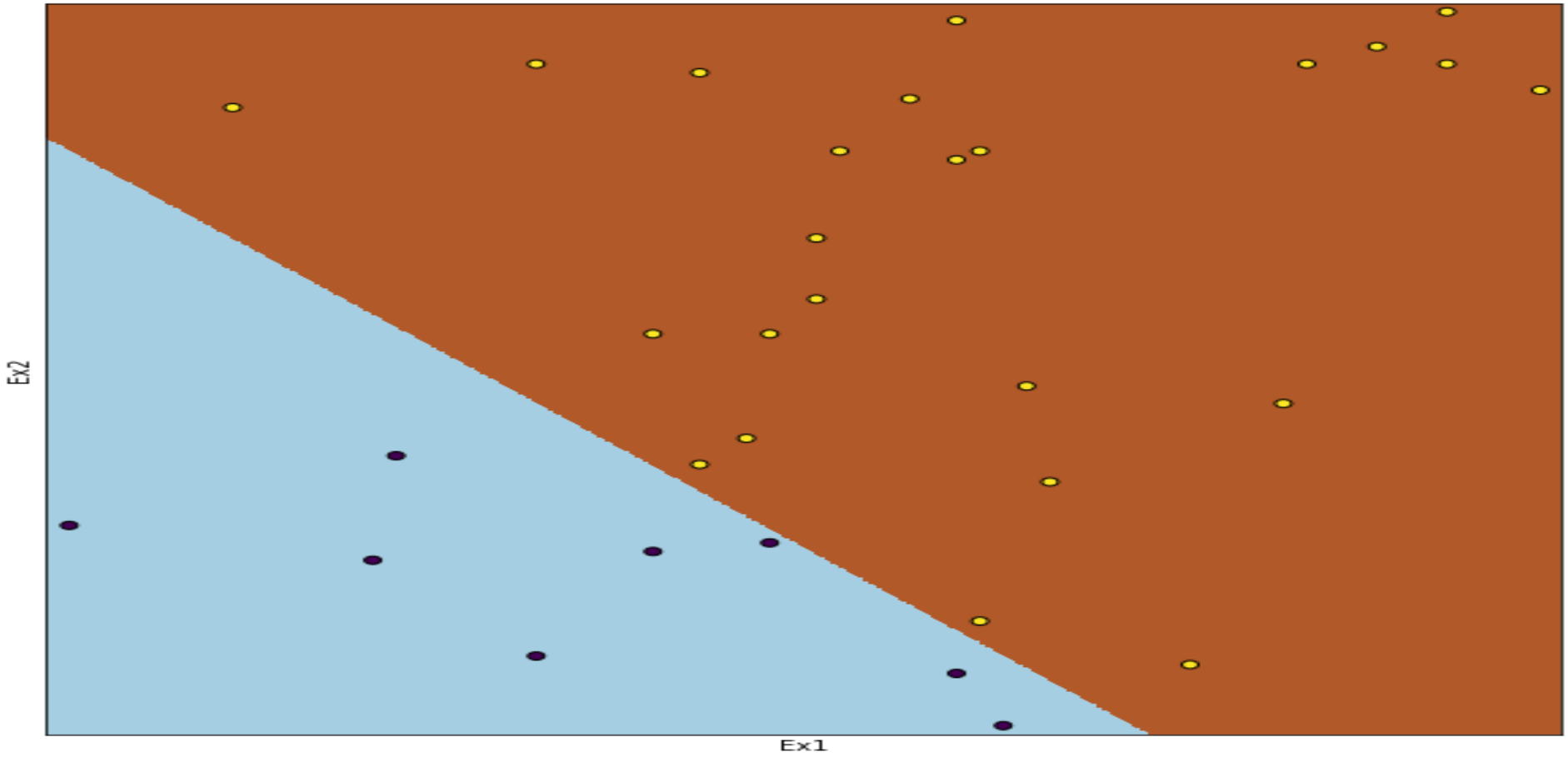
```
df=pd.read_csv("c://PW-PR/eje1dis.csv")
y=df['Nota']
X=df.iloc[:,0:2]
#creating a numerical column  "pass“ to represent classes
lb_make = LabelEncoder()
df["pass"] = lb_make.fit_transform(df["Nota"])
X=df[['E1',"E2"]]
y3=df['pass’]
#Applying the logistic regression and measuring its accuracy
model = LogisticRegression(solver="newton-cg")
model = model.fit(X, y3)
print “Model coefficents", model.coef_
Model coefficients
[[ 0.84415919 0.5817652 ]]
```

Example: Predicting final grade based on E1 and E2 (cont)

```
#Accuracy
model.score(X,y3)
1.0
pred = model.predict(X)
print(pred)
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0]
#Computing the posterior probabilities
pred1=model.predict_proba(X)
print(pred1[0:5,:])
[[ 0. 1.]
 [ 0. 1.]
 [ 0. 1.]
 [ 0. 1.]
  [ 0. 1.]]
```

There is a probability 1 that the first 24 students pass the class and a probability 1 that the last 8 students fail the class. There is a total separability between the classes.

Decision Line for the logistic regression classifier



Example2. Diabetes

```
#The response  variable must be binary (0,1)
y=data['class']-1
X=data.iloc[:,0:8]
#Performing the logistic regression and calculating its accuracy
model = LogisticRegression()
model = model.fit(X, y)
print model.coef_
[[ 1.17057358e-01 2.83881555e-02 -1.68873034e-02 7.55876596e-04 -6.42924257e-04
    5.97710296e-02 6.77534932e-01 7.24147639e-03]]
# Accuracy rate
model.score(X, y)
0.7747395833333337
```

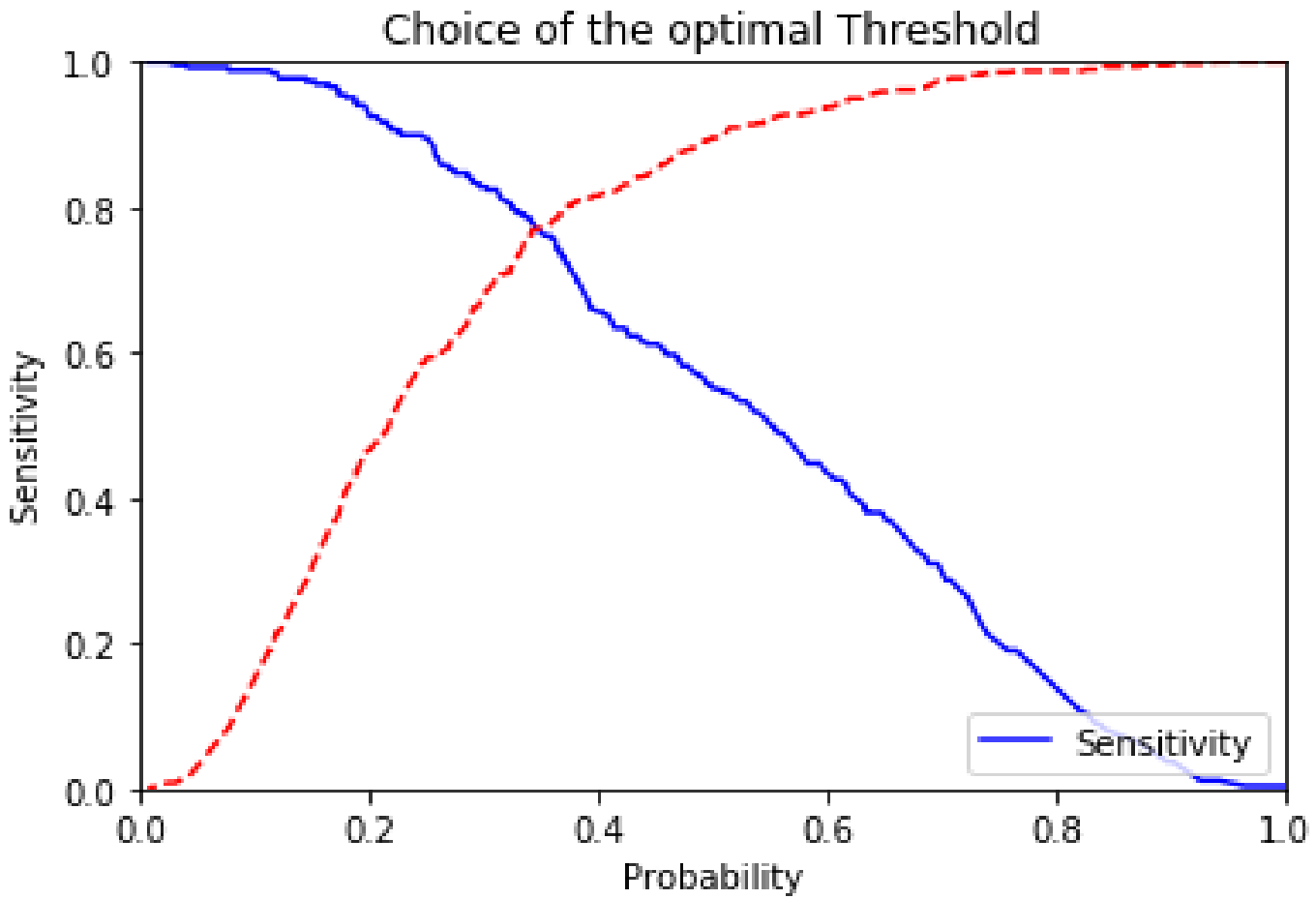
Example: Diabetes [2]

```
#Using the h2o library
diabetes = h2o.import_file("https://academic.uprm.edu/eacuna/diabetes.dat")
myx=['C1','C2','C3','C4','C5','C6','C7','C8']
diabetes['C9']=diabetes['C9'].asfactor()
myy='C9'
glm_model = H2OGeneralizedLinearEstimator(family= "binomial", lambda_ = 0, compute_p_values = True)
glm_model.train(myx, myy, training_frame= diabetes)
y_pred=glm_model.predict(diabetes)
print (y_pred['predict']==diabetes['C9']).sum()/float(len(diabetes))
0.766927083333
```

Example: Diabetes[3]

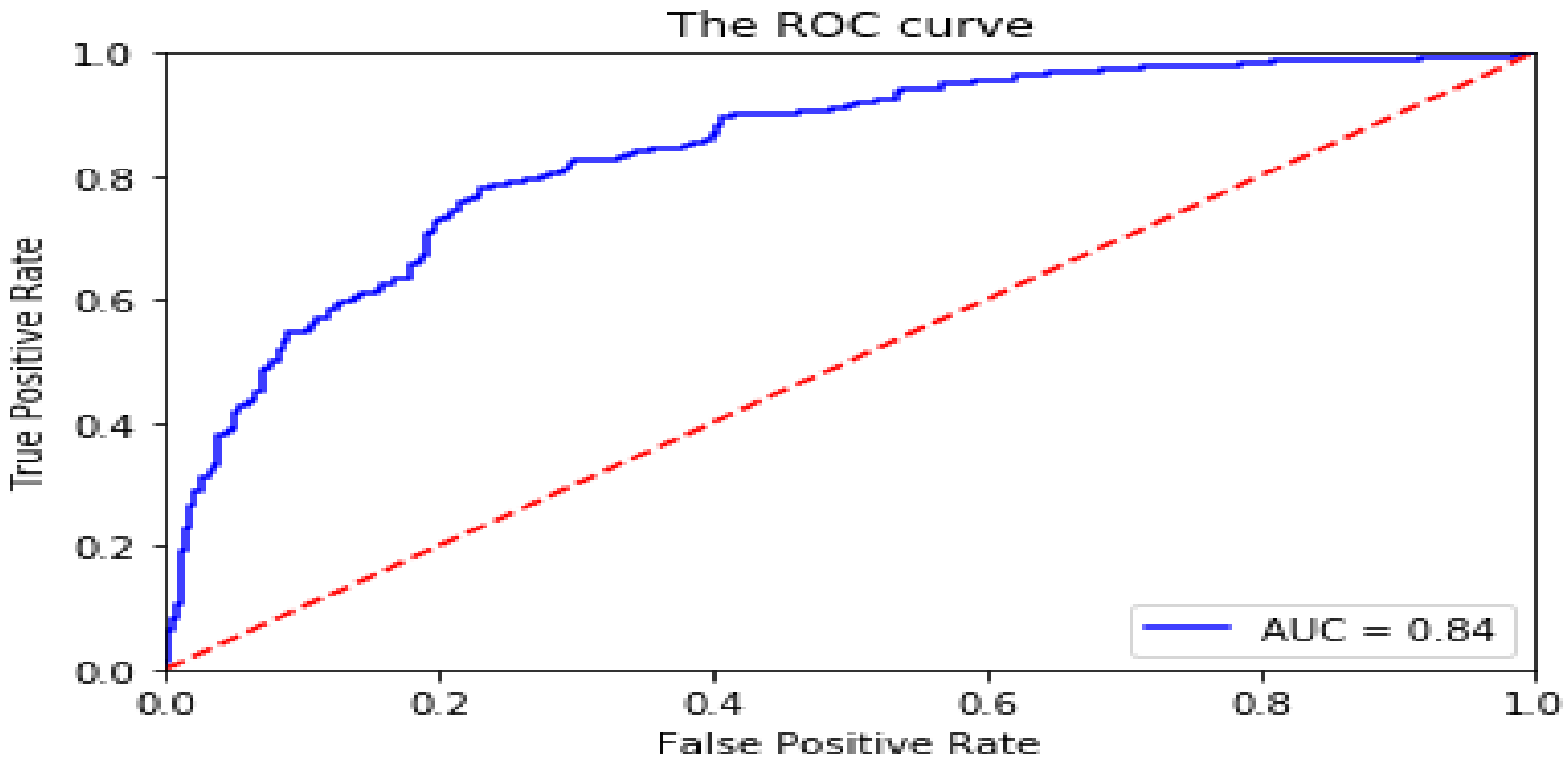
```
#Finding the posterior probabilities
probs = model.predict_proba(X)
preds = probs[:,1]
false_positive_rate, true_positive_rate, thresholds = roc_curve(y, preds)
roc_auc = auc(false_positive_rate, true_positive_rate)> print(tabla)
plt.title('Choice of the optimal Threshold')
plt.plot(thresholds, true_positive_rate, 'b',label='Sensitivity')
plt.legend(loc='lower right')
plt.plot(thresholds, 1-false_positive_rate,'r--')
plt.xlim([0,1.0])
plt.ylim([0,1.0])
plt.ylabel('Sensitivity ')
plt.xlabel('Probability')
```

Example. Diabetes[4]



The optimal threshold is about .35

Example: Diabetes [5]



Example: Diabetes [6]

#Effect after using the threshold
#Number of instances assigned to class 1 using p=.5
dp=data[preds>.5]
dp['class'].value_counts()
2 147
1 52
#Accuracy with p=.5
595*100/768.0=77.47395833333333
#Number of instances assigned to class 1 using p=.35
dp1=data[preds>.35]
dp1['class'].value_counts()
2 205
1 114
#New accuracy
591*100/768.0=76.953125

Multiclass logistic regression

It was discussed in detail by Engel (1988) and it also known as Polytomous (Multinomial) Logistic Regression. When there are K classes then the odds ratios are taken with respect to one class, say the last class. Thus,

$\text{Log}[P(Y=j/x)/P(Y=K/x)]=\alpha_j+\beta'_j\mathbf{x}$ para j=1,2,….K-1

From the above, it follows ,

$$\sum_{j=1}^{K-1} P(Y=j/x) = P(Y=K/x)\exp(\alpha j+\beta'_j x)$$

and using the fact that

$$\sum_{j=1}^{K} P[Y=j/\mathrm{x}] = 1$$

Multiclass logistic regression

It is obtained that

$$p_j = P[Y = j / \mathrm{x}] = \frac{\exp(\alpha_j + \beta_j' \mathrm{x})}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \beta_j' \mathrm{x})}$$

for j=1,2,…K-1, and,

$$p_K = P[Y = K / \mathrm{x}] = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\alpha_j + \beta_j' \mathrm{x})}$$

In the area of neural networks these functions are called softmax. The object x is assigned to class j* such that j*=argmax(p_j).

The library h2o uses the option family="multinomal' to perform logistics regression with more than two classes.

Example: Vehicle

```
vehicle = h2o.import_file("https://academic.uprm.edu/eacuna/vehicle.csv")
predictors=['V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V15','V16','V17','V
    18']
vehicle['class']=vehicle['class'].asfactor()
response_col="class"
glm_model = H2OGeneralizedLinearEstimator(family= "multinomial", lambda_ = 0)
glm_model.train(predictors, response_col, training_frame= vehicle)
glm_model
glm_model.confusion_matrix(vehicle)
```

Example: Vehicle[2]

Confusion Matrix: Row labels: Actual class; Column labels: Predicted class

1	2	3	4	Error	Rate
214.0	1.0	2.0	1.0	0.0183486	4 / 218
1.0	142.0	67.0	2.0	0.3301887	70 / 212
4.0	57.0	154.0	2.0	0.2903226	63 / 217
1.0	2.0	0.0	196.0	0.0150754	3 / 199
220.0	202.0	223.0	201.0	0.1654846	140 / 846

Example: Logistic applied to dataset with mixed type attributes

Sexo	Familia	CasPropia	AnosEmpleo	Sueldo	StatustMarital	Prestamo
Hombre	3	No	17	2500	Soltero	No
Mujer	5	Si	10	3000	Casado	Si
Mujer	4	No	15	2000	Viudo	No
Hombre	3	Si	16	2800	Soltero	Si
Hombre	6	Si	11	4000	Viudo	Si
Mujer	4	Si	26	3200	Soltero	Si
Mujer	2	Si	14	1800	Soltero	No
Hombre	5	Si	10	3750	Casado	Si
Hombre	6	No	18	2970	Divorciado	No
Hombre	4	Si	12	3350	Divorciado	No
Hombre	1	No	23	1950	Soltero	No
Mujer	2	Si	25	2740	Soltero	Si
Mujer	3	No	7	3100	Soltero	Si
Hombre	5	Si	5	3845	Divorciado	No
Hombre	3	No	13	3200	Casado	Si
Mujer	3	Si	9	2800	Soltero	No
Hombre	2	No	6	3200	Soltero	Si
Hombre	3	Si	7	3815	Viudo	Si
Mujer	2	Si	11	2980	Divorciado	No
Hombre	4	Si	15	2850	Viudo	Si
Mujer	1	No	6	3125	Divorciado	No
Hombre	1	No	8	3500	Soltero	Si
Hombre	4	No	22	4500	Divorciado	Si
Hombre	2	Si	10	3200	Casado	Si
Hombre	3	Si	9	3000	Casado	Si

Example: Logistic applied to dataset with mixed type attributes

```
 loan['Sexo'] = loan['Sexo'].asfactor()
loan["CasPropia"]=loan['CasPropia'].asfactor()
loan['StatustMarital']=loan['StatustMarital'].asfactor()
loan['Prestamo']=loan['Prestamo'].asfactor()
predictors = ["Sexo","Familia", "CasPropia","AnosEmpleo", "Sueldo","StatustMarital"]
response_col = "Prestamo“
glm_model = H2OGeneralizedLinearEstimator(family= "binomial", lambda_ = 0, compute_p_values =
True)
glm_model.train(predictors, response_col, training_frame= loan)
print(glm_model.coef())
{u'Familia': -7.283161660798802, u'CasPropia.Si': -1.473452885828509, u'AnosEmpleo':
2.957445208267 4885, u'Sueldo': 0.09868220640412526, u'Intercept': -269.16800091477114,
u'Sexo.Mujer': -4.160890209638011, u'StatustMarital.Soltero': -18.71709764395 0506,
u'StatustMarital.Divorciado': -102.24457434194633, u'StatustMarital.Viudo': -10.697821309701698}
glm_model
```

Nota: If we use three dummay variables to represent Marital Status we get same results.

Example: Logistic applied to dataset with mixed type attributes

```
y_pred=glm_model.predict(loan)
print (y_pred['predict']==loan['Prestamo']).sum()/float(len(loan))
1.0
#To obtain the ROC curve
perf = glm_model.model_performance()  #train=True is the default, so it's not needed
perf.plot()
```

If the variable Marital Status is transformed into a single numerical attribute then the performance of the classifiers deteriorates.

Comparing Logistic regression and LDA

If the normality assumption holds the the LDA classifier beats Logistic Regression. This can be seen when the dataset Iris is used.

Logistic Regression is somehow related to the Support vector Machine (SVM) classifier.