ML in Benefit Decisions

Technical aspects

Riku Sarlin
Chief Architect
Kela, Benefits and Customer Service
@Rsarln

Kela|Fpa

Contents

- Use case
- Data used
- Architectural choices
- Data quality and data pre-processing
- Models and their performance

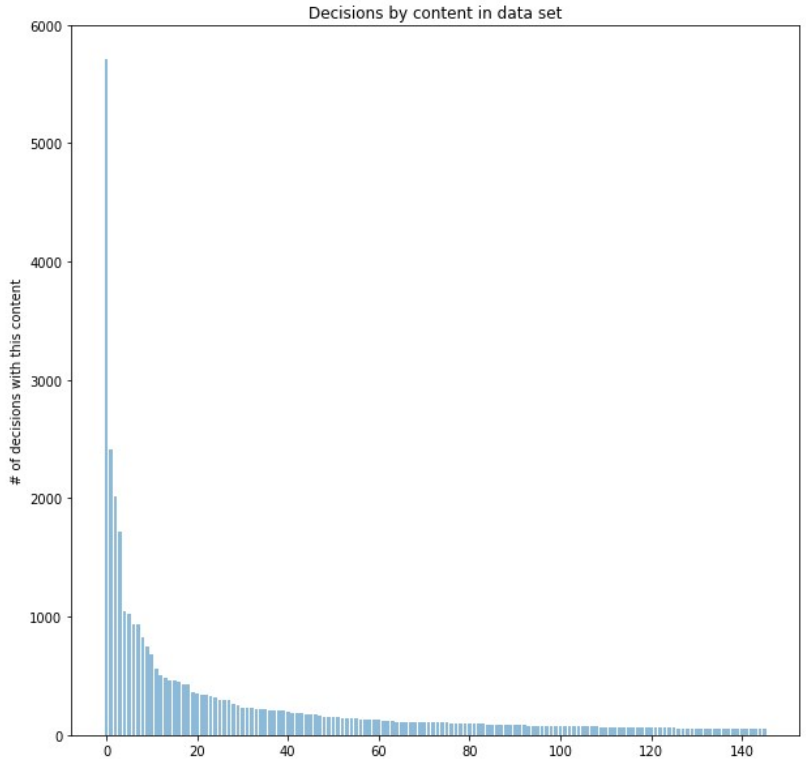
Use Case

- Benefit decisions in mainframe systems can be represented as a series of phrase codes, such as "C70,C60,C69,C72,C64,C65,C74,Y10,M02,Z01,S03,D11, V01,U03,U04,J02,Q25,Q04,Q01,Q02,Q02,Q06,Q07"
- When the actual decision document is crated, these phrase codes are filled with data from the application, other data sources and handling system
- By studying the actual decision documents one can realize that a limited number of input variables affect the contents of the documents
- Still, currently the phrase codes are input largely manually. A typical housing benefit application takes 10 minutes to handle, and there are 400000 applications per year. Any reliable form of automation could make business sense.

Kela|Fpa

Data Used

- Some data (phrases, outcome, income types, other benefit types and subtypes, household member size) was fetched from DW staging area for all 2018 housing benefit applications Preliminary study show that variation across all housing benefits is huge - 357405 decisions made in 2018 produce *93207 different decisions!*
- When limited to new applications only, we have 65947 decisions in 12420 categories.
- Some decision contents are much more popular than others – to the right, 146 most popular decision contents (with over 50 cases)
- Decisions with over 50 cases cover 56% of all applications



Architectural choices

- Data was fetched from DW
  - Readily available
  - Pseudonymized
  - High quality
- Processing took place in IBM Cloud
  - Open source tools commonly available in-house, too (Jupyter, Python, pandas, matplotlib, scikit-learn, Keras, TensorFlow). The results can be utilized in-house, too.
  - Possibility to utilize SystemML / Apache Spark to process data in multiple servers – was not needed, though. We don't have functioning SystemML in-house at the moment.

Data Quality and Pre-processing

- Data is pseudonymized data from operative systems (which contain a lot of data integrity checks). Data is of very high quality!
- We had to do quite a lot of processing with pandas to turn the SQL result set into a Pandas DataFrame suitable for modeling.
  - Grouping
  - Filtering
  - Combining columns
  - Encoding in forms suitable to modeling
- Most of the data features were of categorical nature. These have to be ”one-hot encoded” for the algorithms to work.
- Not all classification algorithms are good with many categories, some work naturally with binary classification, but can be extended to multiclass

Models and their performance

- As performance indicators, f1 score (SVM) and accuracy (neural network) were used. Scikit-learn and Keras with TensorFlow backend were used. All of these do a splendid job!
- From business point of view, all miscategorizations here are just as bad (wrong content in benefit decisions)
- SVM performer poorly (f1 score 0,29-035) – this is mainly due relatively low amount of examples (65k) when compared to number of categories (148) and skewed nature of the data (some very popular categories)
- On this background, the performance of simple neural network – 99,4% accuracy – is phenomenal. In addition, training time was not that bad. This is typically problematic for neural network applications.

Model Algorithm / neural network

- The neural network topology was quite simple
  - Input layer: 124 neurons (4 features encoded, ”relu” activation)
  - Drop layer, 20% dropped (for regularization)
  - Hidden layer: 136 neurons, ”relu” activation
  - Drop layer, 20% dropped (for regularization)
  - Output layer: 148 neurons (147 categories + 1 ”not in the other categories” category, ”sigmoid” activation)
- Hyperparameters
  - Loss: binary_crossentropy, ”adam” optimizer, batch size 64
- Training took place in free tier of IBM Cloud, Keras/TensorFlow
- In training 20 epochs, 3 seconds each, would be enough
- Surprisingly, we get the same result even without drop or hidden layers…
- We get the same results also with batch sizes from 20 to 2000 – it just much faster with bigger batch size.

Thanks for watching!

Riku Sarlin
Chief Architect
Kela, Benefits and Customer Service
@Rsarln

Kela|Fpa