Car Insurance

Prvák, Tomi, Havri
Sumo report - expectations
Sumo report - reality
Bc. Jan Tomášek

Deeper look into data set
Column approach
Reminder

What the hell is this competition about ???
Attributes overview

customer_ID, record_type, dateTime, location, group_size, homeowner, car_age, car_value, risk_factor, age_oldest, age_youngest, married_couple, C_previous, duration_previous, A,B,C,D,E,F,G, cost
Data problems

- lot of nan values in
  - risk factor
  - c_previous
  - nan values replaced with 0
- some attributes have to big granularity
  - date time
    - probably no need to use at all
Column correlation 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Column correlation $\text{corr}([\text{location risk_factor cost A B C}])$
Correlation result

- almost no linear dependency
- no chance to categorize with linear regression
- we need to add at least quadratic/cubic coefficients or use SVM machines with clever kernel function
Column approach Motivation

- last quotes benchmark 53%
- 72% buys previously visited product
- Can we bring our result nearer to 72%?
- average gives 45%
  - need something more clever
  - older are more important than previous views
    - weighted average instead
Future work

- better data filter and normalization
- clever column approach
- keep compatibility with our interface for result combinations
- don’t ever try to win sumo competition again
.Net & Horizontal data view

Štěpán Havránek
Machine learning & .Net

- Accord framework
  - [http://accord-framework.net](http://accord-framework.net)
  - Complex Computer science library
    - Math
    - Statistics
    - Machine learning
    - Neural networks
  - Uniform interface
  - Various data manipulation utilities
Machine learning & .Net

- AForge framework
  - http://www.aforgenet.com/
  - Primary for computer vision
  - Libraries for Computer Science
    - Especially Artificial intelligence
.Net implementation

- Data in MS SQL Server
  - Easy to fetch, aggregate, view, etc...
- Object model and object factory
  - Easy to transform
  - Made in Object Factory
.Net implementation
Data horizontal view

- What the customer info can say about the result purchased product parameters?
  - Seven output parameters
    - mostly 4 options per each
- Let’s try to make a model only on customer parameters and verificate it
Data horizontal view

- Decision trees
  - Input attributes
    - Customer and his car info
      - Ages
      - Car value
      - Group size
      - Is homeowner
      - Is married
      - Risk factor
      - Previous purchase info
Data horizontal view

● Decision trees
  ○ Used learning algorithms
    ■ ID3
    ■ C4.5

● Model verification
  ○ 10 times cross validation
    ■ => 10 different models (trees)
  ○ Process
    ■ Split the data
    ■ Create (learn) model
    ■ Validate outputs
Results
- 50 - 79% mean validation error
  - Actual competition leader has score 54%
- At least two output parameters (A, E) are very dependent on customer
- C, D are less dependent
- B, F, G can’t be resolved from the customer info
Future work

● Environment for experiments is ready...
● Spread out the horizontal data object
  ○ Add product browsing history
● Divide the output parameters between different models and input parameter sets
● Pruning overfits
● Use as much as possible from the Accord Framework
  ○ Unify interfaces, lots of data and ML utilities
Michal Pokorný

SVC model
scikit-learn

- Python (3)
- NumPy, SciPy, matplotlib
- BSD licence

- Classification, regression, clusterization, dimensionality reduction, cross-validation, ...

Current plan

- Most customers choose some browsed plan
- Make some predictors of plan choice probabilities
- From browsed plans, pick the one with highest probability prediction
Plan probability predictor

- RBF support vector machine classifier
  - (Plus feature scaling)
- Possible features:
  - Vector of “customer constants” (no location & time for now)
  - Most commonly browsed plan, last browsed plan, ...
  - Histogram of browsing for every plan feature
Closer look on features

- **One-hot**
  - Day, previous C, home owner?, married couple?
  - A: 3, B: 2, C: 4, D: 4, E: 2, F: 4, G: 4

- **Scalar**
  - Group size, car age, car value, risk factor, age of oldest & youngest, cost of offer
Results so far

- Relatively slow training on all 77607 customers :( 
- Current best result: 53.793% (same as trivial benchmark [doesn’t give the same outputs, through])
  - But this was on scalar representations of categories, so there might be some progress after training on better representation finishes :)}
Scalar vs. one-hot (small dataset)

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.81</td>
<td>0.45</td>
<td>0.57</td>
<td>56</td>
</tr>
<tr>
<td>1</td>
<td>0.97</td>
<td>0.80</td>
<td>0.88</td>
<td>181</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>0.00</td>
<td>0.00</td>
<td>55</td>
</tr>
<tr>
<td>avy / total</td>
<td>0.57</td>
<td>0.69</td>
<td>0.60</td>
<td>292</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.81</td>
<td>0.45</td>
<td>0.57</td>
<td>56</td>
</tr>
<tr>
<td>1</td>
<td>0.97</td>
<td>0.80</td>
<td>0.88</td>
<td>181</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>0.00</td>
<td>0.00</td>
<td>55</td>
</tr>
<tr>
<td>avy / total</td>
<td>0.57</td>
<td>0.69</td>
<td>0.60</td>
<td>292</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>recall</th>
<th>f1-score</th>
<th>support</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.81</td>
<td>0.45</td>
<td>0.57</td>
<td>56</td>
</tr>
<tr>
<td>1</td>
<td>0.97</td>
<td>0.80</td>
<td>0.88</td>
<td>181</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>0.00</td>
<td>0.00</td>
<td>55</td>
</tr>
<tr>
<td>avy / total</td>
<td>0.57</td>
<td>0.69</td>
<td>0.60</td>
<td>292</td>
</tr>
</tbody>
</table>
What’s next?

- “Naive Bayes assumption”: category membership classifier scores are multiplied...
  - Higher-order classifiers?
- Do something about missing values
  - scikit Imputer
- Throw in more features if nothing works...
- Ensemble if something works...