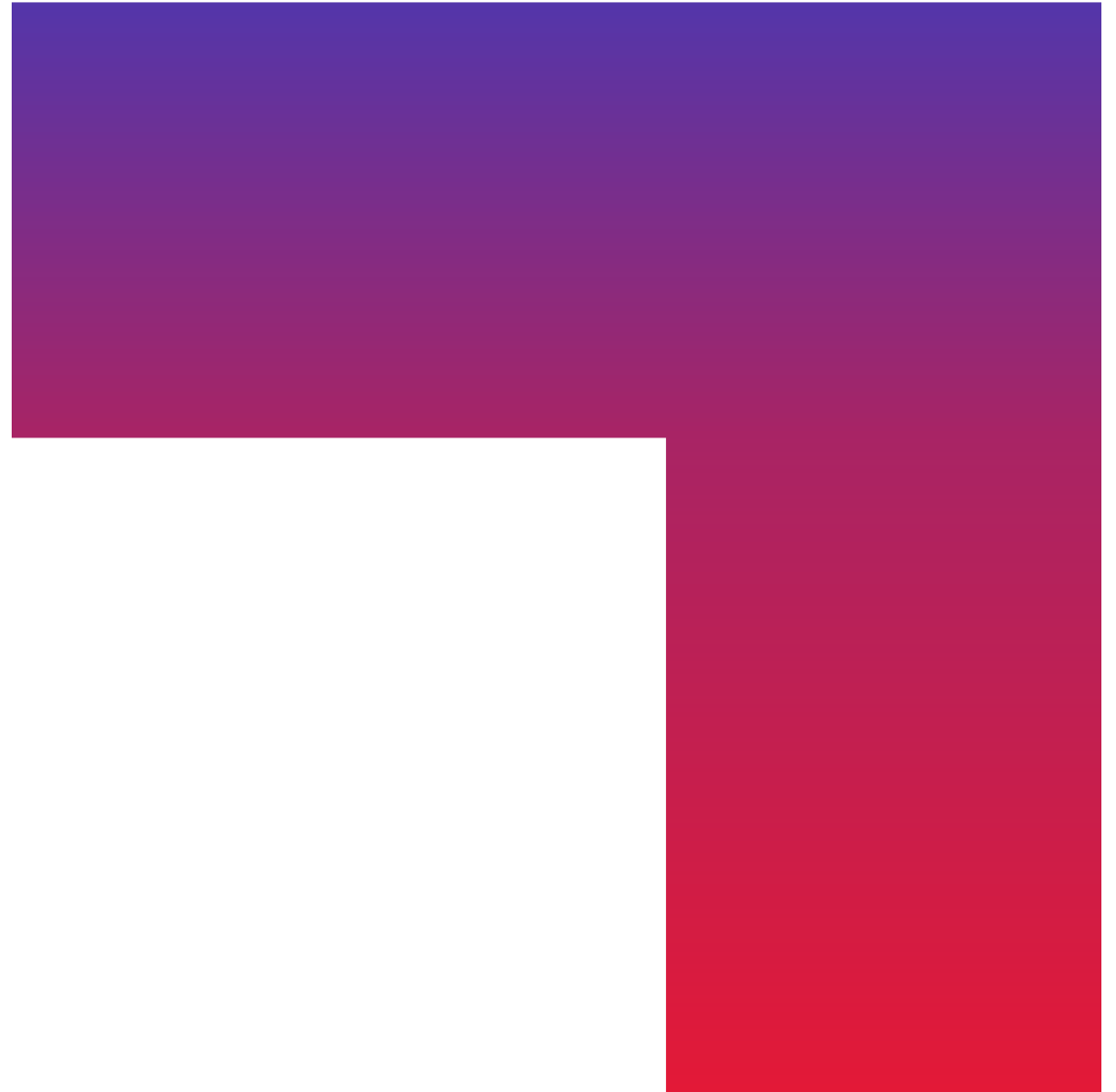


The Power of ML for Application Maintenance

Eric van der Vliet / George Thomas
September 2024



Objectives



“To improve Application Maintenance estimates, the analysis of historical data is essential”

Because the data is mainly categorical, a proper Machine Learning model need to be used to provide data capable to predict the effort for new tickets

Agenda

1 Application Maintenance

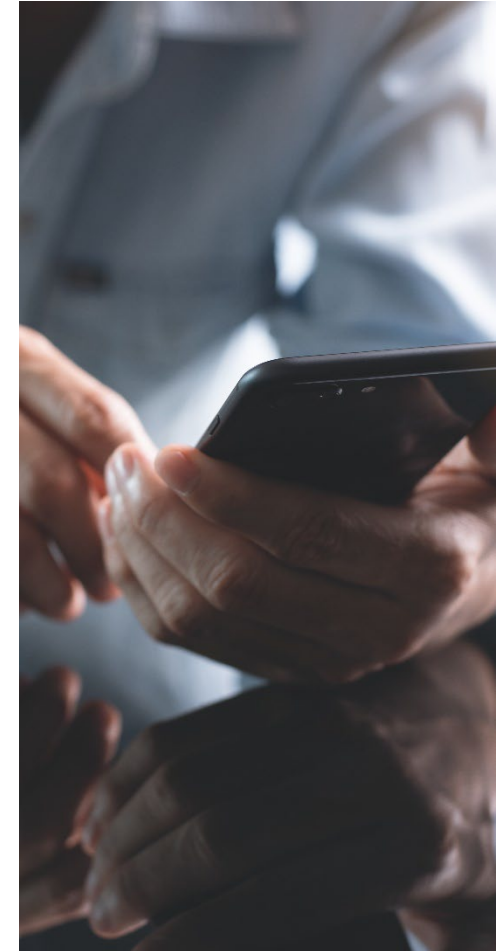
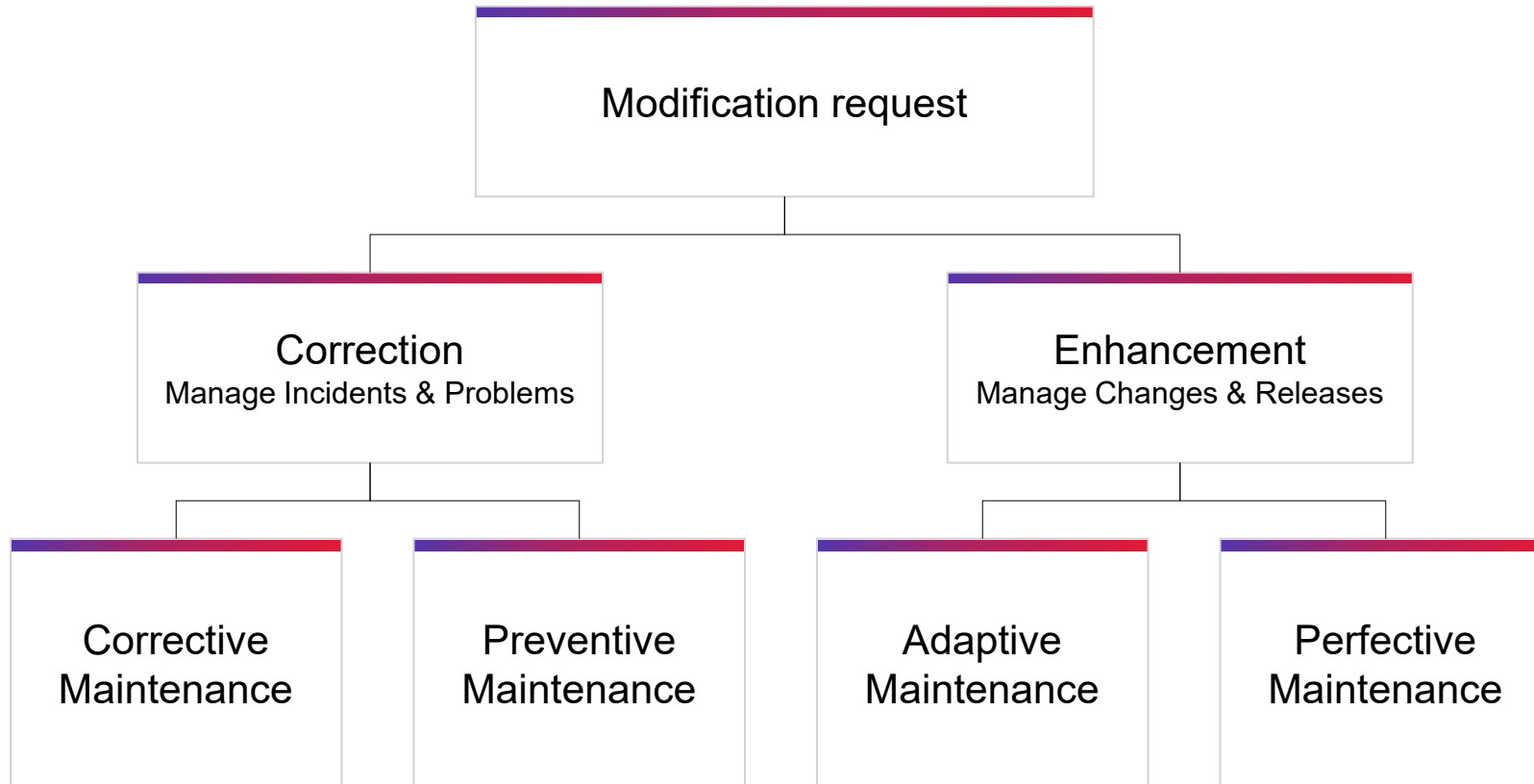
2 Ticket data

3 Data analysis / Machine Learning

4 Summary



Application Maintenance – Categories



Application Maintenance estimation – Process steps



Ticket data – Effort per ticket

- The expected number of tickets to be received
- Ticket data can be corrected based on assumption
- Historical data shall be used to determine the required effort per ticket
- Assignment of the resources per location and determining the costs



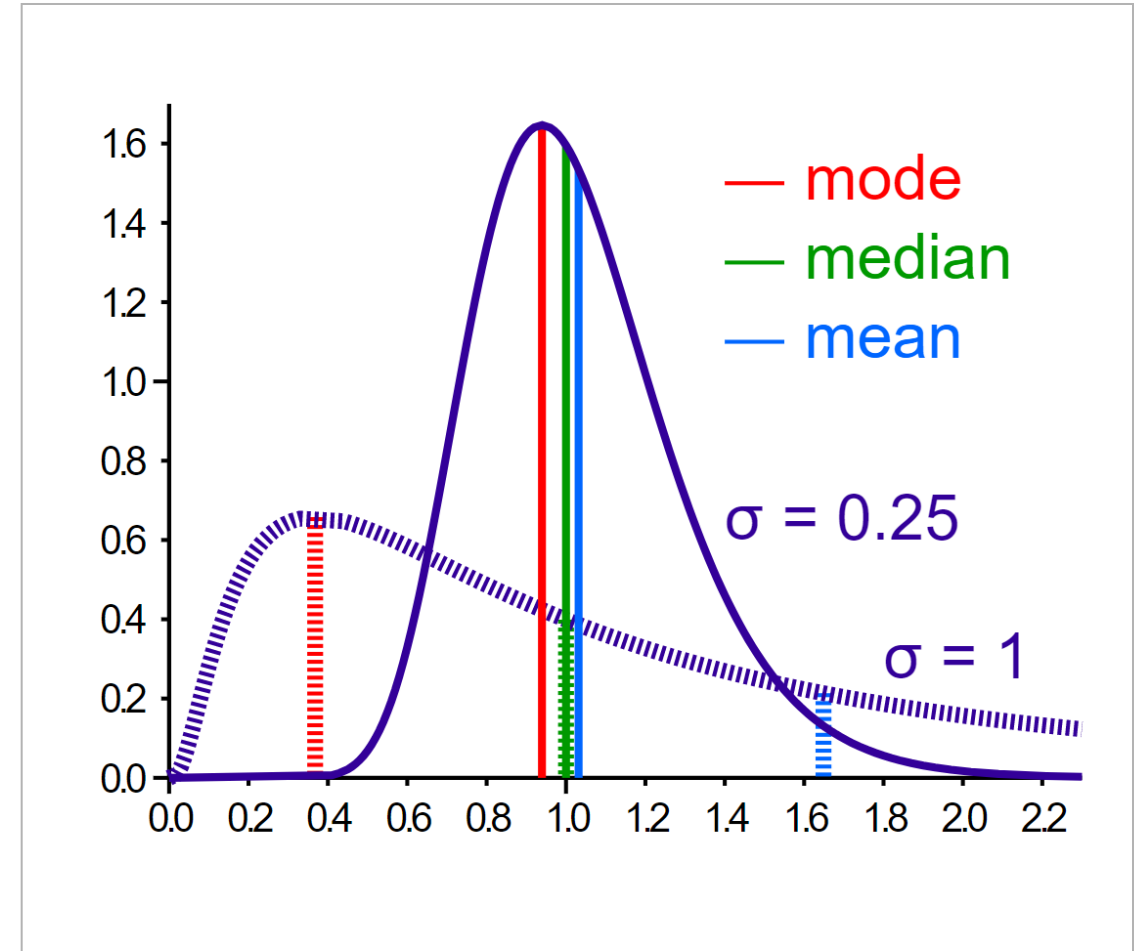
Service data – Effort for services

- ➔ Historical data is required to determine the required effort for additional services
- ➔ The data is preferably analyzed as a percentage of the ticket data effort



Data analysis – Distribution of historical data

- Be careful with right skewing data and averages
- Using averages can give wrong (too high) results
- Right skewed data can be caused by outliers
- Outliers could be incorrect registrations
 - Registration of multiple tickets in one
 - Registration accuracy



Data Analysis and Machine Learning overview

Objective

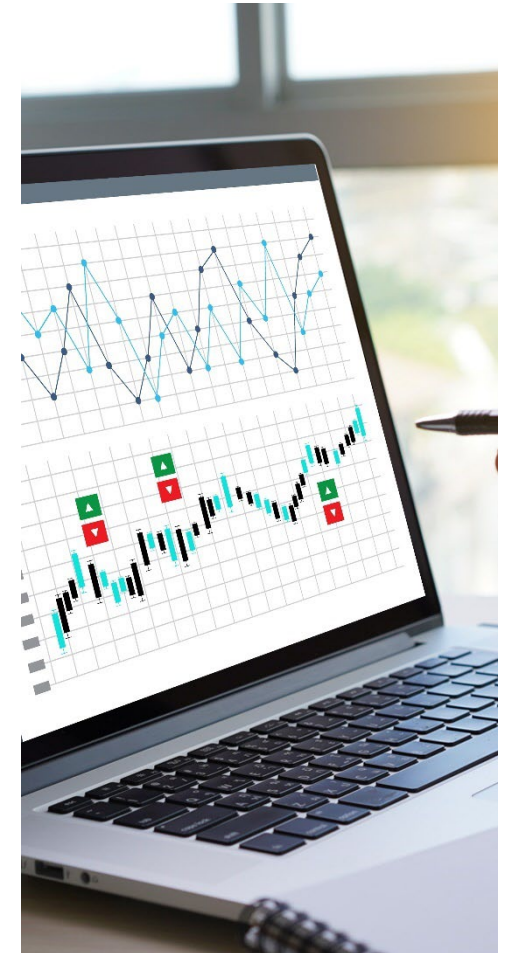
- Accurate prediction of effort spent on various kinds of incident tickets across Business Units, Engagements and Technologies

Solution

- Semi-automated model and feature selection with several models evaluated such Linear Regression, Gradient Boosting, Random Forest, Neural Networks among others
- Automated report generation highlighting key views from data analysis for quick insights to data
- Technologies used – Python, Pandas, NumPy, Scikit

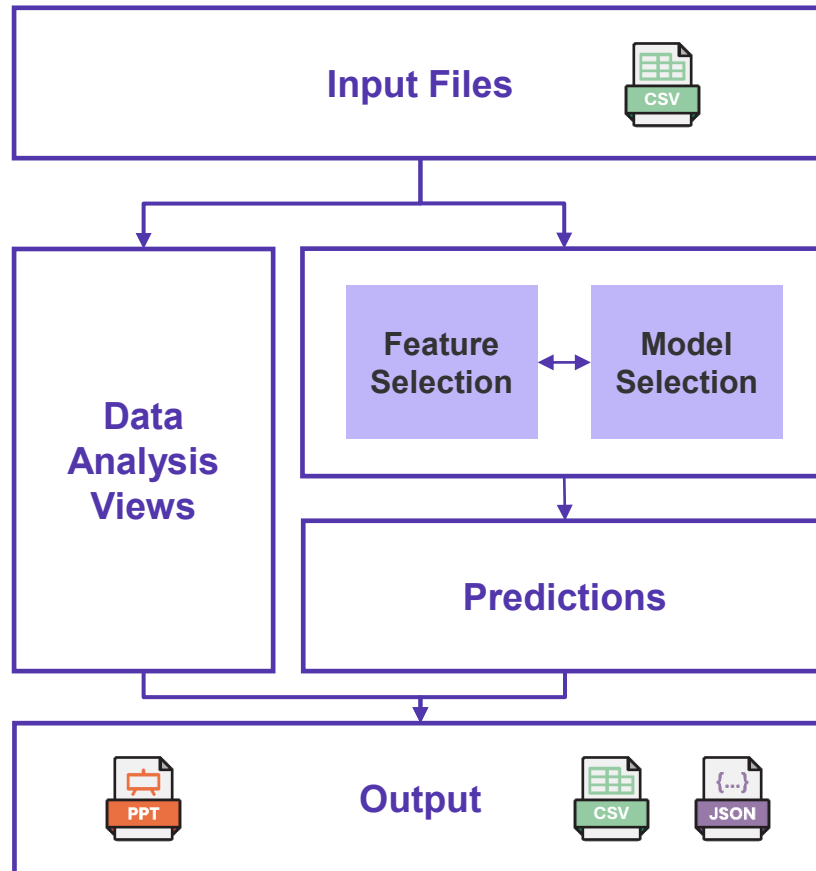
Results

- Automated, on-demand data analysis and report generation
- Predictions at an “engagement - ticket type – priority” level
- Additional features can be included for a further granularity like year, technology, etc.
- Predictions close to actuals (off by ~5%) as opposed to using simple measures like mean or median (~25% off)



Solution Approach

High-level Solution Diagram



Model Evaluation and Selection (Sample)

	Model Score	Root Mean Square Error
Linear Regression	0.200	11.1
Gradient Boosting	0.398	9.6
Support Vector Regression	- 0.085	12.9
Random Forest	0.395	9.7
XGBoost	0.394	9.7
LightGBM	0.397	9.6
CatBoost	0.404	9.6
Neural Networks	0.126	11.6

Benefits of approach

- Semi-automated approach – manual feature selection with automated model selection to find best-fit model

Challenges

- Since we have only categorical predictors, model scores are generally low
- Predictions are generally better when we have many records for a specific combination

Adding additional features (data points)



Technologies used

Use of technology data to further break down the granularity of the predictions

Improved model scores and closer predictions (~ 5-10%)



Year-on-year improvement

Use of creation year from the ticket date as an additional feature to improve the yearly ticket-wise predictions



Additional Features

Additional data points like the results of Code Quality tools, average experience / tenure of members in the engagement may improve accuracy

Model selection approach - 1

Analysis of categorical fields:

- Issue Type
 - Identifies the type of issue logged while creating the support ticket
- Priority
 - classification of tickets based on urgency
- Severity
 - Classification of tickets on complexity
- Technology used
 - Most relevant technology stack used in the issue
- Domain
 - industry functional area that the product/project serviced
- Business unit
 - geographical unit the project is based out of
- Time Spent
 - dependent variable; contains the resolution time



Application Services

To evaluate the effectiveness of different models we calculated the R-squared and Root Mean Square. Our approach was to improve R-squared score while minimizing Root Means Square Error (RMSE)

1

Linear and Gamma regression models

In addition to linear regression, we tried Gamma Regression as the data is extremely right-skewed

However, the model metrics/ performance wasn't very promising for analysis

2

Support Vector Regression

We explored the use of SVR as our data is right skewed with significant outliers.

Model metrics were not significantly better than Linear & Gamma regression

3

Gradient boosting models

Since we have categorical features only, we explored gradient boosting models like XGBoost, LightGBM and CatBoost

All gradient boosting models had better metrics; finalized CatBoost as it is optimized for categorical data

With CatBoost, we were able to consistently achieve 95% accuracy on our predictions

Improving the model by removing outliers

Removing outliers

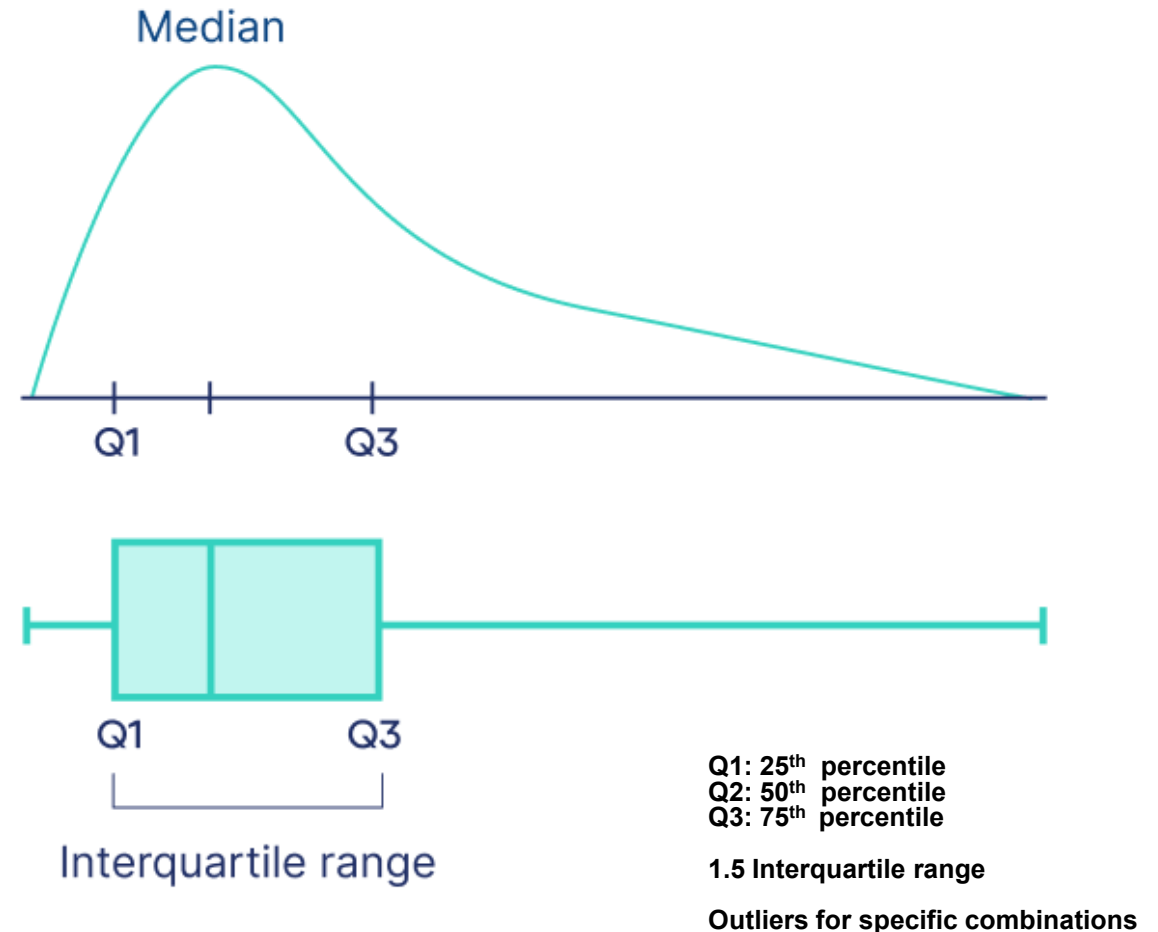
With a few tickets, the time spent does not represent actual resolution time

- Ticket being closed after a long time due to inactivity
- Missed updating closure status, etc.

These values tend to be very big (over 10 – 100 times the median in some cases). It did not make sense to include these for analysis

We use the **Interquartile Range (IQR) approach** to identify and remove outliers for analysis as our data was right-skewed

Removing outliers improved the analysis by **5 to 10 percentage points**



Leveraging technical and domain knowledge to improve the model

Avoiding Sparse Matrices for Machine Learning

- Technologies that were “similar” are grouped
- Two approaches are used
 1. Commonly grouped technologies based on technology stacks
 - MEAN stack, MERN stack; etc.
 2. Similar database technologies grouped together
 - Oracle, MySQL, DB2, etc.
- In our analysis, this **halved** the number of columns in encoded data table for analysis
- We also combined columns to create combinations of the data to reduce the overall number of encoded columns using the context of data
- Sparse matrix: A matrix where most values are zero
- MEAN stack: Java script based framework
- MERN stack: integrates JavaScript technologies to facilitate the creation of full-stack web applications



Leveraging technical and domain knowledge to improve the model

Issue ID	Incident Type	Priority	BU	Engagement	Technology
1001001	Incident	Critical	Telecom	Telecom 1	SQL
1001002	Incident	High	Telecom	Telecom 1	DB2
1001003	Incident	Low	Oil & Gas	Oil Major 1	Oracle
1001004	Problem	Medium	Oil & Gas	Oil Major 2	JAVA
1001005	Problem	High	IP	Oil Major 1	JAVA
1001006	Problem	Low	IP	Oil Major 2	CRM

Leveraging technical and domain knowledge to improve the model

Issue ID	Incident Type	Priority	BU	Engagement	Technology
1001001	Incident	Critical	Telecom	Telecom 1	SQL
1001002	Incident	High	Telecom	Telecom 1	DB2
1001003	Incident	Low	Oil & Gas	Oil Major 1	Oracle
1001004	Problem	Medium	Oil & Gas	Oil Major 2	JAVA
1001005	Problem	High	IP	Oil Major 1	JAVA
1001006	Problem	Low	IP	Oil Major 2	CRM

After regular categorical encoding: all categorical values are converted into columns for ML

Issue ID	Incident	Problem	Critical	High	Low	Medium	Telecom	Oil & Gas	IP	Telecom 1	Oil Major 1	Oil Major 2	SQL	DB2	Oracle	JAVA	CRM
1001001	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0
1001002	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
1001003	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
1001004	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0
1001005	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0
1001006	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1

Leveraging technical and domain knowledge to improve the model

Issue ID	Incident Type	Priority	BU	Engagement	Technology
1001001	Incident	Critical	Telecom	Telecom 1	SQL
1001002	Incident	High	Telecom	Telecom 1	DB2
1001003	Incident	Low	Oil & Gas	Oil Major 1	Oracle
1001004	Problem	Medium	Oil & Gas	Oil Major 2	JAVA
1001005	Problem	High	IP	Oil Major 1	JAVA
1001006	Problem	Low	IP	Oil Major 2	CRM

After regular categorical encoding: all categorical values are converted into columns for ML

Issue ID	Incident	Problem	Critical	High	Low	Medium	Telecom	Oil & Gas	IP	Telecom 1	Oil Major 1	Oil Major 2	SQL	DB2	Oracle	JAVA	CRM
1001001	1	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0
1001002	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
1001003	1	0	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0
1001004	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0
1001005	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	1	0
1001006	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1

Generating a smaller matrix through optimization:

1. Dropping one value from each category
2. Dropping Telecom since it has only one Engagement which is unique to Telecom
3. Grouping SQL, DB2, Oracle into one single 'Database' group

Issue ID	Incident	Critical	High	Low	Telecom 1	Oil Major 1	Oil Major 2	Database	JAVA
1001001	1	1	0	0	1	0	0	1	0
1001002	1	0	1	0	1	0	0	1	0
1001003	1	0	0	1	0	1	0	1	0
1001004	0	0	0	0	0	1	0	0	1
1001005	0	0	1	0	0	0	1	0	1
1001006	0	0	0	1	0	0	1	0	1

Summary




Application Maintenance

A clear definition of the scope of Application Maintenance is required to determine what data is needed for analysis. This will not only consist of ticket data but also data from additional services



Data

Data need to be collected from ongoing engagements where the right level of details of the data and the right set of features is important (e.g. technology, domain).



Analysis

For the analysis it's important to understand the type of data that is available. For Application Maintenance this is often Categorical data what requires applicable models for analysis. The right analysis results in a high predictability.

Questions?



Insights you can act on

Founded in 1976, CGI is among the largest IT and business consulting services firms in the world.

We are insights-driven and outcomes-based to help accelerate returns on your investments. Across hundreds of locations worldwide, we provide comprehensive, scalable and sustainable IT and business consulting services that are informed globally and delivered locally.

cgi.com



CGI