Milan Sherman
DSC 680
Project One
Milestone 2

Business Problem: most users who come to our site generate a quote for life insurance before starting an application. The only information a user provides at quote is age, height, weight, gender, and whether or not they use tobacco. We use Body Mass Index (BMI) based on height and weight to put a user into a risk class, which along with gender drives pricing. The problem is that based on this limited information, over half of our users are put into the best risk class, but less than 15% stay in this risk class after their application is approved. As a result, most users see a substantially larger price after being approved than they were shown at quote, and analysis has shown that this influences users' decision to purchase. In addition, this has ethical implications as users likely take into account whether or not to complete an application based on the quote that they receive.

History:

On average, users are seeing an approved price that is 50-60% higher than the one they were shown at quote. By grouping users by the price that they see after being approved, we saw a much better conversion rate for users who did not see a price increase across price groups. If the issue was just the price, i.e., users seeing a larger increase are likely also the users seeing a larger price, we would have expected to see that difference dissipate as the price increased. However, we see the conversion rate difference persist across price groups, indicating that the difference does influence users' decision to purchase.

We are currently implementing a short-term solution of showing all users, regardless of risk class, a price at quote that is 24% higher than what our system generates based on BMI. Although A/B testing has shown that it has been effective at improving conversion rates, this is a hacky, one-size fits all solution that is showing our best users a price at quote that is higher than what they would see after being approved, and thus could be deterring them from starting an application. An ideal solution would involve finding a better prediction of approved risk class at quote, since this the is driver of price. Right now, only 29% of users are in the same risk class once approved as they were when they received a quote, so there is a fairly low bar to beat to improve on what we're currently implementing.

A major factor in our application approval process is the data vendor calls that we make to obtain verified information on users' identity, criminal history, and medical history. These data vendor calls are the main expense in processing applications and are only made after a user submits an application. The challenge is identifying user inputs in the application that could potentially be moved to the quote portion of the user experience to better predict approved risk class earlier in the process.

Research Questions:

1. Can we create a model that is better at predicting a user's approved risk class at quote than our current system?
2. If so, what data is needed beyond what we currently collect at quote to achieve this?
3. While accurately predicting a user's approved risk class at quote is the ultimate goal, can we also reduce the error for users that we make incorrect predictions for? We have a total of 10 risk classes, and the further away a user is at approved from where they were at quote, the larger the price difference they will see. So, in addition to making more accurate predictions, can we also reduce the size of the error, and therefore the price discrepancy, for users that the mode fails to correctly predict?

Data

Until December of last year, we were selling policies underwritten by Sammons Financial Group (sfg), but started our own life insurance company (Bestow Life Insurance Company (BLIC) in December and started selling policies nationwide in January. Thus, we only have three months of data for our insurance company, which is likely not enough to get a good model, or at least would be a confounding factor in the model's performance. So, I decided to use all of the sfg data from 2022 and create a proof of concept, i.e., if I can answer the research questions with sfg data, then we have good reason to believe that the same approach would work for own insurance company when we have sufficient data.

When considering the data for this model, I considered two alternatives:

1. Create a model with the data that we currently collect at quote and get a baseline, then add features that we currently don't collect until users complete the application.
2. The kitchen sink approach: throw in every feature we could possibly collect from the user during the application (excluding features that related to vendor calls which would not happen until the application is submitted). The strategy of this approach is to try to get the best possible model to start with and then use feature importance to identify a subset that could reasonably be collected at quote.

I opted for the second approach, reasoning that it would be easier to recursive feature extraction to whittle down the features than to iteratively add features, especially when it may be hard to identify which features will be most useful to the model. Thus, the first step of data collection was to identify any fields in our application table that could be reasonably asked of a user during at the quote step of the user experience. This means that any fields related to the results of data vendor calls could not be used. As there are over 700 fields in our application table, this took some time. I ended up with 173,557 rows and 151 columns for my initial dataset prior to cleaning.

The issue with this approach was that it made getting the data to a point that it could be used to train a model was a major lift in this project, due to a couple of factors:

1. The data is input by the user, and they can input whatever they want. This could be an error, or just users who don't want to share accurate information for some reason. For example, I found users who claimed to have 5000 alcoholic drinks per week.
2. We have revised our application a few times, and have retained historical data for all users, so there are duplicate fields, or fields which have similar information (but not exactly the same).

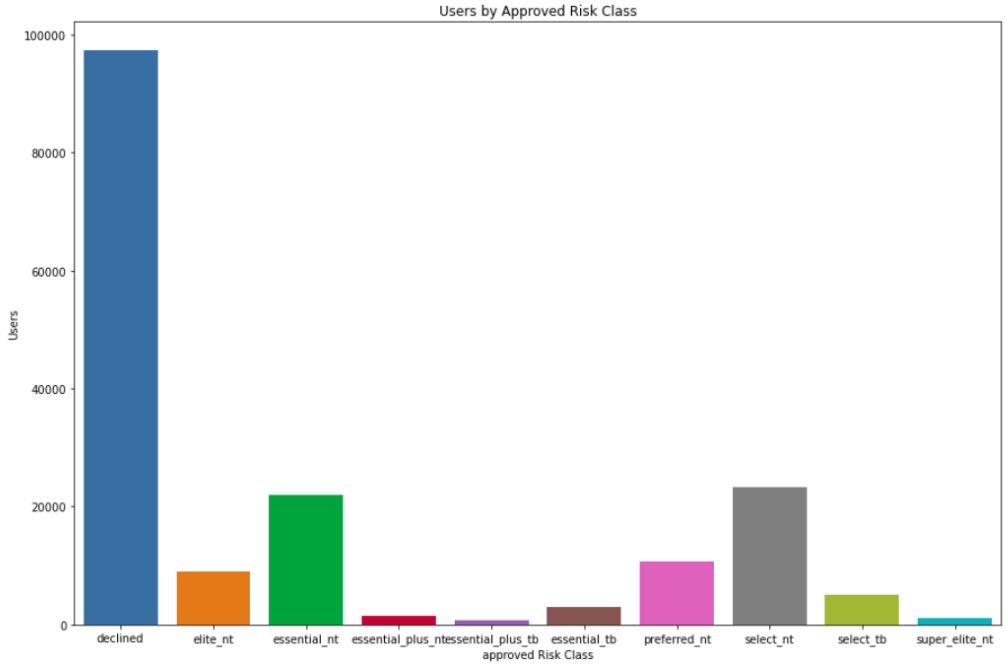
For example, I found six fields related to alcohol consumption, and this was not an isolated incident (I found similar issues related to stroke and seizures, for example). These fields will be highly correlated, so I didn't want to keep them all, but it took some time to explore which were most informative and relevant. In addition, I found 53 fields that had no non-null values and thus could be dropped altogether.

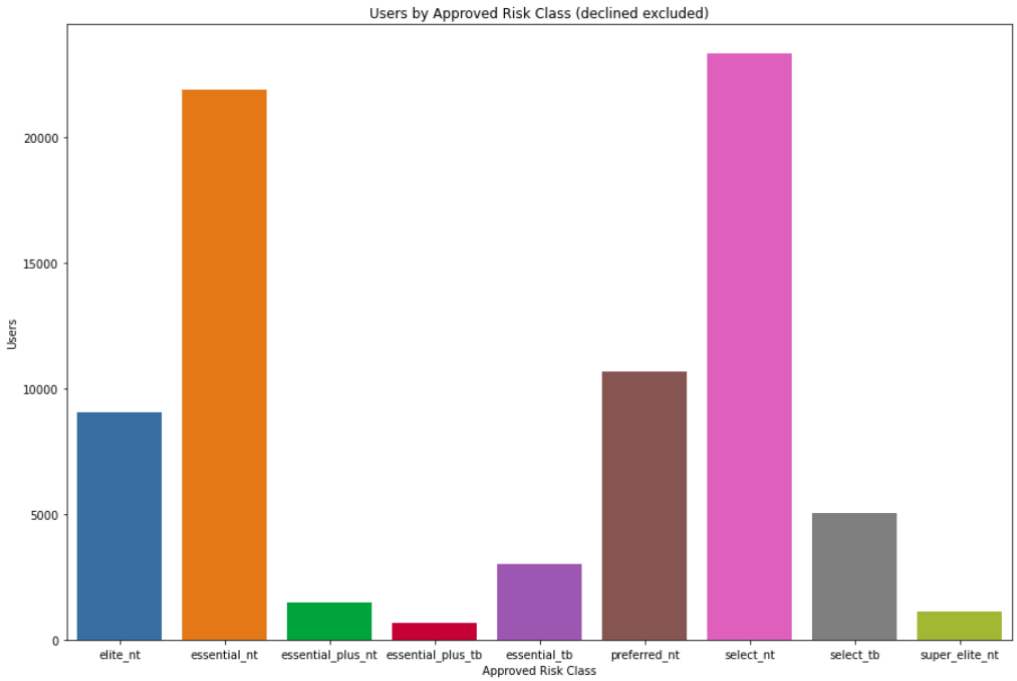
3. There were many fields with a large number of nulls that needed to be dealt with carefully due to the information that they provided. For example, a field that asks the user about whether they've had a stroke, another that asks about what kind of stroke, another that asks about when their last stroke was, etc. Knowing whether or not a user has had a stroke is a key piece of information for the model to correctly categorize the user, i.e., a null is potentially very informative. Thus, it took some time to look at each field with null values (50-60 fields) and decide the best way to replace the null values to retain the information that the model could use to predict risk class.

The first issues that I ran into was a mixed data types warning for about 40 fields when importing the data to Jupyter. This is an issue I ran into in other courses, and knew that it needed to be addressed, but required that I look up each field and create a dictionary specifying the data type for each.

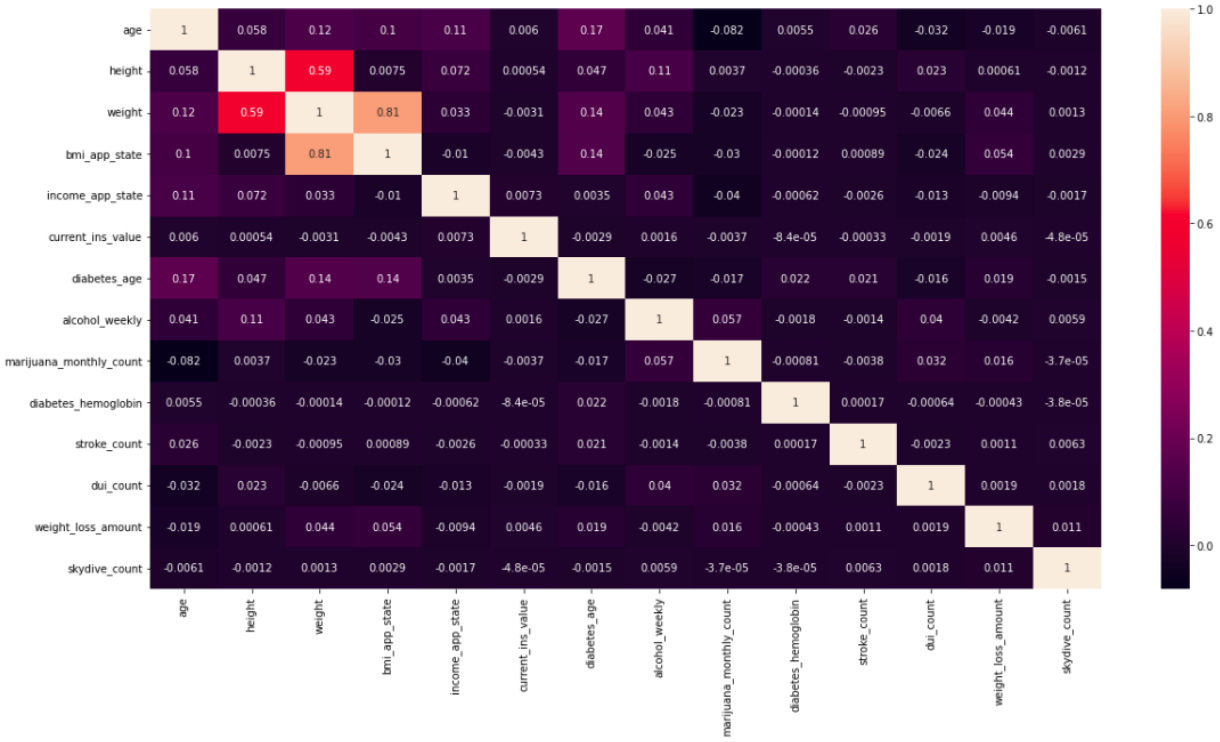
Once I had the data in Jupyter, I used pandas profiling for EDA, which is an incredible tool for generating the summaries needed to understand my data and identify issues. Data cleaning steps:

1. Identify and drop any fields with no non-null values
2. Created a risk class out of 'declined' users (which introduced imbalance in the classes as seen in the first barchart below) and dropped two other risk classes, 'super elite' and 'essential plus', due to small sample (more obvious in the second barchart below)





3. Drop tobacco risk classes. In talking with one of the actuaries at our company, it sounds like the rules for categorizing users who identify themselves as tobacco users are slightly different than non-tobacco users, and these risk classes are generally much smaller in terms of number of users (see above). Including this tobacco users would add noise to the data, and I think a separate model could be created for tobacco users if we have enough data.
4. I used Isolation Forest to remove outliers in numerical fields with no null values, which included age, height, weight, BMI, and income.
5. Because Isolation Forest can only work with fields without null values, I manually checked other numeric fields for outliers and made decisions about thresholds based on what seemed reasonable and how many observations exceeded a given threshold. For example, I removed users who reported having more than 25 DUI's.
6. I removed users who indicated that they were less than 18 years old or more than 60, as well as users whose BMI was less than 18.5 or more than 40. These users would be ineligible to even get a quote and so would be removed at that point.
7. Dealing with nulls: I manually searched through all fields with null values (using the results of the pandas profiling report) and decided how deal with each.
8. Correlation of numerical features:



After cleaning my data, I looked again at the correlations between the numerical features. Height, weight, and BMI are the most highly correlated, while the rest are near 0. This makes sense as BMI is calculated from height and weight, and generally taller people are heavier and shorter people weigh less. I considered removing height and weight, as these are somewhat redundant with BMI, but decided that 0.81 was still low enough that including all three might help the model more than removing them.

At the end of this process, I had 158,745 rows and 65 fields. While data cleaning was very time consuming, I think I ended up with a dataset that gives me the best chance of getting the best possible model for predicting a user's approved risk class.

Methods

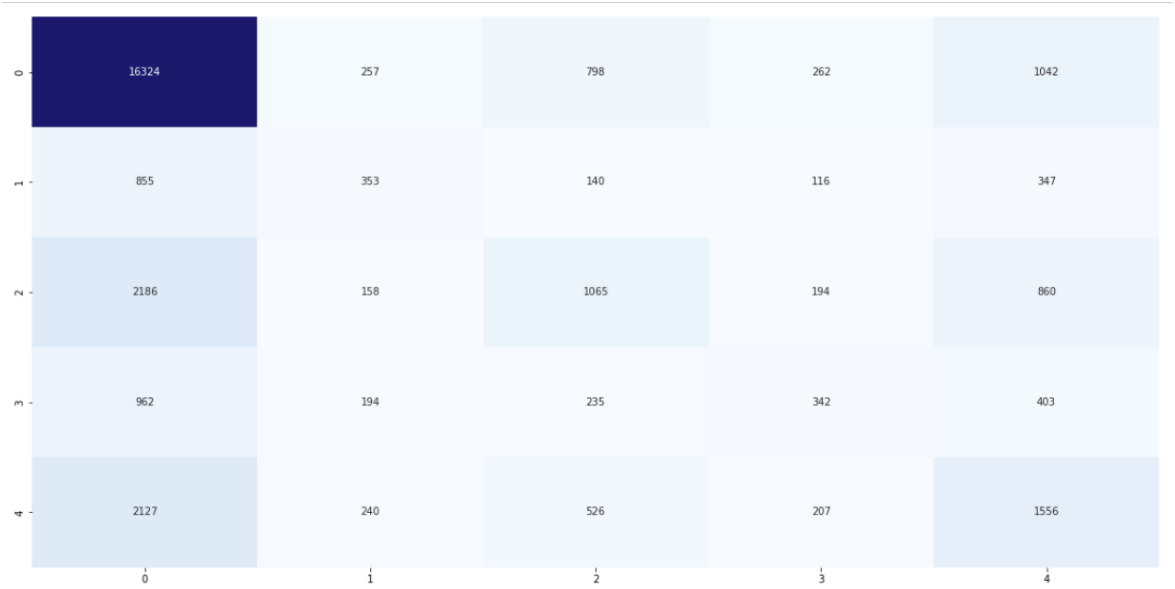
Given the size of the dataset, I used a Target Encoder to transform my categorical features. I had good luck with this method for my project in 630, seeing both a massive improvement in efficiency and a nice bump in accuracy without overfitting. The efficiency gained by not ending up with a dataframe with tens of thousands of columns unlocked the ability to run random or grid searches to tune parameters. I decided to start with a Random Forest multiclass classifier – I've found that Random Forest is generally a great out-of-the-box model to provide a baseline for subsequent models as they are generally efficient, flexible, and accurate without much tuning. After running my initial model, I did run a grid search to tune the parameters of the Random Forest model. Due to spending a couple of weeks on identifying and cleaning my data, this is as far as I've gotten as of this milestone, but I have plans to implement other methods before the final milestone (discussed in the future section).

Analysis

One issue that I have been considering how to deal with is that my dataset contains a large number of users who declined during the approval process, i.e., they were not approved for any of the life insurance products that we offer. But as these users were eligible for a quote, then this model will have to be able to make a prediction for them since they will be shown a price a quote. What I decided to do with these users is to create another risk class for them, declined. However, this creates an imbalanced dataset as there are more users declined than all the other risk classes combined (our approval rates have historically been in the 30-40% range, so this is expected).

I evaluated the accuracy of my models using overall accuracy, a confusion matrix, and a classification report. The accuracy of the initial model was 62%, which was very encouraging given the 29% baseline until I remembered that that baseline did not include declined users, which accounted for a large part of the 62% accuracy. As 59% of users are declined, this is a marginal improvement over simply predicting every user to be declined.

Below is the confusion matrix (0 = declined, 1 = elite, 2 = essential, 3 = preferred, 4 = select):



The imbalance of the dataset introduced by the inclusion of the declined users seems to be affecting the accuracy of the model as evidence by first column, where the model is cheating due to the prevalence of the declined users by classifying more users as declined than any other risk class.

This is further confirmed by the classification report, where we see a better recall than precision for the declined users, while recall is much worse for the smaller risk classes. This confirms what the confusion matrix shows – the model is improving the overall accuracy by classifying users in all classes as declined.

	Precision	Recall	F-1	Support
Declined	0.73	0.87	0.79	18683
Elite	0.29	0.19	0.23	1811
Essential	0.39	0.24	0.29	4463
Preferred	0.31	0.16	0.21	2136
Select	0.37	0.33	0.35	4656
Accuracy			0.62	31749
Macro avg	0.42	0.36	0.38	31749
Weighted avg	0.57	0.62	0.59	31749

I did conduct a grid search to tune the parameters and ran the model again. While the overall accuracy increased by 1%, the model achieved it by increasing its bias toward declined users. The main evidence for this conclusion is that the recall for the declined users increased to 91%, while three of the four other risk classes saw recall decrease by 5%.

Conclusion

My initial model indicates that further development of this model needs to focus on the imbalance in the dataset, as any attempts to increase the accuracy of the model are likely to simply exploit the fact that 59% of the users in the dataset are declined, and thus this will create a ceiling on accuracy for any model that doesn't deal with it.

There are a few options available for dealing with this issue (Nabi, 2018, Machine learning multiclass classification with imbalanced data set, Towards Data Science, retrieved from https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a):

1. Undersampling the declined users
2. Oversampling the other risk classes
3. Tuning the class weight parameter

Another option is to create two models, one that predicts approved vs. declined, i.e., binary classification, and another that predicts risk class for approved users. The 29% benchmark noted above is restricted to approved users, i.e., 29% of users end up in the same risk class after being approved as they were when they received a quote, which assumes the user was approved.

Another option, noted in the proposal, is to use the mortality rate associated with a given risk class as the target, and the convert the problem to a regression problem. We could then bucket the results and convert the predicted mortality rate back to a risk class.

I intend to work through these options in the above order, as time allows.

Assumptions

Given the current state of quoted pricing, I made some simplifying assumptions about the number of risk classes needed at the quote stage. The more classes we have, the less likely we will be able to create an accurate model. By eliminating very small risk classes, such as super elite and essential plus,

as well as tobacco risk classes, we are left with our larger risk classes, elite, preferred, select, and essential. Increasing the accuracy of predicting users in these risk classes would be a great improvement in aligning our pricing from quote to approved, and thus seems justified in light of the business problem we are trying to solve.

Limitations

A major limitation of this model is that it's a proof of concept using historical data from life insurance products underwritten by a carrier we are no longer working with, so now matter how this model is there is work to be done develop a similar model using data from our current carrier.

Challenges

- Data cleaning, i.e., messy user inputted data
- Imbalanced data
- Lack of data with current carrier

Future uses/additional applications

We could potentially re-direct users to other options if the model predicts that they will be declined (perhaps above a certain threshold of certainty). We currently off ramp users to another company with an expanded product selection for riskier users if they are declined. We could potentially "suggest" this option to users before incurring the underwriting costs to process their application.

Recommendations

None at this time. Continue with developing the model.

Implementation Plan

I think any kind of implementation plan is months away, even if the model can achieve the necessary accuracy, due the lack of data with our current carrier. Even if we could achieve good accuracy with data from our current carrier, I think the business would be apprehensive about implementing a model without more data. If we were to get to the point that we were ready to implement this model, I think the main challenge would be getting the model performant enough to make a prediction for a single user in a few seconds, and integrating with our current workflow. Generally, users receive a quote almost immediately when they request one on our site. This model would need to be integrated between user input and returning results. The assistance of our machine learning engineer would be needed for this task.

Ethical Assessment

As noted in the project proposal, the main issue that I'm concerned about is that the accuracy of the model is consistent for all users and does not discriminate against minority users. In talking with one of the data scientists at my company, this would take some work, as he is currently engaged in this sort of analysis. First, we need predict a user's minority status based on their name, which he has found a way to do this (not sure if this is a package or a data enhancement service). Once that is determined, he has used a package that calculates the Adverse Impact Ratio for each protected class.

For my project, I'd like to at least check the accuracy of the model for men vs. women, as I have a field for gender in my data, and the data scientist that I've discussed this with has found that some of the models that the data science team has built do discriminate against women.

One other ethical issue that I realized is related to this project since I submitted my proposal is that the issue we now have with our pricing where quoted price to approved premium increases substantially for most users could be construed as false advertising in the sense that users likely take into account their quoted priced when deciding whether or not to complete an application. So, the business problem we are trying to solve has ethical implications in the sense that a solution that would provide users with more accurate price at quote would address this issue of unintentional false advertising.

Ten Questions an audience could ask:

1. What key metric(s) and thresholds would need to be met to put the model into production?
2. Is the model accurate enough to be put into production? Why or why not?
3. If not, what improvements need to be made?
4. How could this model be leveraged, i.e., besides predicting a user's approved risk class for pricing, how else could this model be used?
5. This is a POC using data from products underwritten by Sammons Financial Group. How does this model perform using data from Bestow Life Insurance Company?
6. What is the anticipated financial impact of using this model? What cost savings or increased revenue can the company expect if we put this model into production?
7. How does the model increase accuracy by risk class, i.e., overall 29% of users are in the same risk class after being approved as they were when they were quoted, but how does that break down by risk class?
8. What should we do with users who the model predicts will be declined? We cannot use the model to decline them – they must go through the underwriting process. So, we will have to decide what risk class these users should be in when they get a quote.
9. Is there enough data to build a similar model for tobacco users? If not, what do we do with them?
10. Which questions on the application are most predictive of approved risk, and how many? The viability of this solution during the quote phase of the user experience depends on the answer to this question, as we can only move so many questions from the application to quote.

References

Nabi, J. (2018). Machine learning multiclass classification with imbalanced data set. Towards Data Science. Retrieved from https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a