EXAMPLE-DEPENDENT COST-SENSITIVE CLASSIFICATION
applications in financial risk modeling and marketing analytics

September 15, 2015

Alejandro Correa Bahnsen
with
Djamila Aouada, SnT
Björn Ottersten, SnT
Motivation

• Classification: **predicting the class of a set of examples given their features.**

• Standard classification methods aim at minimizing the errors

• Such a traditional framework assumes that all **misclassification errors carry the same cost**

• This is not the case in many real-world applications: **Credit card fraud detection, churn modeling, credit scoring, direct marketing.**
Motivation

• **Credit card fraud detection**, failing to detect a fraudulent transaction may have an economical impact from a few to thousands of Euros, depending on the particular transaction and card holder.

• **Credit scoring**, accepting loans from bad customers does not have the same economical loss, since customers have different credit lines, therefore, different profit.

• **Churn modeling**, misidentifying a profitable or unprofitable churner has a significant different economic result.

• **Direct marketing**, wrongly predicting that a customer will not accept an offer when in fact he will, may have different financial impact, as not all customers generate the same profit.
Agenda

• Motivation
• Cost-sensitive classification
  Background
• Real-world cost-sensitive applications
  Credit card fraud detection, churn modeling, credit scoring, direct marketing
• Proposed cost-sensitive algorithms
  Bayes minimum risk, cost-sensitive logistic regression, cost-sensitive decision trees, ensembles of cost-sensitive decision trees
• Experiments
  Experimental setup, results
• Conclusions
  Contributions, future work
Predict the class of set of examples given their features

\[ f: S \rightarrow \{0,1\} \]

Where each element of \( S \) is composed by \( X_i = [x_i^1, x_i^2, ..., x_i^k] \)

It is usually evaluated using a traditional misclassification measures such as Accuracy, F1Score, AUC, among others.

However, these measures assume that different misclassification errors carry the same cost.
We define a cost measure based on the cost matrix [Elkan 2001]

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive $y_i = 1$</th>
<th>Actual Negative $y_i = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive $c_i = 1$</td>
<td>$C_{TP_i}$</td>
<td>$C_{FP_i}$</td>
</tr>
<tr>
<td>Predicted Negative $c_i = 0$</td>
<td>$C_{FN_i}$</td>
<td>$C_{TN_i}$</td>
</tr>
</tbody>
</table>

From which we calculate the **Cost** of applying a classifier to a given set

$$\text{Cost}(f(S)) = \sum_{i=1}^{N} y_i (c_i C_{TP_i} + (1 - c_i) C_{FN_i}) + (1 - y_i) (c_i C_{FP_i} + (1 - c_i) C_{TN_i})$$
Background - Cost-sensitive evaluation

However, the total cost may not be easy to interpret. Therefore, we propose a *Savings* measure as the cost vs. the cost of using no algorithm at all

\[
\text{Savings}(f(S)) = \frac{\text{Cost}_l(f(S)) - \text{Cost}(f(S))}{\text{Cost}_l(f(S))}
\]

Where \( \text{Cost}_l(f(S)) \) is the cost of predicting the costless class

\[
\text{Cost}_l(f(S)) = \min\{\text{Cost}(f_0(S)), \text{Cost}(f_1(S))\} 
\]
Research in example-dependent cost-sensitive classification has been narrow, mostly because of the lack of publicly available datasets [Aodha and Brostow 2013].

Standard approaches consist in re-weighting the training examples based on their costs:

• Cost-proportionate rejection sampling [Zadrozny et al. 2003]

• Cost-proportionate oversampling [Elkan 2001]
• Motivation
• Cost-sensitive classification
  Background
• **Real-world cost-sensitive applications**
  Credit card fraud detection, churn modeling, credit scoring, direct marketing
• **Proposed cost-sensitive algorithms**
  Bayes minimum risk, cost-sensitive logistic regression, cost-sensitive decision trees, ensembles of cost-sensitive decision trees
• **Experiments**
  Experimental setup, results
• **Conclusions**
  Contributions, future work
Credit card fraud detection

Estimate the probability of a transaction being fraud based on analyzing customer patterns and recent fraudulent behavior

Issues when constructing a fraud detection system [Bolton et al., 2002]:
• Skewness of the data
• Cost-sensitivity
• Short time response of the system
• Dimensionality of the search space
• Feature preprocessing
Credit card fraud detection is a **cost-sensitive problem**. As the cost due to a false positive is different than the cost of a false negative.

- **False positives**: When predicting a transaction as fraudulent, when in fact it is not a fraud, there is an administrative cost that is incurred by the financial institution.

- **False negatives**: Failing to detect a fraud, the amount of that transaction is lost.

Moreover, it is not enough to assume a constant cost difference between false positives and false negatives, as the amount of the transactions **varies quite significantly**.
## Credit card fraud detection

### Cost matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive $y_i = 1$</th>
<th>Actual Negative $y_i = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive $c_i = 1$</td>
<td>$C_{TP_i} = C_a$</td>
<td>$C_{FP_i} = C_a$</td>
</tr>
<tr>
<td>Predicted Negative $c_i = 0$</td>
<td>$C_{FN_i} = Amt_i$</td>
<td>$C_{TN_i} = 0$</td>
</tr>
</tbody>
</table>

## Credit card fraud detection

### Raw features

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction ID</td>
<td>Transaction identification number</td>
</tr>
<tr>
<td>Time</td>
<td>Date and time of the transaction</td>
</tr>
<tr>
<td>Account number</td>
<td>Identification number of the customer</td>
</tr>
<tr>
<td>Card number</td>
<td>Identification of the credit card</td>
</tr>
<tr>
<td>Transaction type</td>
<td>ie. Internet, ATM, POS, ...</td>
</tr>
<tr>
<td>Entry mode</td>
<td>ie. Chip and pin, magnetic stripe, ...</td>
</tr>
<tr>
<td>Amount</td>
<td>Amount of the transaction in Euros</td>
</tr>
<tr>
<td>Merchant code</td>
<td>Identification of the merchant type</td>
</tr>
<tr>
<td>Merchant group</td>
<td>Merchant group identification</td>
</tr>
<tr>
<td>Country</td>
<td>Country of trx</td>
</tr>
<tr>
<td>Country 2</td>
<td>Country of residence</td>
</tr>
<tr>
<td>Type of card</td>
<td>ie. Visa debit, Mastercard, American Express...</td>
</tr>
<tr>
<td>Gender</td>
<td>Gender of the card holder</td>
</tr>
<tr>
<td>Age</td>
<td>Card holder age</td>
</tr>
<tr>
<td>Bank Issuer</td>
<td>bank of the card</td>
</tr>
</tbody>
</table>
## Credit card fraud detection

Transaction **aggregation** strategy [Whitrow, 2008]

<table>
<thead>
<tr>
<th>TrxId</th>
<th>Time</th>
<th>Type</th>
<th>Country</th>
<th>Amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/1 18:20</td>
<td>POS</td>
<td>Lux</td>
<td>250</td>
</tr>
<tr>
<td>2</td>
<td>1/1 20:35</td>
<td>POS</td>
<td>Lux</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>1/1 22:30</td>
<td>ATM</td>
<td>Lux</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>2/1 00:50</td>
<td>POS</td>
<td>Ger</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>2/1 19:18</td>
<td>POS</td>
<td>Ger</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>2/1 23:45</td>
<td>POS</td>
<td>Ger</td>
<td>150</td>
</tr>
<tr>
<td>7</td>
<td>3/1 06:00</td>
<td>POS</td>
<td>Lux</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Raw Features</th>
<th>Aggregated Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No Trx last 24h</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
Credit card fraud detection

Proposed **periodic** features

When is a customer expected to make a new transaction?

Considering a **von Mises distribution** with a period of 24 hours such that

\[ P(\text{time}) \sim \text{vonmises}(\mu, \sigma) = \frac{e^{(\sigma \cos(\text{time} - \mu))}}{2\pi I_0(\sigma)} \]

where \( \mu \) is the mean, \( \sigma \) is the standard deviation, and \( I_0 \) is the Bessel function.
Proposed periodic features

Credit card fraud detection

Credit scoring

Classify which potential customers are likely to default a contracted financial obligation based on the customer’s past financial experience.

It is a cost-sensitive problem as the cost associated with approving a bad customer, i.e., false negative, is quite different from the cost associated with declining a good customer, i.e., false positive. Furthermore, the costs are not constant among customers. This is because loans have different credit line amounts, terms, and even interest rates.
Credit scoring

Cost matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive $\mathbf{y}_i = 1$</th>
<th>Actual Negative $\mathbf{y}_i = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Positive $c_i = 1$</td>
<td>$C_{TP_i} = 0$</td>
<td>$C_{FP_i} = r_i + C^a_{FP}$</td>
</tr>
<tr>
<td>Predicted Negative $c_i = 0$</td>
<td>$C_{FN_i} = Cl_i \times L_{gd}$</td>
<td>$C_{TN_i} = 0$</td>
</tr>
</tbody>
</table>

Predict the probability of a customer defecting using historical, behavioral and socioeconomical information.

This tool is of great benefit to subscription based companies allowing them to maximize the results of retention campaigns.
Churn modeling

Churn management campaign [Verbraken, 2013]

Inflow
New Customers

Customer Base
Active Customers

Predicted Churners
TP: Actual Churners
FP: Actual Non-Churners

Predicted Non-Churners
FN: Actual Churners
TN: Actual Non-Churners

Outflow
Effective Churners

Churn Model Prediction

\[ 1 - \gamma \]
## Cost matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive ( y_i = 1 )</th>
<th>Actual Negative ( y_i = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Predicted Positive</strong> ( c_i = 1 )</td>
<td>( C_{TP_i} = γ_i C_{oi} + (1 - γ_i)(CLV_i + C_a) )</td>
<td>( C_{FP_i} = C_{oi} + C_a )</td>
</tr>
<tr>
<td><strong>Predicted Negative</strong> ( c_i = 0 )</td>
<td>( C_{FN_i} = CLV_i )</td>
<td>( C_{TN_i} = 0 )</td>
</tr>
</tbody>
</table>

Direct marketing

Classify those customers who are more likely to have a certain response to a marketing campaign.

This problem is example-dependent cost sensitive, as the **false positives** have the cost of contacting the client, and **false negatives** have the cost due to the **loss of income** by failing to making the an offer to the right customer.
Direct marketing

Cost matrix

<table>
<thead>
<tr>
<th></th>
<th>Actual Positive</th>
<th>Actual Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$y_i = 1$</td>
<td>$y_i = 0$</td>
</tr>
<tr>
<td>Predicted Positive $c_i = 1$</td>
<td>$C_{TP_i} = C_a$</td>
<td>$C_{FP_i} = C_a$</td>
</tr>
<tr>
<td>Predicted Negative $c_i = 0$</td>
<td>$C_{FN_i} = Int_i$</td>
<td>$C_{TN_i} = 0$</td>
</tr>
</tbody>
</table>

Agenda

• Motivation
• Cost-sensitive classification
  Background
• Real-world cost-sensitive applications
  Credit card fraud detection, churn modeling, credit scoring, direct marketing
• Proposed cost-sensitive algorithms
  Bayes minimum risk, cost-sensitive logistic regression, cost-sensitive decision trees, ensembles of cost-sensitive decision trees
• Experiments
  Experimental setup, results
• Conclusions
  Contributions, future work
Proposed cost-sensitive algorithms

- **Bayes minimum risk (BMR)**

- **Cost-sensitive logistic regression (CSLR)**

- **Cost-sensitive decision trees (CSDT)**

- **Ensembles of cost-sensitive decision trees (ECSDT)**
Bayes Minimum Risk

Decision model based on **quantifying tradeoffs** between various decisions using probabilities and the costs that accompany such decisions

**Risk of classification**

\[
R(c_i = 0|x_i) = C_{TN_i}(1 - \hat{p}_i) + C_{FN_i} \cdot \hat{p}_i
\]

\[
R(c_i = 1|x_i) = C_{FP_i}(1 - \hat{p}_i) + C_{TP_i} \cdot \hat{p}_i
\]

Using the different risks the prediction is made based on the following condition:

\[
c_i = \begin{cases} 
0 & R(c_i = 0|x_i) \leq R(c_i = 1|x_i) \\
1 & \text{otherwise}
\end{cases}
\]
Cost-Sensitive Logistic Regression

- Logistic Regression Model

\[ \hat{p}_i = P(y_i = 0|x_i) = h_\theta(x_i) = g\left( \sum_{j=1}^{k} \theta_j x_i^j \right) \]

- Cost Function

\[ J_i(\theta) = -y_i \log(h_\theta(x_i)) - (1 - y_i) \log(1 - h_\theta(x_i)) \]

- Cost Analysis

\[ J_i(\theta) \approx \begin{cases} 0 & \text{if } y_i \approx h_\theta(x_i) \\ \infty & \text{if } y_i \approx 1 - h_\theta(x_i) \end{cases} \]

\[ C_{TP_i} = C_{TN_i} \approx 0 \]
\[ C_{FP_i} = C_{FN_i} \approx \infty \]
Cost-Sensitive Logistic Regression

• **Actual Costs**

\[
J^c(\theta) = \begin{cases} 
C_{TP_i} & \text{if } y_i = 1 \text{ and } h_\theta(x_i) \approx 1 \\
C_{TN_i} & \text{if } y_i = 0 \text{ and } h_\theta(x_i) \approx 0 \\
C_{FP_i} & \text{if } y_i = 0 \text{ and } h_\theta(x_i) \approx 1 \\
C_{FN_i} & \text{if } y_i = 1 \text{ and } h_\theta(x_i) \approx 0 
\end{cases}
\]

• **Proposed Cost-Sensitive Function**

\[
J^c(\theta) = \frac{1}{N} \sum_{i=1}^{N} y_i (h_\theta(x_i)C_{TP_i} + (1 - h_\theta(x_i))C_{FN_i}) + (1 - y_i)((1 - h_\theta(x_i))C_{FP_i} + (1 - h_\theta(x_i))C_{TN_i})
\]
A decision tree is a classification model that iteratively creates binary decision rules $(x^j, l^j_m)$ that maximize certain criteria (gain, entropy, ...). Where $(x^j, l^j_m)$ refers to making a rule using feature j on value m.

- Maximize the accuracy is different than maximizing the cost.
- To solve this, some studies had been proposed method that aim to introduce the cost-sensitivity into the algorithms [Lomax 2013]. However, research have been focused on class-dependent methods.

We proposed:
- Example-dependent cost based impurity measure
- Example-dependent cost based pruning criteria
Cost-Sensitive Decision trees

Proposed Cost based impurity measure

\[
S^l = \{x | x_i \in S \land x_i^j \leq l_m^j\} \quad \text{and} \quad S^r = \{x | x_i \in S \land x_i^j > l_m^j\}
\]

- The impurity of each leaf is calculated using:
  \[
  I_c(S) = \min \{\text{Cost}(f_0(S)), \text{Cost}(f_1(S))\}
  \]
  \[
  f(S) = \begin{cases} 
  0 & \text{if Cost}(f_0(S)) \leq \text{Cost}(f_1(S)) \\
  1 & \text{otherwise}
  \end{cases}
  \]

- Afterwards the \textbf{gain} of applying a given rule to the set \( S \) is:
  \[
  \text{Gain}_c((x^j, l_m^j)) = I_c(\pi_1) - \left( I_c(\pi_1^l) + I_c(\pi_1^r) \right)
  \]
Cost-Sensitive Decision trees

Decision trees construction

- The rule that **maximizes the gain** is selected
  \[
  (best_x, best_l) = \arg \max_{(j,m)} \left( \text{Gain} \left( (x^j, l^j_m) \right) \right)
  \]

- The process is repeated until a stopping criteria is met:
Cost-Sensitive Decision trees

Proposed cost-sensitive pruning criteria

• Calculation of the Tree savings and pruned Tree savings

\[
P_{Cc} = \frac{\text{Cost}(f(S, \text{Tree})) - \text{Cost}(f(S, \text{EB(Tree, branch)}))}{|\text{Tree}| - |\text{EB(Tree, branch)}|}
\]

• After calculating the pruning criteria for all possible trees. The maximum improvement is selected and the Tree is pruned.
• Later the process is repeated until there is no further improvement.
Ensembles of Cost-Sensitive Decision trees

Typical ensemble is made by combining $T$ different base classifiers. Each base classifier is trained by applying algorithm $M$ in a random subset

$$M_j \leftarrow M(S_j) \quad \forall j \in \{1, \ldots, T\}$$
Ensembles of Cost-Sensitive Decision trees

The core principle in ensemble learning, is to **induce random perturbations** into the learning procedure in order to produce several different base classifiers from a single training set, then **combining** the base classifiers in order to make the final prediction.
Ensembles of Cost-Sensitive Decision trees

Training set

Bagging

Pasting

Random forest

Random patches
After the base classifiers are constructed they are typically combined using one of the following methods:

- **Majority voting**

  \[ H(S) = f_{mv}(S, M) = \arg \max_{c \in \{0, 1\}} \sum_{j=1}^{T} 1_c \left( M_j(S) \right) \]

- **Proposed cost-sensitive weighted voting**

  \[ H(S) = f_{wv}(S, M, \alpha) = \arg \max_{c \in \{0, 1\}} \sum_{j=1}^{T} \alpha_j 1_c \left( M_j(S) \right) \]

\[ \alpha_j = \frac{1 - \varepsilon \left( M_j(S_{j1}^{ooob}) \right)}{\sum_{j_1=1}^{T} 1 - \varepsilon \left( M_{j_1}(S_{j_1}^{ooob}) \right)} \]

\[ \alpha_j = \frac{Savings \left( M_j(S_{j1}^{ooob}) \right)}{\sum_{j_1=1}^{T} Savings \left( M_{j_1}(S_{j_1}^{ooob}) \right)} \]
Ensembles of Cost-Sensitive Decision trees

- Proposed cost-sensitive stacking

\[ H(S) = f_S(S, M, \beta) = \frac{1}{1 + e^{-\left(\sum_{j=1}^{T} \beta_j M_j(S)\right)}} \]

Using the cost-sensitive logistic regression [Correa et. al, 2014] model:

\[ J(S, M, \beta) = \sum_{i=1}^{N} y_i (f_S(S, M, \beta)(C_{TP_i} - C_{FN_i}) + C_{FN_i}) + (1 - y_i)(f_S(S, M, \beta)(C_{FP_i} - C_{TN_i}) + C_{TN_i}) \]

Then the weights are estimated using

\[ \hat{\beta} = \arg \min_{\beta} J(S, M, \beta) \]
• Motivation
• Cost-sensitive classification
  Background
• Real-world cost-sensitive applications
  Credit card fraud detection, churn modeling, credit scoring, direct marketing
• Proposed cost-sensitive algorithms
  Bayes minimum risk, cost-sensitive logistic regression, cost-sensitive decision trees, ensembles of cost-sensitive decision trees

• Experiments
  Experimental setup, results

• Conclusions
  Contributions, future work
## Experimental setup - Datasets

<table>
<thead>
<tr>
<th>Database</th>
<th># Examples</th>
<th>% Positives</th>
<th>Cost (Euros)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraud</td>
<td>1,638,772</td>
<td>0.21%</td>
<td>860,448</td>
</tr>
<tr>
<td>Churn</td>
<td>9,410</td>
<td>4.83%</td>
<td>580,884</td>
</tr>
<tr>
<td>Kaggle Credit</td>
<td>112,915</td>
<td>6.74%</td>
<td>8,740,181</td>
</tr>
<tr>
<td>PAKDD09 Credit</td>
<td>38,969</td>
<td>19.88%</td>
<td>3,117,960</td>
</tr>
<tr>
<td>Direct Marketing</td>
<td>37,931</td>
<td>12.62%</td>
<td>59,507</td>
</tr>
</tbody>
</table>
Experimental setup - Methods

- **Cost-insensitive (CI):**
  - Decision trees (DT)
  - Logistic regression (LR)
  - Random forest (RF)
  - Under-sampling (u)

- **Cost-proportionate sampling (CPS):**
  - Cost-proportionate rejection-sampling (r)
  - Cost-proportionate over-sampling (o)

- **Bayes minimum risk (BMR)**

- **Cost-sensitive training (CST):**
  - Cost-sensitive logistic regression (CSLR)
  - Cost-sensitive decision trees (CSDT)
Experimental setup - Methods

• Ensemble cost-sensitive decision trees (ECSDT):

  Random inducers:
  • Bagging (CSB)
  • Pasting (CSP)
  • Random forest (CSRF)
  • Random patches (CSRP)

Combination:
• Majority voting (mv)
• Cost-sensitive weighted voting (wv)
• Cost-sensitive staking (s)
Each experiment was carried out 50 times.

For the parameters of the algorithms a grid search was made.

Results are measured by savings and F1Score.

Then the Friedman ranking is calculated for each method.
### Results

<table>
<thead>
<tr>
<th>Database</th>
<th>Algorithm</th>
<th>Savings</th>
<th>Savings (Euros)</th>
<th>% Pos.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraud</td>
<td>CSRP-wv-t</td>
<td>0.73</td>
<td>628,127</td>
<td>0.21</td>
</tr>
<tr>
<td>Churn</td>
<td>CSRP-s-t</td>
<td>0.17</td>
<td>98,750</td>
<td>4.83</td>
</tr>
<tr>
<td>Credit1</td>
<td>CSRP-mv-t</td>
<td>0.52</td>
<td>4,544,894</td>
<td>6.74</td>
</tr>
<tr>
<td>Credit2</td>
<td>LR-t-BMR</td>
<td>0.31</td>
<td>966,568</td>
<td>19.88</td>
</tr>
<tr>
<td>Marketing</td>
<td>LR-t-BMR</td>
<td>0.51</td>
<td>30,349</td>
<td>12.62</td>
</tr>
</tbody>
</table>

**Percentage of the highest savings**

![Graph showing percentage of the highest savings for different databases and algorithms](image-url)

- **Fraud**: CSRP-wv-t (0.21%), LR-t-BMR (0.31%), CSRP-s-t (0.17%)
- **Churn**: CSRP-mv-t (6.74%), LR-t-BMR (0.52%), CSRP-s-t (0.17%)
- **Credit1**: CSRP-mv-t (19.88%), LR-t-BMR (0.31%), CSRP-s-t (0.17%)
- **Credit2**: LR-t-BMR (12.62%), CSRP-mv-t (0.52%), CSRP-s-t (0.17%)
- **Marketing**: LR-t-BMR (4.83%), CSRP-mv-t (6.74%), CSRP-wv-t (0.21%)
Results of the **Friedman rank** of the savings (1=best, 28=worst)

<table>
<thead>
<tr>
<th>Family</th>
<th>Algorithm</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESDT</td>
<td>CSRP-wv-t</td>
<td>2.6</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSRP-s-t</td>
<td>3.4</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSRP-mv-t</td>
<td>4</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSB-wv-t</td>
<td>5.6</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSP-wv-t</td>
<td>7.4</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSB-mv-t</td>
<td>8.2</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSRF-wv-t</td>
<td>9.4</td>
</tr>
<tr>
<td>BMR</td>
<td>RF-t-BMR</td>
<td>9.4</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSP-s-t</td>
<td>9.6</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSP-mv-t</td>
<td>10.2</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSB-s-t</td>
<td>10.2</td>
</tr>
<tr>
<td>BMR</td>
<td>LR-t-BMR</td>
<td>11.2</td>
</tr>
<tr>
<td>CPS</td>
<td>RF-r</td>
<td>11.6</td>
</tr>
<tr>
<td>CST</td>
<td>CSDT-t</td>
<td>12.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Family</th>
<th>Algorithm</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>CST</td>
<td>CSLR-t</td>
<td>14.4</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSRF-mv-t</td>
<td>15.2</td>
</tr>
<tr>
<td>ESDT</td>
<td>CSRF-s-t</td>
<td>16</td>
</tr>
<tr>
<td>CI</td>
<td>RF-u</td>
<td>17.2</td>
</tr>
<tr>
<td>CPS</td>
<td>LR-r</td>
<td>19</td>
</tr>
<tr>
<td>BMR</td>
<td>DT-t-BMR</td>
<td>19</td>
</tr>
<tr>
<td>CPS</td>
<td>LR-o</td>
<td>21</td>
</tr>
<tr>
<td>CPS</td>
<td>DT-r</td>
<td>22.6</td>
</tr>
<tr>
<td>CI</td>
<td>DT-u</td>
<td>24.4</td>
</tr>
<tr>
<td>CPS</td>
<td>DT-o</td>
<td>25</td>
</tr>
<tr>
<td>CI</td>
<td>DT-t</td>
<td>26</td>
</tr>
<tr>
<td>CI</td>
<td>RF-t</td>
<td>26.2</td>
</tr>
</tbody>
</table>
Results of the **Friedman rank** of the savings organized by family
Results within the ECSDT family

By random inducer

By combination method

<table>
<thead>
<tr>
<th>Friedman Ranking</th>
<th>Bagging</th>
<th>Pasting</th>
<th>R. Forest</th>
<th>R. Patches</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Friedman Ranking</th>
<th>Majority Voting</th>
<th>Weighted Voting</th>
<th>Staking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison of the Friedman ranking of the savings and F1Score sorted by F1Score ranking
• Motivation
• Cost-sensitive classification
  Background
• Real-world cost-sensitive applications
  Credit card fraud detection, churn modeling, credit scoring, direct marketing
• Proposed cost-sensitive algorithms
  Bayes minimum risk, cost-sensitive logistic regression, cost-sensitive decision trees, ensembles of cost-sensitive decision trees
• Experiments
  Experimental setup, results
• Conclusions
  Contributions, future work
Conclusions

• New framework of example dependent cost-sensitive classification

• Using five databases, from four real-world applications: credit card fraud detection, churn modeling, credit scoring and direct marketing, we show that the proposed algorithms significantly outperforms the state-of-the-art cost-insensitive and example-dependent cost-sensitive algorithms

• Highlight the importance of using the real example-dependent financial costs associated with the real-world applications
Future research directions

- Multi-class example-dependent cost-sensitive classification
- Cost-sensitive calibration
- Staking cost-sensitive decision trees
- Example-dependent cost-sensitive boosting
- Online example-dependent cost-sensitive classification
## Contributions - Papers

<table>
<thead>
<tr>
<th>Date</th>
<th>Name</th>
<th>Conference / Journal</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>July 2013</td>
<td>Cost Sensitive Credit Card Fraud Detection using Bayes Minimum Risk</td>
<td>IEEE International Conference on Machine Learning and Applications</td>
<td>Published</td>
</tr>
<tr>
<td>October 2013</td>
<td>Improving Credit Card Fraud Detection with Calibrated Probabilities</td>
<td>SIAM International Conference on Data Mining</td>
<td>Published</td>
</tr>
<tr>
<td>June 2014</td>
<td>Credit Scoring using Cost-Sensitive Logistic Regression</td>
<td>IEEE International Conference on Machine Learning and Applications</td>
<td>Published</td>
</tr>
<tr>
<td>October 2014</td>
<td>Example-Dependent Cost-Sensitive Decision Trees</td>
<td>Expert Systems with Applications</td>
<td>Published</td>
</tr>
<tr>
<td>January 2015</td>
<td>A novel cost-sensitive framework for customer churn predictive modeling</td>
<td>Decision Analytics</td>
<td>Published</td>
</tr>
<tr>
<td>March 2015</td>
<td>Ensemble of Example-Dependent Cost-Sensitive Decision Trees</td>
<td>IEEE Transactions on Knowledge and Data Engineering</td>
<td>Under review</td>
</tr>
<tr>
<td>March 2015</td>
<td>Feature Engineering Strategies for Credit Card Fraud Detection</td>
<td>Expert Systems with Applications</td>
<td>Under review</td>
</tr>
<tr>
<td>June 2015</td>
<td>Detecting Credit Card Fraud using Periodic Features</td>
<td>IEEE International Conference on Machine Learning and Applications</td>
<td>In Press</td>
</tr>
</tbody>
</table>
Contributions - Costcla

**costcla** is a Python module for **cost-sensitive classification** built on top of Scikit-Learn, SciPy and distributed under the 3-Clause BSD license.

In particular, it provides:

- A set of example-dependent cost-sensitive algorithms
- Different real-world example-dependent cost-sensitive datasets.

**Installation**

`pip install costcla`

**Documentation:**

https://pythonhosted.org/costcla/

---

**Prepare dataset and load libraries**

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from costcla.metrics import savings_score
from costcla.datasets import load_creditscoring2
from costcla.sampling import cost_sampling
from costcla import models
data = load_creditscoring2()
X_train, X_test, y_train, y_test, cost_mat_train, cost_mat_test = 
train_test_split(data.data, data.target, data.cost_mat)
```

**Random forest**

```python
In [19]: f_RF = RandomForestClassifier()
y_pred = f_RF.fit(X_train, y_train).predict(X_test)
print(savings_score(y_test, y_pred, cost_mat_test))
0.042197359989
```

**cost-sensitive decision tree**

```python
In [2]: f_CSDT = models.CSDecisionTreeClassifier()
f_CSDT.fit(data.data, data.target, data.cost_mat)
y_pred = f_CSDT.predict(data.data)
print(savings_score(data.target, y_pred, data.cost_mat))
0.289489571352
```

**cost-sensitive random patches**

```python
In [33]: f_CSRP = costcla.models.CSRandomPatchesClassifier()
f_CSRP.fit(data.data, data.target, data.cost_mat)
y_pred = f_CSRP.predict(data.data)
print(savings_score(data.target, y_pred, data.cost_mat))
0.36666740467
```
Thank You!!

Alejandro Correa Bahnsen
Alejandro Correa Bahnsen
University of Luxembourg

albahnsen.com
al.bahnsen@gmail.com
http://www.linkedin.com/in/albahnsen
https://github.com/albahnsen/CostSensitiveClassification