9. CLASSIFICATION 2: LOGISTIC REGRESSION, METRICS, TRANSFORMATIONS,

LEARNING OBJECTIVES

- Why not use a simple linear regression to predict classification?
- Binomial Logistic Regression
- Transformations in Scikit
- Recap on encoding categorical values

REVIEW OF LESSON 8

LAST LESSON REVIEW

- Classification
- KNN
- Optimizing meta parameters with Grid search
- Visualization of higher dimensions
- Curse of dimensionality

LAST SESSION

ANY QUESTIONS?

- 3 types of classifications?
- What's KNN? What does the K stands for?
- How to visualize Higher Dimensions?
- The Curse of Dimensionality

LOGISTIC REGRESSION

MORE ON CLASSIFICATION VS REGRESSION

Why not use linear regression to predict some medical condition such as

- 0: Stroke,
- 1: Epileptic seizure
- 2: Overdose

MORE ON CLASSIFICATION VS REGRESSION

Why not use linear regression to predict some medical condition such as

- 0: Stroke,
- 1: Epileptic seizure
- 2: Overdose

Encoding it like that and using Linear Regression implies:

- order of the encoding
- equal distance between codes

IN THE BINARY CASE:

- 0: Stroke,
- 1: Epileptic seizure

Possible to use linear regression as a proxy for a probability

- May end up with results outside the [0,1] range

So classification specific models better!

THE DEFAULT DATASET

Predict whether a person will default on his/her credit card payment

The data:

- Annual Income
- Monthly credit card balance
- is the person a student?

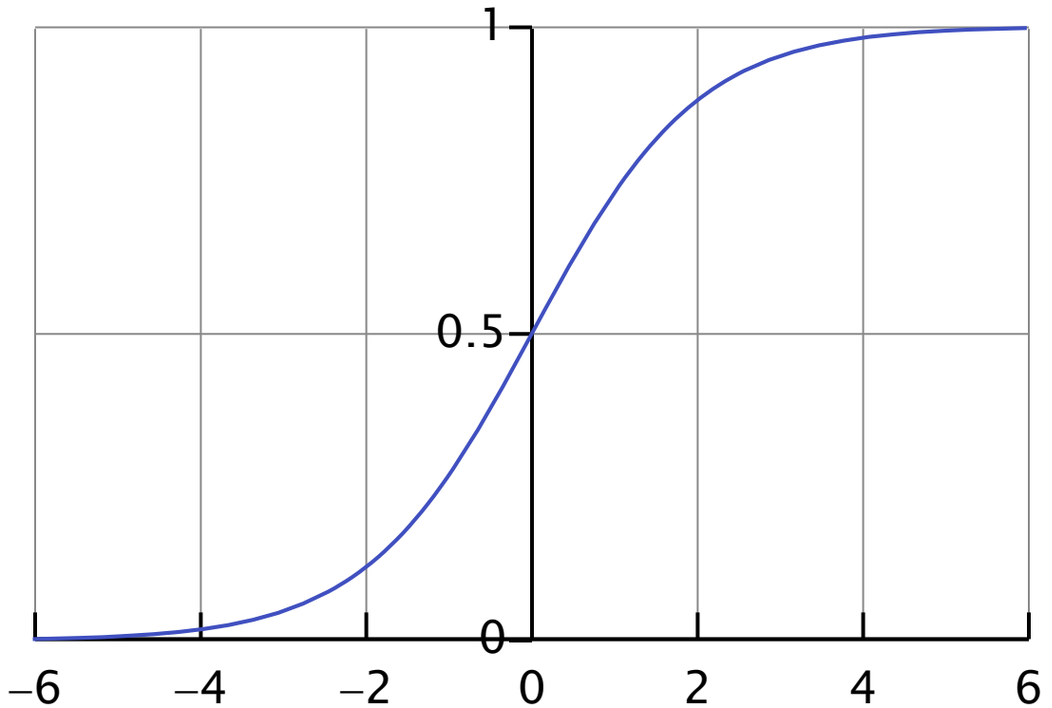
Notebook

- Load the dataset df = pd.read_csv('../data/Default.csv')
- Describe it
- boxplot balance and income vs default

$P(Y = 1/ X)$

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

SIGMOID FUNCTION



$$p(X) = \frac{e^{\beta_0 + \beta_1 X} }{1 + e^{\beta_0 + \beta_1 X}}$$

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X$$

ODDS RATION INTERPRETATION

Exemple in the default dataset

$$ p(X) = 0.2 \iff \frac{0.2}{1 -0.2} = 0.25 $$

- 1/5 people with ods 1/4 will default

$$ p(X) = 0.9 \iff \frac{0.9}{1 -0.9} = 9 $$

- 9 out of 10 people (90%) with ods 9 will default

ESTIMATION OF LR COEFFS

Maximum Likelihood Estimation

1500 and income of \

LAB ON DEFAULT DATASET

- Z test: the regression coefficient divided by its standard error. The larger the better
- P values: % of the null hypothesis that the coefficient is pure random

METRICS

So far we've only looked at accuracy: Number of correctly identified classes

Correctly identified:

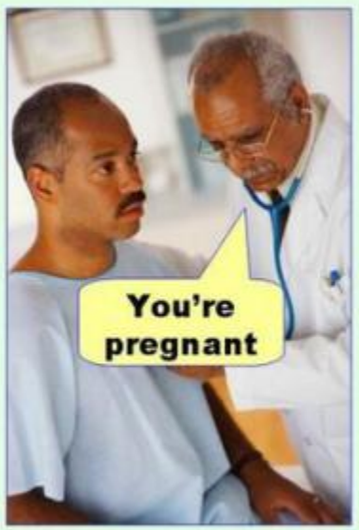
- TP = True Positive
- TN = True Negatives

Incorrectly identified:

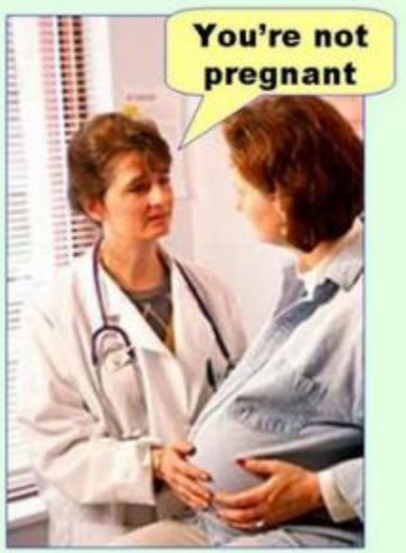
- FP = False Positive
- FN = False Negatives

METRICS

Type I error
(false positive)



Type II error
(false negative)



ACCURACY

- TP = True Positive
- TN = True Negatives
- FP = False Positive
- FN = False Negatives

How to you define accuracy?

ACCURACY

- TP = True Positive
- TN = True Negatives
- FP = False Positive
- FN = False
  Negatives

$$ Accuracy = \frac{ TP + TN }{TP + FP + TN + FN} $$

CONFUSION MATRIX

		Prediction	
		Cat	Dog
Actual	Cat	15	35
	Dog	40	10

- TP, TN?
- FN, FP?
- Accuracy?

CONFUSION MATRIX

In scikit:

```
from sklearn.metrics import confusion_matrix
y_true = [0,0,0,0,0,1,1,1,1,1]
y_pred = [0,0,0,1,1,0,1,1,1,1]
confusion_matrix(y_true, y_pred)
```

NOTEBOOK

Split Default into train and test (80/20)

- On training dataset: 5 fold CV with Logistic Regression on all regressors
- Use GridSearchCV with

```
parameters = {'solver': ['newton-cg','lbfgs','liblinear','sag']}
```

- What's the best model? Accuracy? best parameter?
- Print the confusion matrix for the best model on the test set
- Compare with best model with KNN 5 fold CV and K = [2,5,10, 20]

Same results?

WHAT'S THE RIGHT THRESHOLD?

- for your best LR model, print the results of predict() and predict_proba()
- predict() returns a class for all new samples
- predict_proba() returns a probability

What's the threshold?

Use predict_proba and a different threshold => you should find a different confusion matrix

So what's the best one?

AUX AND ROC CURVE

- TPR = $\frac{ TP }{ P} = \frac{ TP }{ TP + FN}$ aka **Sensitivity** or **Recall**
- FPR = $\frac{ FP }{ N} = \frac{ FP }{ FP + TN}$ aka **Fall-out**

AUX AND ROC CURVE

RECEIVER OPERATING CHARACTERISTIC

ROC = TPR vs FPR for different Thresholds

sklearn.metrics.roc_curve returns TPR, FPR

plot to get the ROC Curve

notice that KN also has a predict_proba

on the same plot: ROC curve for KNN

AUX AND ROC CURVE

The AUX is Area under the Curve

sklearn.metrics.roc_auc_score

AUX AND ROC CURVE

So what's your best model KNN / LR according to AUC?

PRE PROCESSING THE DATA WITH SCIKIT

preprocessing

- scaling
- normalizing
- remove linear correlation with PCA
- binarization
- encoding categorical features

with

- fit
- transform

ENCODE CATEGORICAL VALUE

One Hot encoding, DictVectorizer or get_dummies Label encoder Factorize on pandas

http://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizor and http://fastml.com/converting-categorical-data-into-numbers-with-pandas-and-scikit-learn/

SCALING AND CENTERING

Let's scale and normalize the income and balance and see if that helps

```
load default dataset
train, test split
```

train 3 models over df[balance, income]

```
linear regression
Knn K = 2,
SVC
```

Compare accuracy before and after scaling

PCA TO DECORRELATE

Use 3 regressor

then PCA to 2

and diff?

http://scikit-learn.org/stable/auto_examples/plot_digits_pipe.html#example-plot-digits-pipe-py

MAKE CLASSIFICATION

make_classification

n_features = n_informative + n_redundant + n_repeated + noisy feature

- what's the influence of n_redundant
- what's the influence of noisy features,
- does repeating features help? why?
- what's the catch?

PROJECT 3

LINKS

- http://blog.echen.me/2011/04/27/choosing-a-machine-learning-classifier/
- https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms