How one user became familiar and comfortable with Machine Learning

… and developed some product classification abilities

… and made every possible mistake

along the way

About the user

- Hello! I am Claude Julien, the Machine Learning project manager
- Before that, I led the Methodology Direction at Statistics Canada from which I retired after a 33+ year career
- My background and expertise are in statistics, survey methodology and quality assurance
- Prior to managing the ML project, I had no knowledge about it
- I gained some knowledge through interactions with the great people that contributed to the numerous studies and developments on the project
- Being curious by nature and having some time on my hand, I decided to get my hands dirty and other parts of my brain working to learn how to use ML myself with the code and data shared by Statistics Poland and the kind, patient and knowledgeable assistance of colleagues on the project
- What follows are my experiences in becoming familiar with ML, making mistakes and learning all the way
- The ultimate goal to my experiences was to see if and how ML could be integrated in a manual operation to achieve better results at a lower or similar cost

Outline

- My objectives and means to achieve them
- Getting started and first lessons learned
- Cleaning the data with the help of ML and more lessons learned
- Simulating the introduction of ML into a coding operation and more lessons learned
- Some caveats and last words

My objectives and means to achieve them

- Get familiar and comfortable with machine learning
  - Experiment with the ML code and grocery product description data shared by Statistics Poland
  - Count on the knowledge, experience and patient assistance of experts in the ML group
- Propose a scheme to integrate ML in an existing process
  - Assisted by the ML code, identify classification errors and inconsistencies in the data
  - Combine my long-acquired methodology, statistics and quality assurance background with my “acquired on-the-fly” ML/Python knowledge and my “by-the-way” burgeoning grocery product coding skills
  - Simulate, on a small scale, the integration of ML into a fictitious well established, yet not perfect, coding process
- Add to the learning material delivered by the ML project
  - Document my experiences (this is the short version)
  - Share lessons learned and other thoughts as I executed my experiments

Getting started – ML code and data

- I have the ML code and product description data shared by Statistics Poland
- The data contains over 16500 product descriptions to which Statistics Poland assigned a ECOICOP class
  - The class was not reviewed by coding experts
- The ML code allows me to run four different ML methods
  - One code is used to assess different combinations of parameters
    - This code delivers a set of key measures for each set of parameters
  - One code is used to run the ML method with the selected parameters
    - This code delivers class predictions and a likelihood of being accurate associated to each prediction
    - It also delivers a set of measures of accuracy and a confusion matrix derived from a subset of records

Getting started – My points of interest

- I want to see the results that come out of ML; in particular, how accurately in can predict the product class (accuracy rate)
- I am less interested in how ML produces its predictions or in the selection of parameters; most combinations seem to me to produce the similar measures; i.e. I am impatient to see the results
- I run the ML code for the Logistic Regression method with the recommended split (70% for training; 15% for validation; 15% for testing) and the <u>parameters suggested by Statistics Poland pilot study</u>
    - Training subset is used to fit the model.
    - Validation subset is used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.
    - Test subset is used to provide an unbiased evaluation of a final model fit on the training dataset.

Getting started – My first results

- In this first run, the predicted class is accurate when it is equal to the initial class provided on the shared dataset
- From the testing subset with 2488 products, I observe that:
    - 90.0% of the predicted classes are accurate
    - Since the testing subset is a random sample from the whole dataset, I can also measure that this estimated accuracy rate can range from 88.9% to 91.1% (confidence interval)
    - That is good but not great
- I could try to tinker with the ML parameters or to preprocess the data to improve the overall accuracy, but I also observe that:
    - 52.8% of the predicted classes have a likelihood value above 0.90 and show a 98.9% accuracy rate
    - 65.9% of the predicted classes have a likelihood value above 0.80 and show a 97.8% accuracy rate
    - The predicted classes with a likelihood value between 0.50 and 0.80 have an accuracy rate above 80%
- These results indicate that there is likely something that I can get from ML predictions

Getting started - Lessons learned

- Sharing and collaborating makes it much easier for me to quickly experiment with ML
- Having a random sample of product descriptions (testing subset) is important to obtain an informative, statistically sound and credible measure of accuracy
- Assessing the performance of ML on the overall results is not sufficient, i.e. the performance of ML is not “all or nothing” or trying to get to the “all”, but rather what one can do with the “something” in between
- Hence, I am now interested in:
  - The percentage of class predictions with a high likelihood of being accurate
  - The percentage of these high-likelihood predictions that are actually accurate (accuracy rate)
- The ML renders good results; are they good enough? To answer that question, one has to have a baseline or objective against which to compare the ML results

Product class changes – Help from ML

- In analyzing the results of the first run, I noticed some incorrect or inconsistent product classes in the dataset, e.g.
  - coffee product classified to “Tea” (incorrect)
  - two products with very similar descriptions classified to different codes (inconsistent)
- I improvised an expert review of product classes
- I focussed my attention on the products in the testing subset with a predicted class with a high likelihood value that differed from the initial class on the dataset

Product class changes – Number of changes

- Several runs of ML and reviews of initial and predicted classes lead me to proposing corrections to some of the initial classes
  - There are 89 changes to products in the testing dataset (3.6%)
  - There are 529 changes to products on the whole dataset (3.2%); most them are the results of querying the whole dataset on situations observed in the testing subset, e.g. check all products with the word “coffee” classified to Tea
- I now have a reference (the corrected classes) against which I can assess and compare the accuracy of both the ML predicted classes and the initial classes on the dataset

Product class changes – First impact

- A class (predicted or initial) is now accurate when equal to the corrected class value
- The accuracy rate of the ML predicted classes is not better (90%); in fact, there is one less accurate predicted class in the testing subset (I show why later)
- The estimated accuracy rate of the initial class is 96.9% with a confidence interval of 96.2% to 97.5%
- Among the products with a predicted class with a likelihood value above 0.90, the accuracy rate of the initial classes and the predicted classes are the same (98.9%)
- Among the products with a predicted class with a likelihood value above 0.80, the accuracy rate of the initial classes is slightly higher (98.3% vs 98.1%)

Product class changes – Second impact

- The results of the class changes do not improve the added value of ML
- One problem is that the model is built on the initial class values
- Building the model on the corrected class values produce the following results
  - The accuracy rate of the ML predicted classes is now better (92.4%), but still lower than 96.9% with the initial classes
  - There are more predicted classes with a likelihood value above 0.80 (67.4%)
  - Among the products with a predicted class with a likelihood value above 0.90, the accuracy rate of the predicted classes is now higher (99.5%) than the initial classes (97.7%)
  - Among the products with a predicted class with a likelihood value above 0.80, the accuracy rate of the predicted classes is also higher (98.1%) than the initial classes (95.2%)
- Generally, the ML model is more confident and more accurate, and starts to show some potential added value

Product class changes – Micro impacts (1/2)

- It is interesting to see some of the dynamics between the ML predicted classes, initial and corrected classes; the following are observations on the products in the testing subset
- After the changes to the classes only (no update to the ML model), the ML predicted classes did not change, but:
    - Some ML predictions went from being inaccurate (not equal to the initial) to being accurate (equal to the correct) – ML was correct
    - Some ML predictions went from being accurate (equal to initial) to being inaccurate (not equal to correct) – ML learned from inaccurate initial
    - A few ML predictions are not accurate (not equal to initial) and are still inaccurate (not equal to correct) – Three-way disagreement

Product class changes – Micro impacts (2/2)

- After the changes to the classes and the update to the ML model, most ML predicted classes did not change and similar situations than those on the previous slide were observed
- Some ML predicted classes did change
  - Some products went from having “Old prediction = initial ≠ correct” to “New prediction = correct ≠ initial” – ML learned better from better training data
  - Some products went from “Old prediction ≠ initial ≠ correct” to “New prediction = correct ≠ initial” - ML learned better and resolved 3-way disagreement
  - Some products went from having “Old prediction ≠ initial = correct” to “New prediction = initial = correct” – ML learned better
  - Some products went from having “Old prediction = initial = correct” to “New prediction ≠ initial = correct” – Loss of accuracy
- Changes to product classes had a positive direct impact (showing that predicted is correct) and indirect impact (enabling ML to predict more correctly), but they also incurred some inaccurate ML predictions; but overall, the gains significantly outweigh the losses

Product class changes – Lessons learned (1/2)

- No process is flawless
  - Coding products (even relatively simple ones) is not always easy, even for humans; this very likely leads to some level of inaccuracy in manual coding; similar products can be classified differently by different coders, or even the same coder at different occasions
- ML predictions enabled me to identify misclassified products or lead me to group of products that were likely misclassified; ML also helped me to learn about product classification and quickly exposed me to certain of its challenges
- In order to have an assessment of the added value of ML, it is essential to compare it to a target objective or an alternative process
- In order to produce this comparison, it is essential to count on an expert and independent review of both processes (ML and current)
- The use of a random sample enables a statistically sound, credible and cost efficient comparison between the ML and an alternative process

Product class changes – Lessons learned (2/2)

- The objective in using ML should be to improve the process, not replicate it
- Having good quality training, validation and testing data is very important to the performance of ML
  - Accurate testing data is obviously essential to adequately assessing the accuracy of ML versus a current process
  - More accurate training and validation data makes things more clear to the ML, as it would also do to humans, and enables the ML to make better predictions
- It is important and interesting to see and understand what more accurate training data can bring, but one must look at it from a global/macro perspective and accept some losses at the micro/record level
- Just this basic experience highlights the essential interactions between subject-matter, operations, IT, statistics and methodology expertise to fully and adequately assess the potential value added of any new process, such as the use of ML

Using ML to achieve better accuracy and lower cost

- I now have a dataset of product descriptions with an initial class and a corrected class
  - The initial classes are 96.8% accurate according to my amateur-expert review assisted by ML
  - I recognize that the dataset is somewhat biased in the sense that several inaccurate initial classes were identified with the help of ML predictions
- With the realistically imperfect dataset at my disposal, I run a simulation in the following fictitious context
- A well-established coding operation is responsible for assigning ECOICOP class value to product descriptions
- The accuracy of their work is regularly assessed from a random sample of the classified products that are reviewed by subject-matter and coding experts
- I attempt to introduce the use of ML in this operation in the following manner

Simulation – Month 1 operations

- The operation is tasked with classifying 3000 products per month
- In the first month, all products are classified by the coding operation
    - Cost is 3000 classifications
- Afterwards, I split the production data in three subsets: 70% for ML training; 15% for ML validation; 15% to assess the accuracy of the ML predicted values and the coding operation values (testing subset)
    - The coding operation values are the initial classes in my dataset
- I run the ML like I did in the previous runs, but on 3000 products rather than the whole dataset
    - The coding operation and the ML predicted classes from the testing subset are assessed against the correct classes (from the class changes)
    - Cost is 450 expert classifications

Simulation – Month 1 results

- The accuracy of the coding operation is 96.7% with a confidence interval from 95.1% to 98.2%
- The accuracy of the ML predictions is 80.4% with a confidence interval from 77% to 83.9%
- 28.5% of the predicted classes have a likelihood value above 0.80, and all but one of them is inaccurate; however, the classes from the coding operation are all accurate
- The accuracy rate of the predicted classes with a likelihood value between 0.40 and 0.80 is over 95%
- Encouraged by these results, I decide to take some cautious risk and further integrate ML into the operation in the following manner
- The total cost of the production run is 3450 classifications and some IT time (baseline cost)

Simulation – Month 2 operation

- I take the ML model from the first month to predict a ML class on the 3000 products in this month's production
- I decide to accept the ML predictions with a likelihood value above 0.80 (**high enough predictions**) without sending them to the coding operation
- I decide to assist the coding operation by providing them with ML predictions with a likelihood value between 0.50 and 0.80 (**good enough predictions**)
    - There are several ways of doing this in practice that I can't simulate adequately
    - For the simulation, I make the key assumption that when a coder combines his/her knowledge and the ML prediction, she/he takes the best decision, i.e. if either the initial class or the predicted class is equal to the correct class, the coder will choose the correct one
    - I let you judge, where this assumption stands between realism and optimism

Simulation – Month 2 expert review

- I still select 15% of the production to be reviewed by experts
- The classes from the coding operation or the ML prediction are compared to the expert (correct) classes
- To ensure a complete and independent review, the products in the testing subset with a high enough prediction must be sent to the coding operation to be classified “blindly”, i.e. without providing the ML predicted class, to independently assess how accurately they will be classified by the coding operation

Simulation – Month 2 results

- 30.8% of the products have a ML predicted class with a high enough prediction (efficiency)
  - Their accuracy is higher than the classes provided by the coding operation
- Estimated accuracy and confidence interval over all products in the testing subset:
  - ML predicted: 83.1% (79.9% to 86.4%)
  - Coding operation: 94.9% (93.0% to 96.8%)
  - Coding operation + ML predicted: 96.9% (95.4% to 98.4%)
- Efficiency: 12% less coding operation
- Outcome: better accuracy at a slightly lower cost

Simulation – Month 3 to 5 operation

- Every subsequent month, the setup used in month 2 is repeated with more and more data
- Combine the training, validation and testing subsets, respectively, from the previous months; e.g. training subset of month 3 = training subset from months 1 and 2
    - The training and validation subsets now include some ML predicted classes accepted without any review (more accurate training data => better learning)
    - The testing subset is solely used to monitor accuracy of the ML model and coding operation; it is used neither for training, nor validation
- Rebuild the model every month to predict the product classes
- Use the high enough predictions to automatically classify products and the good enough predictions to assist the coding operation

Simulation – Month 1 to 5 results

- After each month’s production, the ML model is rebuilt by appending its training, validation and testing subsets to those from the previous months
  - The following slide shows performance measures on the ML Model and the coding operation calculated from the testing subset
- At the start of each month’s production (except Month 1), the ML model predicts a class for each product
  - The next slide shows performance measures on the predicted classes and the combination of the coding operation with the predicted class
  - The performance measures are calculated on the whole production (efficiency) and the testing subset (accuracy)

Simulation – Month 1 to 5 Model results

Performance of the ML model based on the products in the testing subset						
Month	# products	% with high enough predictions	% accuracy among high enough predictions		% accuracy among all products	
			Model	Operation	Model	Operation
1	450	28.4%	99.2%	100.0%	80.4%	96.7%
2	900	49.0%	98.6%	98.0%	86.1%	95.8%
3	1350	55.3%	98.7%	98.1%	88.1%	96.2%
4	1800	61.2%	98.5%	98.0%	89.2%	96.2%
5	2488	67.2%	98.5%	97.8%	90.8%	96.4%
Notes:	The ML model for month M is used to predict the classes for the month M+1 products					
	The "high enough likelihood" are the products with a predicted class with a likelihood value above 0.80					

- The % of high enough predictions steadily increases (ML is more confident)
- The high enough predictions are more accurate than the coding operation (ML is steadily reliable)
- Over the whole testing subset, the ML model improves but never reaches the coding operation (ML can’t work alone)

Simulation – Month 2 to 5 Production results

Performance of the ML predictions								
	Whole production			Testing subset				
Month	# products	% with high enough predictions	Efficiency	Accuracy		Difference	Confidence interval	
				Operation	Operation + ML	Estimate	Lower limit	Upper limit
2	3000	30.9%	12%	94.9%	96.9%	2.0%	0.8%	3.2%
3	3000	46.3%	26%	97.1%	98.0%	0.9%	-0.1%	1.9%
4	3000	55.7%	34%	96.0%	97.6%	1.6%	0.2%	2.9%
5	4586	60.3%	38%	97.1%	97.7%	0.6%	-0.4%	1.6%
Notes:	The "high enough likelihood" are the products with a predicted class with a likelihood value above 0.80							
	When the confidence interval of the difference does not cover the value "0", the two accuracy estimates are significantly differ							

- The efficiency steadily increases
- Integrating ML into the operation increases accuracy
  - Again, recognizing that a part of this is due to the assumption mentioned on a previous slide
- The increase in accuracy is statistically significant in two of the four production runs
- The simulation demonstrates the added value of ML (more accuracy at lower cost)

A first quirk in my simulation

- After running my simulation using the Logistic regression method, I realized that I had run it with its default parameters, not the ones proposed by the Statistics Poland pilot study
- The results on the first month production showed a very different distribution of prediction likelihood values (much less values above 0.90; much less values lower than 0.10)
- At the end of this re-simulation, I observe (compared to using the default parameters):
  - Slightly more efficiency (more predictions with likelihood above 0.8)
  - A distribution of likelihood values that evolves towards the one with the default parameters (but still fewer predictions with likelihood above 0.9, but much more with likelihood above 0.8)
  - Slightly more accurate predictions, especially in the earlier months
  - However, the coding+ML classes are more accurate in the first three months, but not the last two
    - The accuracy on the last month production is barely higher than the coding operation only
- I interpret this as ML taking a slightly different path to get to a similar pleasant destination (should that reassure or worry me?)

Second quirk in my simulation

- The last observation on the previous slide bothers me a bit
- Looking at some testing products, I notice that some obvious products with the word “crisps” (“Chrupki” in Polish) are predicted as Other bakery products by the ML (ECOICOP clearly classes them as Crisps)
- Further investigation shows that the problem started in the first month when 6 of 7 “crisps” products were classified as Other bakery products by the coding operation
  - the ML became pretty confident that crisps products were in Other bakery products
  - Even if this error was picked up among a few products in the testing subset, there is no direct mechanism to feed this information back to the ML modelling
  - This information can be fed back to the coding operation who would start classifying these products correctly which would eventually make it to the ML model
  - The other possibility is that these crisps/chrupki products should actually be classified as Other bakery products and I am wrong

Final simulation

- Just to see, I ran a final simulation in which the whole month 1 production underwent an expert review to ensure that the first ML model was built on the best possible data
- At the last month production, both the efficiency and accuracy were slightly higher (not surprising)
- Over the five production runs, the combined efficiency was still quite positive

Lessons learned 1/2

- ML can add value to a coding operation; it should not be considered simply to replicate the performance of an existing operation
- ML cannot completely replace a coding operation
- Integrating ML into a coding operation should produce more accuracy at a lower cost; the extent of this added value depends on the quality, quantity and complexity of the data being classified and the classification system
- Combining the expertise of informatics, statistics, sampling and QA methodology, subject matter, operations and data science is essential to achieve the most added value
  - Subject matter and operations are essential to producing the best quality data to train and monitor the performance of ML and its integration in the coding operation
  - Methodology is essential to ensure a cost efficient, sound and credible assessment of the added value (accuracy and cost)
  - IT is essential to ensuring the development and maintenance of a robust and error-free production system
  - Statistics and data science is essential to develop efficient, understand and communicate their results
- Relying on one or even just a few individuals to cover all these expertise is extremely prone to errors and loss of time

Lessons learned 2/2

- It is essential to focus on the performance of the overall operation (coding + ML) and accept a few errors at the record level
- ML can be a valuable tool in training coding staff
- ML likely requires some time to deliver its added value; it will deliver benefits in the medium or longer term, as long as it is closely monitored
- It is very important to start with a good quantity of high quality data to train the ML
- Choosing a good set of parameters, i.e. data science, and some understanding of how ML produces its predictions, i.e. some explainability, is important (more than I thought at the beginning)

Other contributions and lessons learned

- In sharing some of this work and asking a few questions, I received the following feedback
- (from Claus Sthamer) It really shows how important a good “Ground Truth” or “Golden Standard” data sets are. Do you know the extent to which the products with likelihood value belong to minority classes? If I remember correctly from the StatsCanada pilot study and implementation, they code items belonging to minority classes manually. Which seems to be the same conclusion in your work.
  - My reply: It seems to be the case
- In response to the question: Do I need to have the validation subset reviewed by experts as well?) (Reply from Alex Measure; It’s solely a matter of priority. The problem with expert coding is that it’s very expensive so we have to decide where we will use it. We use it only for test data because we consider the questions answered by the test data to be the most important, specifically:
  - How accurately do we expect the system to assign codes in production?
  - How does this accuracy compare to the accuracy of the alternative (likely manual) process.
  - If we prioritized other things, like finding the best possible model, we might use expert coding for training or validation instead, but the improvements we’re likely to get from that are small and we don’t think that’s nearly as important as having a good idea of how well the system works.

Caveats on the simulation

- Many of the classification errors assumed in the dataset were identified by the analysis of ML predictions; this likely introduces some bias in favor of ML; the question is how much is "some"?
- The cost derived excluded IT costs; in this simulation, the IT cost were negligible because of the code shared by Statistics Poland and its ease of use
  - It just took a list of products and initial ECOICOP values; no pre-processing

Last words… for now!

- The value of this work is not as much on the results than on the approach taken to integrate ML into an operation, e.g. gradual integration with statistically sound monitoring of all processes
- I hope that it highlights the needs and interactions between several expertise
- I hope that it can provide a link between the Statistics Poland pilot study and coding operations assisted by ML setup or being setup in other organisations, e.g. Statistics Norway, Statistics Netherlands