BUSINESS INTELLIGENCE LABORATORY

MICROSOFT AZURE ML

Anna Monreale
Computer Science Department, University of Pisa
Data Mining Techniques

- Classification/Regression
- Association Rule Discovery
- Clustering
- Sequential Pattern Discovery
- Deviation Detection
- Text Mining
- Web Mining
- Social Network Analysis
- ...

Business Intelligence Lab
Tools for data mining

- From **DBMS**
  - SQL Server Analysis Services
  - Oracle Data Miner
  - IBM DB2 Intelligent Miner (discontinued)

- From **Statistical analysis**
  - IBM Modeler (formerly SPSS Clementine)
  - SAS Miner

- From **Machine-Learning**
  - Knime
  - Weka

- An updated list
Standards

- XML representation of data mining models
  - Predictive Modelling Markup Language: PMML

- API for accessing data mining services
  - Microsoft OLE DB for DM
  - Java JDM

- SQL Extensions for data mining
  - Standard SQL/MM Part 6 Data Mining
  - Oracle, DB2 & SQL Server have non-standard extensions
    - SSAS DMX query language and Data Mining queries
The KDD process starts with raw data and ends up with a model derived from that data.
What Azure ML Provides?

- To make life easier for data scientists, Azure ML provides several different components
- **ML Studio**
  - a graphical tool that can be used to control the process from beginning to end.
  - Once an effective model is found it helps to deploy that model on Microsoft Azure.
- A set of **data preprocessing modules**
- A set of **data mining and machine learning algorithms**
- An **Azure ML API** that lets applications access the chosen model once it is deployed on Azure
Azure ML Studio

- To use Azure Machine Learning Studio, you need to have a Machine Learning workspace
  - https://studio.azureml.net/?selectAccess=true&o=2

- This workspace contains the tools you need to create, manage, and publish experiments

- Azure Machine Learning for free
  - it is not possible to create more workspace

- Sign-in to the Microsoft Azure classic portal
  - you can create new workspace.
Azure ML user interface

- Drag and drop datasets, data preprocessing modules, algorithms and more onto its design surface.
- Connect components together graphically.
- Run experiments and evaluate the model.
Modules for Preparing Data

- **Clean Missing Data**
  - fill in missing values in a dataset: replace all missing values with zero, mean, median, or mode of the other values in this column in the table

- **Project Columns**
  - removes columns that are not useful

- **Metadata Editor**
  - changing the types of columns in a dataset

- **Apply Math Operations**
  - performing mathematical operations on data

- **Data preprocessing modules written in R or Python**
ML/DM Algorithms

- Algorithms for classification
  - Multiclass Decision Jungle, Two-Class Boosted Decision Tree, and One-vs-All Multiclass

- Algorithms for regression
  - Bayesian Linear Regression, Boosted Decision Tree Regression, and Neural Network Regression.

- Algorithms for clustering
  - K-Means Clustering

- It is possible to add any new algorithm
Reminds on classification
Who are my best customers?

- ... given their age and frequency of visit!
- Good customers = top buyers, buy more than X, ...

<table>
<thead>
<tr>
<th>Age (years)</th>
<th>NVisits</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 26</td>
<td></td>
</tr>
<tr>
<td>&gt;= 38</td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing customer demographics](image-url)

- Age <= 26
- Age >= 38
- NVisits > 20
- Good customers
- Bad customers

… given their age and frequency of visit!
... described with a decision tree!
A set of examples (or instances or cases) which described a concept or event (class) given predictive attributes (or features)

- Attributes can be either continuous or discrete (maybe discretized)
- The class is discrete

<table>
<thead>
<tr>
<th>outlook</th>
<th>temperature</th>
<th>humidity</th>
<th>windy</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>85</td>
<td>85</td>
<td>false</td>
<td>Don’t Play</td>
</tr>
<tr>
<td>sunny</td>
<td>80</td>
<td>90</td>
<td>true</td>
<td>Don’t Play</td>
</tr>
<tr>
<td>overcast</td>
<td>83</td>
<td>78</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>70</td>
<td>96</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>68</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>65</td>
<td>70</td>
<td>true</td>
<td>Don’t Play</td>
</tr>
<tr>
<td>overcast</td>
<td>64</td>
<td>65</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>72</td>
<td>95</td>
<td>false</td>
<td>Don’t Play</td>
</tr>
<tr>
<td>sunny</td>
<td>69</td>
<td>70</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>75</td>
<td>80</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>sunny</td>
<td>75</td>
<td>70</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>72</td>
<td>90</td>
<td>true</td>
<td>Play</td>
</tr>
<tr>
<td>overcast</td>
<td>81</td>
<td>75</td>
<td>false</td>
<td>Play</td>
</tr>
<tr>
<td>rain</td>
<td>71</td>
<td>80</td>
<td>true</td>
<td>Don’t Play</td>
</tr>
</tbody>
</table>
A function \( f(sample) = \text{class} \), called a **classification model**, that describes/predict the class value given the feature values of a sample obtained by generalizing the samples of the training set.

- **Usage of a classification model:**
  - **descriptively**
    - Which customers have abandoned?
  - **predictively**
    - Over a **score set** of samples with unknown class value
    - Which customers will respond to this offer?
How to evaluate a class model?

- **Holdout method**
  - Split the available data into two sets
  - Training set is used to build the model
  - Test set is used to evaluate the interestingness of the model
    - Typically, training is 2/3 of data and test is 1/3
How good is a classification model?

- **Stratified holdout**
  - Available data is divided by stratified sampling with respect to class distribution

- **(Stratified) n-fold cross-validation**
  - Available data divided into \( n \) parts of equal size
  - For \( i=1 \ldots n \), the \( i \)-th part is used as test set and the rest as training set for building a classifier
  - The average quality measure of the \( n \) classifiers is statistically more significant than the holdout method
  - The FINAL classifier is the one training from all the available data
    - Cross-validation is useful when data is scarce or attribute distributions are skewed
Quality measures: accuracy

- **Accuracy**: percentage of cases in the test set that is correctly predicted by the model
  - E.g., accuracy of 80% means that in 8 cases out of 10 in the test set the predicted class is the same of the actual class

- **Misclassification % = (100 − accuracy)**

- **Lower bound on accuracy**: majority classifier
  - A trivial classifier for which \( f(\text{case}) = \) majority class value
  - Its accuracy is the percentage of the majority class
  - E.g., two classes: fraud 2% legal 98%
  - Its hard to beat the 98% accuracy
Quality measures: confusion matrix

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly Classified Instances</td>
<td>42</td>
<td>73.6842%</td>
</tr>
<tr>
<td>Incorrectly Classified Instances</td>
<td>15</td>
<td>26.3158%</td>
</tr>
<tr>
<td>Total Number of Instances</td>
<td>57</td>
<td></td>
</tr>
</tbody>
</table>

== Confusion Matrix ==

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>&lt;-- classified as</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>14</td>
<td>6</td>
<td>a = bad</td>
</tr>
<tr>
<td>b</td>
<td>9</td>
<td>28</td>
<td>b = good</td>
</tr>
</tbody>
</table>
Quality measures: precision

**Precision**: accuracy of predicting “C”

\[
\text{Precision} = \frac{\text{# Cases predicted Class=C and with real Class=C}}{\text{# Cases predicted Class=C}}
\]

76% of times predictions >50K are correct
Recall: coverage of predicting “C”

# Cases predicted Class=C and with real Class=C
# Cases with real Class=C

Quality measures: recall

59.4% of real class >50K are found by predictions
Measures: lift chart

- **Classifier:** $f(\text{sample, class}) = \text{confidence}$
  - and then $f(\text{sample}) = \text{argmax}_{\text{class}} f(\text{sample, class})$
  - E.g., $f(\text{sample, play}) = 0.3$  $f(\text{sample, don’t play}) = 0.7$

- **Samples in the test set can be ranked according to a fixed class**
  - Rank customers on the basis of the classifier confidence they will respond to an offer

- **Lift chart**
  - **X-axis:** ranked sample of the test set
  - **Y-axis:** percentage of the total cases in the test set with the actual class value included in the ranked sample of the test set (i.e., recall)
  - Plots: performance of a classifier vs random ranking
  - Useful when resources (e.g., budget) are limited
Contacting only 50% of customer will reach 80% of those who respond. Lift = 80/50 = 1.6
Lift Chart - variants

- Lift(X) = recall(X)
  - Estimation of random classifier lift
  - Previous example, Lift(50%) = 80%

- LiftRatio(X) = recall(X) / X
  - Ratio of lift over random order
  - Previous example, LiftRatio(50%) = 80% / 50% = 1.6

- Profit chart
  - Given a cost/benefit model, the Y axis represent the total cost/gain when contacting X and not contacting TestSet \( \setminus X \)
The unbalancing problem

- For unbalanced class values, it is difficult to obtain a good model
  - Fraud = 2% Normal = 98%
    - The majority classifier is accurate at 98% but it is not useful

- Oversampling and Undersampling
  - Select a training set with a more balanced distribution of class values A and B
    - 60-70% for class A and 30-40% for class B
    - By increasing the number of cases with class B (oversampling) or by reducing those with class A (undersampling)
  - The training algorithm has more chances of distinguishing characteristics of A VS B
    - The test set MUST have the original distribution of values

- Cost Sensitive Classifier, Ensembles (bagging, boosting, stacking)
  - Weights errors, build several classifiers and average their predictions
German Credit Data

- available at UCI repository but also in Azure ML Studio
- Contains observations on 30 variables for 1000 past applicants for credit
- Each applicant was rated as “good credit” (700 cases) or “bad credit” (300 cases).
- Develop a classifier to determine if a new applicant is a good credit risk or a bad credit risk
Let us assume that a correct decision of the bank would result in a profit.

A correct decision here means that the bank predicts an application to be good and it actually turns out to be good.

When the opposite is true, i.e. bank predicts the application to be good but it turns out to be bad credit, then the loss is 100% of the credit.

|       | Bad   | Good
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>0</td>
<td>-500DM</td>
</tr>
<tr>
<td>Good</td>
<td>0</td>
<td>100DM</td>
</tr>
</tbody>
</table>
Benefit Chart
Cost Matrix

- Misclassification have been assessed as follows: the costs of incorrectly saying an applicant is a good credit risk outweigh the incorrectly saying an applicant is a bad credit risk by a factor of 5.

<table>
<thead>
<tr>
<th></th>
<th>Bad</th>
<th>Good</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bad</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Good</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Meta Cost: cost-sensitive

- On the train data apply a classifier getting probability of a class label $P(j \mid x)$
- Compute expected risk of classifying $x$ with class $i$:

$$R(i \mid x) = \sum_j P(j \mid x)C(i, j)$$

- Re-label the train data with the class $i$ having lower risk
- Learn a model on the cost-sensitive train data