Python Data Products

Course 2: Design thinking and predictive pipelines

Lecture: classification in Python

Learning objectives

In this lecture we will...

- Demonstrate how to set up classification problems in Python
- Introduce the **LogisticRegression** model from the **sklearn** library

Dataset – Polish bankruptcies

We'll look at a simple dataset from the UCI repository: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data

- This dataset is concerned with which (Polish) companies go bankrupt

Polish companies bankruptcy data Data Set

Download: Data Folder, Data Set Description

Abstract: The dataset is about bankruptcy prediction of Polish companies.The bankrupt companies were analyzed in the period 2000-2012, while the still operating companies were evaluated from 2007 to 2013.

Data Set Characteristics:	Multivariate	**Number of Instances:**	10503	**Area:**	Business
Attribute Characteristics:	Real	**Number of Attributes:**	64	**Date Donated**	2016-04-11
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	44898

Source:

Creator: Sebastian Tomczak
-- Department of Operations Research, WrocÅ‚aw University of Science and Technology, wybrzeÅ¼e WyspiaÅ„skiego 27, 50-370, WrocÅ‚aw, Poland

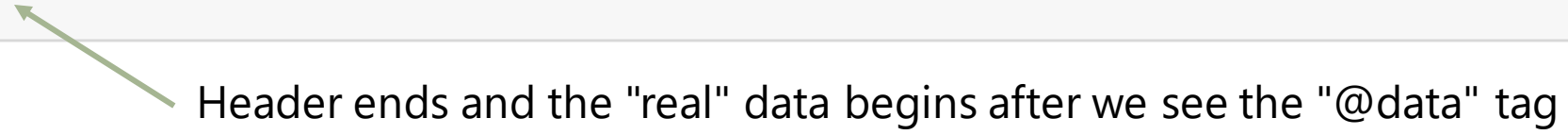
Donor: Sebastian Tomczak (sebastian.tomczak '@' pwr.edu.pl), Maciej Zieba (maciej.zieba '@' pwr.edu.pl), Jakub M. Tomczak (jakub.tomczak '@' pwr.edu.pl), Tel. (+48) 71 320 44 53

Reading the data

- Data is in CSV format, but first contains a header that we need to skip

In [1]:
```
f = open("datasets/bankruptcy/5year.arff", 'r')
```

In [2]:
```
while not '@data' in f.readline():
    pass
```



- Next we read the CSV data. We (a) skip rows with missing entries; (b) convert all fields to floats; and (c) convert the label to a bool

In [3]:
```
dataset = []
for l in f:
    if '?' in l:
        continue
    l = l.split(',')
    values = [1] + [float(x) for x in l]
    values[-1] = values[-1] > 0 # Convert to bool
    dataset.append(values)
```

Processing the data

- Next let's look at some simple statistics about our data

```
In [4]: len(dataset)
Out[4]: 3031
```

Number of samples (after discarding missing values)

```
In [5]: sum([x[-1] for x in dataset])
Out[5]: 102
```

Number of **positive** samples

- Next we extract our features (X) and labels (y), much as we would do for a regression problem

```
In [6]: X = [values[:-1] for values in dataset]

In [7]: y = [values[-1] for values in dataset]
```

True/False labels

Concept: The **sklearn** library

The **sklearn** library contains a number of different regression and classification models

For example:

- linear_model.LinearRegression() - linear regression
- linear_model.LogisticRegression() - logistic regression
- svm.SVC - Support Vector Classifier

- In this lecture we'll use the **LogisticRegression** module

Fitting the logistic regression model

- First we import the library and create an instance of the model, before fitting it to data

```
In [8]: from sklearn import linear_model

In [9]: model = linear_model.LogisticRegression()

In [10]: model.fit(X, y)

Out[10]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                   penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                   verbose=0, warm_start=False)
```

- Note that this function doesn't produce any output, rather it just updates the class instance to store the model

Making predictions

- Make predictions from the data:

```
In [11]: predictions = model.predict(X)

In [12]: predictions
Out[12]: array([False, False, False, ..., False, False, False])
```

- Check whether they match the labels

```
In [13]: correctPredictions = predictions == y

In [14]: correctPredictions
Out[14]: array([ True,  True,  True, ..., False, False, False])
```

- And compute the error

```
In [15]: sum(correctPredictions) / len(correctPredictions)
Out[15]: 0.9663477400197954
```

Training vs. Testing?

We achieved fairly high accuracy using a simple classifier "off the shelf"

- But note that we're evaluating our classifier on the same data that was used to train it
- How can we be sure that our classifier will work well on **unseen data?**
- This is something we'll cover in the next course, when we look at **training, testing, and validation**

Other classification algorithms in sklearn

This example showed how to use **logistic regression,** but other classifiers are available in sklearn and have a similar interface:

- sklearn.svm.SVC: **Support Vector Classifier**
- sklearn.tree.DesicionTreeClassifier: **Decision trees**
- sklearn.naive_bayes: **Naïve Bayes**
- sklearn.neighbors.KNeighborsClassifier: **Nearest Neighbors**

- see http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html for additional comparisons

Summary of concepts

- Introduced the sklearn library
- Showed how to set up a simple classification problem in Python

On your own...

- Try to set up a similar classification problem using another of the UCI datasets – look for classification datasets that have numerical attributes (i.e., datasets similar to the one used for this exercise)