The Algorithm of 10,000 Variables
A Case Study On Machine Learning in Software Assessment

Cobec Consulting Inc.

May 3rd, 2021
Agenda

- Introduction
  - Who We Are
  - Project Background
- Methodology
- Data Source
  - Examples Data Pipeline
- Models
- Experimental Results
- Conclusion
Introduction - Morgan Heimbrock

Morgan Heimbrock is a Consultant with Cobec Consulting supporting the acquisition oversight group at the Federal Aviation Administration’s Office of Investment Planning and Analysis. Currently, she is helping with financial data aggregation, software development, and product management. Ms. Heimbrock has a Bachelor of Science in Finance and Accounting from the University of Kentucky.
Bryan Anderson is a Consultant with Cobec Consulting focusing on OR and software engineering in the Federal Aviation Administration’s AFI and AJW-2000 offices. Mr. Anderson’s academic background includes a BA in Economics & Mathematics from Augsburg College and MS in Industrial and Systems Engineering from the University of Minnesota - Twin Cities. Mr. Anderson has presented at various conferences and engaged in the ICEAA community. Prior to being working in the federal government, he was an Industrial Engineer at a national grocer optimizing supply chain operations.

Figure 2: Bryan Anderson
Introduction - Project Background

- FAA owns and operates equipment across the whole National Airspace System (NAS). The FAA Logistics Center (FAALC) manages the NAS inventory and distribution of needed parts. FAALC provides:
  - support for more than 48,000 systems
  - management of 62,000 National Stock Numbers
  - inventory value of more than $900 million
  - customer support of over 8,000 FAA, DoD, State, and foreign country facilities

- There is a need to enhance the FAA supply chain operations by replacing legacy logistics support system with a COTS ERP. Due to the complexity of supply chain operations, changing needs of NAS needs, and consolidation of information incurred impediments in program implementation.
With delayed requirements, program functional and non-functional needs continued to evolve causing software rework. Assessment of the cost and schedule of this software work was critical for planning but difficult in practice with traditional methods.

Function point analysis was initially performed, but thought not to be able to capture all the work required due to non-functionality rework and data quality. SLOC was also not readily available and not a practical approach.

This project attempts to use a novel approach to assess resource needs for software programs using actual data collected by the development’s Issue Tracking System (i.e. Jira).

Extracting this historical data, several statistical models where trained and used to predict developer level work activities.
Methodology - Overview

- General approach has two major components:
  - Data Engineering
    - Data Pipeline - Data is extracted from the source and goes through several steps to transform, validate, and formatted for analytics.
    - Feature Extraction - Feature extraction and engineering is done at each step. Each step provides more rigorous methods to generate features.
  - Statistical Modeling - Several models were generated using an advanced analytical tool. Model results were analyzed for quality and enhancement options.
Methodology - Data Pipeline

Figure 3: Data Pipeline
Methodology - Data Pipeline Descriptions

1. **ITS Export**
   - Initial Jira data exported as CSV file. Contains raw data such as issue type, issue comments, individuals assigned to roles, status, creation date, resolution date etc.

2. **Enhanced ITS Export (EITSE)**
   - Even though it was standard report from Jira, report structure wasn’t standard among the several exports. E.g. comments on an issue would generate a new column. Data was transformed in Excel using VBA and formulas to standardize data structure to combine the several reports.
3 Java Feature Extraction

- A Java program parses the EITSE and performed another round of feature extraction. Using Snowball evaluated basic text token extraction (token identification and stemming) and other data transformations (e.g. parse strings to timestamps).

4 Relational Database

- The parsed data from the previous step was then persisted in a relational database. More feature engineering was done in this layer such as encoding of categorical data. This layer provided a way to create a standard way to create initial feature sets with stored procedures and data lineage.
Analytical Tool

Feature sets created in the RDBS is then brought into a analytical tool for advanced statistical analysis. Here several models were trained, validated, analyzed, and forecast remaining unresolved issues.

Schedule Analysis

The net outcome from the previous step is prediction of unresolved issues. To account for resource constraints (e.g. can’t work all unresolved simultaneously), results were simplified into 12 week durations and limits imposed on number of issues that can be resolved per month.
Final Analysis

Findings from the Schedule Analysis is then combined with SEER-SIM using a heuristic to formulate the schedule estimate.
Data Source

- ITS provides project members to plan and organize work activities. In this use case, the popular Atlassian Jira system was used.
- Developers ‘work against’ issues. Issues contain the work that needs to be done.
- ITS captures historical activity on the developer’s work in these issues.
- Issues are atomic units of work. Issues contain data elements such as:
  - Types with standardized workflows
  - Textual description of the work (summaries, description, comments).
  - Attachments
  - Specific individuals assigned to roles (assignee, creator, commentator, watcher).
  - Specific timestamps (creation, resolution, comments)
- Development frameworks (e.g. SAFe) are desired for mature workflows. This makes data more predictable and reliable.
Data Source Example

First draft of WIM in python

Description
Need a first draft of WIM in Python to help support various Jira analytics.
- Cleanup the analysis directory so git is clean.
- Draft Scikit pipelines to process, model, and forecast.

Activity
Show: Comments  History

Figure 4: Example Jira Ticket
Figure 5: Example Enhanced ITS Export 0
Figure 6: Example Enhanced ITS Export 1
Data Source Example - RDBS Example

Figure 7: RDBS Example
Figure 8: RDBS Schema
The objective of the analysis is to predict how long it will take a developer level work item to be resolved. Thus, the label of the models was the time from the creation to the resolution date. Data is then binned by quantiles\(^1\) of the labels to create classes for prediction.

RapidMiner (RM) was the analytical tool\(^2\) used to evaluated the statistical models. RM can create prediction models from among seven prominent algorithms: Naive Bayes, Generalized Linear Models, Fast Large Margin, Decision, Tree, Random Forrest, Gradient Boosted Trees, and Support Vector Machine). RM also provides further sophisticated-automatic feature engineering, model validation, and tools to analyze model results.

---

\(^1\)Bins have equal number of issues.

\(^2\)Using an advanced analytical tool vs development using coding languages saved many labor hours.
The age of an unresolved ticket influences prediction of it’s total life-time. This is because - all else being equal - a younger ticket has less developed attributes\(^3\). To account for this, three age-appropriate modeling groups were created for each of the algorithms. These models where trained with data that has been “alive” for a minimum amount of time.

\(^3\)Fewer comments, attachments, individuals connected, etc. Analogy used was predicting life expectancy of a person based on a resume. Less the resume has, more likely the individual is younger.
Models - Groups (cont. 2)

Model group 1 - Issues that are considered young.
Model group 2 - Issues that were considered middle aged.
Model group 3 - Issues that were considered old.
Models - Groups (cont. 2)

- Model group 1 - Issues that are considered young.
Models - Groups (cont. 2)

- Model group 1 - Issues that are considered young.
- Model group 2 - Issues that were considered middle aged.
Models - Groups (cont. 2)

- Model group 1 - Issues that are considered young.
- Model group 2 - Issues that were considered middle aged.
- Model group 3 - Issues that were considered old.
Models - Train Process

Figure 9: RapidMiner Train Process
Figure 10: RapidMiner Apply Process
Models - GBT Confusion Matrix

accuracy: 78.49% +/- 2.88% (micro average: 78.49%)

<table>
<thead>
<tr>
<th></th>
<th>true range1</th>
<th>true range2</th>
<th>true range3</th>
<th>true range4</th>
<th>class precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred. range1</td>
<td>214</td>
<td>23</td>
<td>13</td>
<td>6</td>
<td>83.59%</td>
</tr>
<tr>
<td>pred. range2</td>
<td>37</td>
<td>122</td>
<td>17</td>
<td>5</td>
<td>67.40%</td>
</tr>
<tr>
<td>pred. range3</td>
<td>13</td>
<td>17</td>
<td>160</td>
<td>30</td>
<td>72.73%</td>
</tr>
<tr>
<td>pred. range4</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>157</td>
<td>89.71%</td>
</tr>
<tr>
<td>class recall</td>
<td>79.26%</td>
<td>73.94%</td>
<td>80.40%</td>
<td>79.29%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 11: GBT Example Confusion Matrix

Figure 12: GBT Example Confusion Matrix Plot
### Figure 13: GBT Prediction Example

<table>
<thead>
<tr>
<th>Row No.</th>
<th>ftd</th>
<th>prediction(ftd)</th>
<th>confidence(range1)</th>
<th>confidence(range2)</th>
<th>confidence(range3)</th>
<th>confidence(range4)</th>
<th>prj</th>
<th>deploy_me</th>
<th>evc</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>range4</td>
<td>range4</td>
<td>0.006</td>
<td>0.149</td>
<td>0.160</td>
<td>0.585</td>
<td>LCSS</td>
<td>MISSING</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.040</td>
<td>0.138</td>
<td>0.817</td>
<td>LCSS</td>
<td>f</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.042</td>
<td>0.217</td>
<td>0.736</td>
<td>LCSS</td>
<td>f</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>range4</td>
<td>range4</td>
<td>0.010</td>
<td>0.042</td>
<td>0.129</td>
<td>0.819</td>
<td>LCSS</td>
<td>f</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>range4</td>
<td>range4</td>
<td>0.006</td>
<td>0.056</td>
<td>0.240</td>
<td>0.698</td>
<td>OCD</td>
<td>MISSING</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>range4</td>
<td>range4</td>
<td>0.006</td>
<td>0.052</td>
<td>0.143</td>
<td>0.800</td>
<td>OCD</td>
<td>MISSING</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.041</td>
<td>0.132</td>
<td>0.822</td>
<td>LCSS</td>
<td>MISSING</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.044</td>
<td>0.253</td>
<td>0.699</td>
<td>LCSS</td>
<td>f</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>range4</td>
<td>range3</td>
<td>0.007</td>
<td>0.172</td>
<td>0.543</td>
<td>0.277</td>
<td>OCD</td>
<td>f</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>range4</td>
<td>range3</td>
<td>0.006</td>
<td>0.070</td>
<td>0.241</td>
<td>0.683</td>
<td>LCSS</td>
<td>MISSING</td>
<td>65</td>
</tr>
<tr>
<td>11</td>
<td>range4</td>
<td>range4</td>
<td>0.288</td>
<td>0.048</td>
<td>0.202</td>
<td>0.462</td>
<td>LCSS</td>
<td>f</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>range4</td>
<td>range1</td>
<td>0.630</td>
<td>0.048</td>
<td>0.131</td>
<td>0.191</td>
<td>LCSS</td>
<td>f</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.041</td>
<td>0.142</td>
<td>0.812</td>
<td>LCSS</td>
<td>f</td>
<td>7</td>
</tr>
<tr>
<td>14</td>
<td>range4</td>
<td>range1</td>
<td>0.436</td>
<td>0.051</td>
<td>0.172</td>
<td>0.341</td>
<td>OCD</td>
<td>f</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>range4</td>
<td>range4</td>
<td>0.006</td>
<td>0.063</td>
<td>0.270</td>
<td>0.661</td>
<td>LCSS</td>
<td>f</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>range4</td>
<td>range4</td>
<td>0.006</td>
<td>0.056</td>
<td>0.201</td>
<td>0.738</td>
<td>LCSS</td>
<td>f</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>range4</td>
<td>range1</td>
<td>0.516</td>
<td>0.045</td>
<td>0.142</td>
<td>0.197</td>
<td>LCSS</td>
<td>f</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>range4</td>
<td>range1</td>
<td>0.005</td>
<td>0.040</td>
<td>0.110</td>
<td>0.836</td>
<td>LCSS</td>
<td>MISSING</td>
<td>27</td>
</tr>
<tr>
<td>19</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.041</td>
<td>0.187</td>
<td>0.767</td>
<td>LCSS</td>
<td>f</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>range4</td>
<td>range3</td>
<td>0.008</td>
<td>0.242</td>
<td>0.498</td>
<td>0.252</td>
<td>LCSS</td>
<td>MISSING</td>
<td>18</td>
</tr>
<tr>
<td>21</td>
<td>range4</td>
<td>range3</td>
<td>0.005</td>
<td>0.113</td>
<td>0.168</td>
<td>0.714</td>
<td>LCSS</td>
<td>f</td>
<td>6</td>
</tr>
<tr>
<td>22</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.040</td>
<td>0.120</td>
<td>0.830</td>
<td>LCSS</td>
<td>f</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>range4</td>
<td>range4</td>
<td>0.141</td>
<td>0.044</td>
<td>0.127</td>
<td>0.688</td>
<td>LCSS</td>
<td>f</td>
<td>7</td>
</tr>
<tr>
<td>24</td>
<td>range4</td>
<td>range1</td>
<td>0.651</td>
<td>0.039</td>
<td>0.121</td>
<td>0.179</td>
<td>LCSS</td>
<td>f</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>range4</td>
<td>range1</td>
<td>0.006</td>
<td>0.058</td>
<td>0.356</td>
<td>0.580</td>
<td>OCD</td>
<td>MISSING</td>
<td>6</td>
</tr>
<tr>
<td>26</td>
<td>range4</td>
<td>range3</td>
<td>0.015</td>
<td>0.256</td>
<td>0.506</td>
<td>0.223</td>
<td>LCSS</td>
<td>f</td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td>range4</td>
<td>range4</td>
<td>0.009</td>
<td>0.056</td>
<td>0.279</td>
<td>0.657</td>
<td>LCSS</td>
<td>f</td>
<td>18</td>
</tr>
<tr>
<td>28</td>
<td>range4</td>
<td>range4</td>
<td>0.005</td>
<td>0.056</td>
<td>0.279</td>
<td>0.657</td>
<td>LCSS</td>
<td>f</td>
<td>18</td>
</tr>
</tbody>
</table>
Models - GBT Tree Example

Figure 14: GBT Tree Example
Models - GBT Tree Example Zoom

Figure 15: GBT Tree Example Zoom
So far, the pipeline has predicted creation to resolution duration for each unresolved issue (in weeks). This was not yet ready for the schedule and cost models. The schedule in particularly needed to account for limited (and competing) resources to work on issues. This final analysis determine a single forecast number and guidance of limit on number of issues that can be completed in a time period.

Since each modeling group has somewhat different class ranges, the results where simplified into duration categories that are multiple of 12-weeks (nominally a calendar quarter).
The duration forecast is for the entire issue life-time, including pre-work (e.g. process engineering, requirements refinement, waiting that occurs before ticket is allocated to a sprint, testing).

To translate forecast to quarters:

1. Determine the min and max of the model groups class range.
2. Evaluate the PERT formula.
   \[ \frac{\text{min\_range}}{2} + 4 \times \frac{\text{min\_range} + \text{max\_range}}{2} + \text{max\_range} \]
3. Evaluate the ceiling of the prior results divided by 12.
To provide guidance on a constraint on issue resolutions in a time period, analysis on the historical Jira database provide statistics on resolutions per month:

- Mean = 119 resolutions/month
- Standard deviations = 42 issues/month
- Median = 120 resolutions/month

Working with the developers, an assumption that for every known unresolved issue, another one will be created to satisfy the original issue’s intent. So, in practice this number of reduced by half. This information was then passed onto the schedule and cost estimators. The final analysis consisted of a custom heuristic that combined the results with SEER-SEM.
Experimental Results - Data Summary & Key Findings

- Models were trained over 4,000 issues and predicted over 1,000 issues.
- Over 1,000 features were extracted and derived from historical data.
- Examples of model features included issue summary, description, comments, number of times issue was commented, issue type, specific roles fulfilled by an individual, all individuals associated with the issue (commented, mentioned in text), and others.
- Gradient Boosted Trees turned out to be the most accurate trained model.
- Individuals' roles and affiliated with the issue seem to be the most important feature.
Conclusion - Future Considerations

- Usage of the more detail historical changes retrieved via the API.
- Model enhancements to:
  - accommodate issue current age
  - refine feature sets
  - simplify pipeline and manual steps
  - port analytical tool’s work to tool that doesn’t require licensing
- Develop program specific productivity model driven by Function Point size and by non-Function Point Analysis attributes.
- Determine if we can measure the quality of data.
- Refine and formalize model outputs for tracking results and facilitate the hand-off to next estimators.
Goal was to create greater fidelity in a cost and schedule estimate.

- A unique approach was took to help estimate non-Function Point elements of the project.
- Data was processed through a standard and rigorous pipeline.
- Several models where generated and evaluated for use.
- Model forecasts had a final analysis to make results more useable for schedule and cost estimates.
- These durations aided in the development of a schedule and cost models needed in the acquisition process.
Gradient Boosted Trees is a popular algorithm for modeling various problems. There is much to learn about this approach and how to use it properly. This quick example is just for helping understanding the ideas in this method. This example is from Cory Maklin from Medium. The example looks at predicting housing prices based on a few features.

References:


4This example problem seems to be a popular way to explain GBT. Several blogs had the exact or similar datasets and explanation.
### GBT Example - Artifact Data

<table>
<thead>
<tr>
<th>Age</th>
<th>Sqr Footage</th>
<th>Location</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1500</td>
<td>5</td>
<td>480</td>
</tr>
<tr>
<td>11</td>
<td>2030</td>
<td>12</td>
<td>1090</td>
</tr>
<tr>
<td>14</td>
<td>1442</td>
<td>6</td>
<td>350</td>
</tr>
<tr>
<td>8</td>
<td>2501</td>
<td>4</td>
<td>1310</td>
</tr>
<tr>
<td>12</td>
<td>1300</td>
<td>9</td>
<td>400</td>
</tr>
<tr>
<td>10</td>
<td>1789</td>
<td>11</td>
<td>500</td>
</tr>
</tbody>
</table>
GBT Example - Calculate Average of Target

\[
\frac{480 + 1090 + 350 + 1310 + 400 + 500}{6} = 688
\]
From the previous step, calculate the residuals by subtracting the actual price from the average.

\[ \text{residual} = \text{actual\_price} - \text{avg\_price} \]

<table>
<thead>
<tr>
<th>Age</th>
<th>Sqr Footage</th>
<th>Location</th>
<th>Price</th>
<th>Residuals</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1500</td>
<td>5</td>
<td>480</td>
<td>-208</td>
</tr>
<tr>
<td>11</td>
<td>2030</td>
<td>12</td>
<td>1090</td>
<td>402</td>
</tr>
<tr>
<td>14</td>
<td>1442</td>
<td>6</td>
<td>350</td>
<td>-338</td>
</tr>
<tr>
<td>8</td>
<td>2501</td>
<td>4</td>
<td>1310</td>
<td>622</td>
</tr>
<tr>
<td>12</td>
<td>1300</td>
<td>9</td>
<td>400</td>
<td>-288</td>
</tr>
<tr>
<td>10</td>
<td>1789</td>
<td>11</td>
<td>500</td>
<td>-188</td>
</tr>
</tbody>
</table>
GBT Example - Construct Decision Tree

Figure 16: First Decision Tree
GBT Example - Construct Decision Tree Caveat

In the event of multiple residuals in a leaf, compute averages. Thus the prior decision tree is:
GBT Example - Predict Target

\[ \text{pred\_price} = \text{avg\_price} + \text{learning\_rate}\times\text{residual} \]

\[ 654.2 = 688 + 0.1x - 388 \]
GBT Example - Compute New Residuals

Calculate new set of residuals by subtracting actual from predictions made in the previous step.

$$ \text{residual} = \text{actual\_price} - \text{pred\_price} $$

$$ -304.2 = 350 - 654.2 $$

Repeat these steps for configurable amount (either iterations or stopping criteria in Hyperparameters).
Final prediction will equal the first step’s mean plus the residuals (and learning rate).

\[
pred\_price = avg\_price + learning\_rate \times \sum_{i}^{n} r_i
\]
GBT Example - Regression vs Classification

This GBT example is a regression problem. For Classification, the general approach is the same, but instead of calculating residuals of the label, the label is probabilities of belonging in the class (i.e. binary classification). For multi-class, N-models are generated for each class and the model with the highest probability is chosen.

References:

Scikit-learn defines hyper-parameters as parameters that are not directly learnt within the models. Generally hyper-parameters for models are the same among various implementations (e.g. R, Python, analytical tools) but can differ. Hyper-parameter configuration is a full analysis in itself.

Note, RapidMiner uses the H20 GBM implementation.

References:

- docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html