A diagram of a logistic regression

Description automatically generated A screenshot of a survey

Description automatically generated

A screenshot of a computer program

Description automatically generated A screenshot of a white text

Description automatically generated

A screenshot of a graph

Description automatically generated

A graph of a function

Description automatically generated

A white paper with black text and numbers

Description automatically generated

A white paper with blue text and red text

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Read dataset :

churn\_df = pd.read\_csv("ChurnData.csv")

churn\_df.head()

1. Preprocessing
   1. select some features for the modeling

churn\_df = churn\_df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip', 'callcard', 'wireless','churn']]

(200, 10)

* 1. change the target data type to be an integer (requirement library scikit learn)

churn\_df['churn'] = churn\_df['churn'].astype('int')

1. Let's define X, and y for our dataset:

X = np.asarray(churn\_df[['tenure', 'age', 'address', 'income', 'ed', 'employ', 'equip']])

(200, 7)

y = np.asarray(churn\_df['churn'])

1. normalize the dataset using standardScaler

from sklearn import preprocessing

X = preprocessing.StandardScaler().fit(X).transform(X)

1. Split dataset into TRAIN and TESTset

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, random\_state=4)

print ('Train set:', X\_train.shape, y\_train.shape)

print ('Test set:', X\_test.shape, y\_test.shape)

Train set: (160, 7) (160,)

Test set: (40, 7) (40,)

From 200 break into 80% Train / 20% Test

1. build our model using LogisticRegression from the Scikit-learn

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix

LR = LogisticRegression(C=0.01, solver='liblinear').fit(X\_train,y\_train)

LR

# **C** parameter indicates **inverse of regularization strength**

1. Predict using test set

yhat = LR.predict(X\_test)

yhat

1. predict\_proba returns estimates for all classes, ordered by the label of classes

yhat\_prob = LR.predict\_proba(X\_test)

yhat\_prob

1. Evaluation

from sklearn.metrics import jaccard\_score

jaccard\_score(y\_test, yhat,pos\_label=0)

Using confusion matrix

# Compute confusion matrix

cnf\_matrix = confusion\_matrix(y\_test, yhat, labels=[1,0])

np.set\_printoptions(precision=2)

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=['churn=1','churn=0'],normalize= False, title='Confusion matrix')

A diagram of a confused matrix

Description automatically generated

1. Print Classification Report

print (classification\_report(y\_test, yhat))

A screenshot of a graph

Description automatically generated

A white background with black text

Description automatically generated

1. Log Loss for evaluation

from sklearn.metrics import log\_loss

log\_loss(y\_test, yhat\_prob)

**SUPPORT VECTOR MACHINEs (SVM)**

A supervised algotirhm that classifies cases by finding a separator.

* Mapping data to a High-dimensional feature space
* Finding a separator (Line/Hyperplane)

A graph with red line and blue dots

Description automatically generatednon separable

A diagram of a function

Description automatically generated

A screenshot of a computer

Description automatically generated

A white background with blue text

Description automatically generated

LAB :

1. Load datasets

#Click here and press Shift+Enter

!wget -O cell\_samples.csv <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/cell_samples.csv>

cell\_df = pd.read\_csv("cell\_samples.csv")

cell\_df # 699 rows × 11 columns

The Class field contains the diagnosis, as confirmed by separate medical procedures, as to whether the samples are benign (value = 2) or malignant (value = 4).

1. look at the distribution of the classes based on Clump thickness and Uniformity of cell size

ax = cell\_df[cell\_df['Class'] == 4][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='DarkBlue', label='malignant');

cell\_df[cell\_df['Class'] == 2][0:50].plot(kind='scatter', x='Clump', y='UnifSize', color='Yellow', label='benign', ax=ax);

plt.show()

1. Preprocessing data
2. Let's first look at columns data types:

Drop non numerical data types :

cell\_df.dtypes

cell\_df = cell\_df[pd.to\_numeric(cell\_df['BareNuc'], errors='coerce').notnull()]

cell\_df['BareNuc'] = cell\_df['BareNuc'].astype('int')

cell\_df.dtypes

1. Change to array feature datasets#´

feature\_df = cell\_df[['Clump', 'UnifSize', 'UnifShape', 'MargAdh', 'SingEpiSize', 'BareNuc', 'BlandChrom', 'NormNucl', 'Mit']]

X = np.asarray(feature\_df)

X[0:5]

1. We want the model to predict the value of Class (that is, benign (=2) or malignant (=4)).

y = np.asarray(cell\_df['Class'])

y [0:5]

1. Split our dataet into train and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.2, random\_state=4)

print ('Train set:', X\_train.shape, y\_train.shape)

print ('Test set:', X\_test.shape, y\_test.shape)

From 699 rows × 11 columns

* Now 683 rows x 10 columns
* Train set: (546, 9) (546,)
* Test set: (137, 9) (137,)

1. Modeling SVM (Scikit Learn)

from sklearn import svm

clf = svm.SVC(kernel='rbf') **#RBF kernel**

clf.fit(X\_train, y\_train)

1. After being fitted, can be used to predict new values :

yhat = clf.predict(X\_test)

yhat [0:5]

1. Evaluation

# Compute confusion matrix

cnf\_matrix = confusion\_matrix(y\_test, yhat, labels=[2,4])

np.set\_printoptions(precision=2)

print (classification\_report(y\_test, yhat))

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(cnf\_matrix, classes=['Benign(2)','Malignant(4)'],normalize= False, title='Confusion matrix')

A screenshot of a computer screen

Description automatically generated

A diagram of a confusion matrix

Description automatically generated

1. Evaluation using F1 Score

from sklearn.metrics import f1\_score

f1\_score(y\_test, yhat, average='weighted')

1. Evaluation using accuracy

from sklearn.metrics import jaccard\_score

jaccard\_score(y\_test, yhat,pos\_label=2)

1. Kalau menggunakan Linear Kernel

clf2 = svm.SVC(kernel='linear')

clf2.fit(X\_train, y\_train)

yhat2 = clf2.predict(X\_test)

print("Avg F1-score: %.4f" % f1\_score(y\_test, yhat2, average='weighted'))

print("Jaccard score: %.4f" % jaccard\_score(y\_test, yhat2,pos\_label=2))

**Multiclass Prediction**

**SoftMax Regression**

SoftMax regression is similar to logistic regression, the SoftMax function converts the actual distances i.e. dot products of 𝑥with each of the parameters 𝜃 𝑖 θ i ​ for 𝐾 classes in the range from 0 to 𝐾-1.

A screenshot of a math test

Description automatically generated

A diagram of a graph

Description automatically generated with medium confidence

For each class we take the class samples we would like to classify, and the rest will be labeled as a “dummy” class. For example, to build a classifier for the blue class we simply assign all other labels that are not in the blue class to the Dummy class, we then train the classifier accordingly. The result is shown in fig 3 where the classifier predicts blue 𝑦 ^ = 0 y ^ ​ =0 and in the purple region where we have our “dummy class” 𝑦 ^ = 𝑑 𝑢 𝑚 𝑚 𝑦 y ^ ​ =dummy.

A diagram of a math problem

Description automatically generated

A diagram of a class

Description automatically generated with medium confidence

A white paper with black text

Description automatically generated

A screenshot of a white paper with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Final Assignment**

A white paper with black text

Description automatically generated

A white background with black text

Description automatically generatedA white text on a white background

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated