Titanic Story

Jan 2019

This document describes how we can exploit Machine Learning with VisiRule and combine both human and machine intelligence. It is illustrated using the Titanic dataset.

There are 4 phases

A) Machine Learning converts data into decision tree
B) Chart editing converts decision tree into VisiRule chart
C) Deployment executes VisiRule chart
D) Analytics & Visualisation of chart execution

The results from [D] can be fed back into [A]
Introduction to the Titanic Dataset

On April 15, 1912, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for everyone.

The internet provides us with a dataset containing a collection of parameters for 891 passengers.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>survival</td>
<td>Survival (0 = no; 1 = yes)</td>
</tr>
<tr>
<td>class</td>
<td>Passenger class (1 = first; 2 = second; 3 = third)</td>
</tr>
<tr>
<td>name</td>
<td>Name</td>
</tr>
<tr>
<td>sex</td>
<td>Sex</td>
</tr>
<tr>
<td>age</td>
<td>Age</td>
</tr>
<tr>
<td>sibsp</td>
<td>Number of siblings/spouses aboard</td>
</tr>
<tr>
<td>parch</td>
<td>Number of parents/children aboard</td>
</tr>
<tr>
<td>ticket</td>
<td>Ticket number</td>
</tr>
<tr>
<td>fare</td>
<td>Passenger fare</td>
</tr>
<tr>
<td>cabin</td>
<td>Cabin</td>
</tr>
<tr>
<td>embarked</td>
<td>Port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)</td>
</tr>
<tr>
<td>boat</td>
<td>Lifeboat (if survived)</td>
</tr>
</tbody>
</table>

We can use this data to build an interactive system which can help us predict survival.
We can run the data through a Classifier (we use KNIME here) to generate a decision tree.

**Titanic data in KNIME**

**KNIME workflow**

Knime provides basic statistics and a Decision Tree.
Only 38% of the passengers survived. Not good. The Titanic was only carrying 20 lifeboats, not nearly enough for the 1,317 passengers and 885 crew members aboard.

Did all of the passengers have an equal chance of survival?

Social classes were heavily stratified in the early twentieth century. This was especially true on the Titanic, where the luxurious first-class areas were completely off limits to the middle-class passengers in second class, and especially to those who carried a third class “economy price” ticket.

For instance, passengers in first class had a 62% chance of survival, compared to a 25.5% chance for those in 3rd class. Additionally, the lower classes generally consisted of younger people, and the ticket prices for first class were predictably much higher than those for second and third class. The average ticket price for first class (£87.5) is equivalent to $13,487 in 2016.

While the Titanic was sinking, the officers prioritized who was allowed in a lifeboat with the strict maritime tradition of evacuating women and children first. The statistical results reflect the first part of this policy as, across all classes, women were much more likely to survive than the men.

The effectiveness of the second part of this “Women and children first” policy can be deduced by looking at the survival rate by age.

Machine Learning packages such as KNIME and RapidMiner can produce a decision tree which can be exported using a PMML export function.

PMML is a special XML format supported by many vendor ML implementations including KNIME and RapidMiner and covers many different ML areas.

https://en.wikipedia.org/wiki/Predictive_Model_Markup_Language

PMML provides a way for analytic applications to describe and exchange predictive models produced by data mining and machine learning algorithms. It supports common models such as logistic regression and feedforward neural networks. Version 0.9 was published in 1998. Subsequent versions have been developed by the Data Mining Group.

Since PMML is an XML-based standard, the specification comes in the form of an XML schema. PMML itself is a mature standard with over 30 organizations having announced products supporting PMML.
VisiRule provides a special component which can import PMML data and generate trees in the Prolog/VisiRule world.

This above screen shows some of the raw PMML and also the induced rules.

Sex
  equal(male)
  Fare
    lessOrEqual(52.27)
    Age
      lessOrEqual(9.5)
      Fare
        lessOrEqual(20.825)
        => 1
        greaterThan(20.825)
        => 0
        greaterThan(9.5)
        => 0
      greaterThan(52.27)
      Fare
        lessOrEqual(59.0875)
        => 1
        greaterThan(59.0875)
We can visualize the tree in Prolog

Next, we can use the Save VisiRule button to convert the tree into an initial VisiRule chart.
In the browser we can move around the various nodes with ease:
We can edit this chart in VisiRule. Why?

A] Most classifiers use binary splits on continuous fields, we can combine them using structured expressions

B] We may also want to converge common sub-trees.

C] We could also add explanations, HTML links

D] We could incorporate some report generation or letter writing output

E] We may want to add restorative actions to conclusions

F] We could for instance include the scores or probabilities associated with each conclusion, and use that to indicate some measure of confidence in the prediction.
We can generate horn clause rules from the tree as follows – these are directly executable in Prolog.

\begin{verbatim}
start( A ) :-
    'Sex1'('female'),
    'SibSp1'('>' 2.50'),
    A = '0'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('>' 48.20'),
    A = '1'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('=< 48.20'),
    'Pclass1'(2),
    A = '1'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('=< 48.20'),
    'Pclass1'(3),
    'Age2'('>' 29.50'),
    A = '0'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('=< 48.20'),
    'Pclass1'(3),
    'Age2'('=< 29.50'),
    'Fare5'('>' 15.37'),
    A = '1'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('=< 48.20'),
    'Pclass1'(3),
    'Age2'('=< 29.50'),
    'Fare5'('=< 15.37'),
    'Fare6'('>= 15.37'),
    A = '0'.

start( A ) :-
    'Sex1'('female'),
    'SibSp1'('=< 2.50'),
    'Fare4'('=< 48.20'),
    'Pclass1'(3),
    'Age2'('=< 29.50'),
    'Fare5'('=< 15.37'),
    'Fare6'('=< 7.90'),
\end{verbatim}
A = '1'.

\[
\text{start}(A) := \\
\quad 'Sex1'( 'male' ), \\
\quad 'Fare1'(' > 52.28 '), \\
\quad 'Fare3'(' > 59.09 '), \\
\quad A = '0'.
\]

\[
\text{start}(A) := \\
\quad 'Sex1'( 'male' ), \\
\quad 'Fare1'(' =< 52.28 '), \\
\quad 'Fare3'(' =< 59.09 '), \\
\quad A = '1'.
\]

\[
\text{start}(A) := \\
\quad 'Sex1'( 'male' ), \\
\quad 'Age1'(' > 9.50 '), \\
\quad 'Fare2'(' > 20.82 '), \\
\quad A = '0'.
\]

\[
\text{start}(A) := \\
\quad 'Sex1'( 'male' ), \\
\quad 'Age1'(' =< 9.50 '), \\
\quad 'Fare2'(' =< 20.82 '), \\
\quad A = '1'.
\]

We also present them in Flex as:

\[
\text{relation start(A)} \\
\quad \text{if 'Sex1'= 'male' and 'Fare1'= ' =< 52.28 ' and 'Age1'= ' =< 9.50 ' and 'Fare2'= ' =< 20.82 ' and A = '1'.}
\]

\[
\text{relation start('1')} \\
\quad \text{if 'Sex1'= 'male' and 'Fare1'= ' =< 52.28 ' and 'Age1'= ' =< 9.50 ' and 'Fare2'= ' =< 20.82 '}. \\
\]

\[
\text{OR} \\
\text{IF 'Sex1'= male AND 'Fare1' =< 52.28 AND 'Age1' =< 9.50 AND 'Fare2' =< 20.82 THEN 'Outcome'= survived}
\]
We can export all the nodes and connections of the chart as XML for other processes to consume:

### VisiRule Chart

<table>
<thead>
<tr>
<th>Node</th>
<th>Type</th>
<th>Text</th>
<th>From</th>
<th>To</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>node1</td>
<td>number_input</td>
<td>1 Fare2</td>
<td>node33</td>
<td>node31</td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 Fare?</td>
<td></td>
<td>node9</td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Fare</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node2</td>
<td>end</td>
<td>1 &lt;p&gt; 0 Count=0.0 &lt;p&gt; 1 Count=5.0</td>
<td>node8</td>
<td></td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 1?</td>
<td></td>
<td></td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node3</td>
<td>number_input</td>
<td>1 SibSp1</td>
<td>node14</td>
<td>node42</td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 SibSp?</td>
<td></td>
<td>node39</td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 SibSp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node4</td>
<td>expression</td>
<td>1 &gt; 29.50</td>
<td>node18</td>
<td>node22</td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 &gt; 29.50?</td>
<td></td>
<td></td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 &gt; 29.50</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node5</td>
<td>expression</td>
<td>1 ?'Fare2' &gt; 20.82</td>
<td>node30</td>
<td>node6</td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 &gt; 20.82?</td>
<td></td>
<td></td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 &gt; 20.82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node6</td>
<td>end</td>
<td>0 &lt;p&gt; 0 Count=8.63070539419087 &lt;p&gt; 0 Count=2.315332697093436</td>
<td>node5</td>
<td></td>
<td>X Y Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 0?</td>
<td></td>
<td></td>
<td>Height</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We can publish the VisiRule chart in different ways:

1] Use VisiRule Connect to upload this chart to a cloud based version of VisiRule Server:

![VisiRule Web Upload](image)

- Server IP address: 54.72.18.109
- User name: [paexamples]
- Password: ********
- VisiRule Name: cs_titanic0
- Title: Titanic

VisiRule File:
- c:\users\clive\documents\misc\vrs\cs1_acpm_v5\vrs\cs_titanic0.vrs
- c:\users\clive\documents\misc\vrs\cs1_acpm_v5\vrs\Untitled-0
- c:\fastchart\data\cs_titanic0.vrs

Start goal:
- start1

2] Choose to generate some XML which can be executed using a small JS engine

![VSR to HTML](image)

- Convert c:\fastchart\data\annotated_cs_titanic0\vrs
- Filename: c:\fastchart\data\annotated_cs_titanic0\HTML
1] VisiRule Server will process the chart and generate an interactive web app with an SVG view of the chart:

Now we can answer questions to see if any specific instance would likely have survived!
Once we have answered all the questions, we will get a conclusion/prediction (with the associated counts)

We can see the Trail and session transcript
We can see the path taken:
2] If we choose to generate XML + JS then we get something more compact and responsive:
In addition, we can export the executable rules in the chart to process external data sets. We may want to see how the predicted conclusions compares with the actual historical data.

We can visualize how our chart performed using the aggregated paths overlaid on the chart.
We can also deliver the rulebase an interactive ChatBot

**TITANIC ChatBot**

Sex?
- female
- male

Number of siblings/spouses aboard?
- > 2
- <= 2

Enter fare paid