Milan Sherman
DSC 680
Project One
Milestone 3

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 theSFG data from 2021 though March of 2023 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:

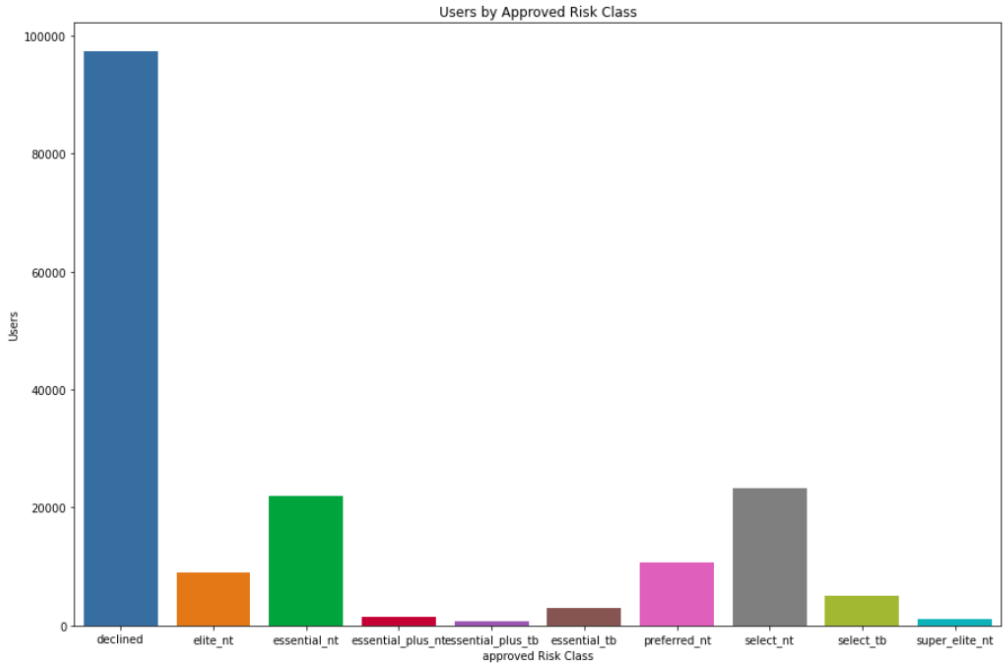
1. The data is input by the user, and they can input whatever they want. This could be an error, or just users who don't want to share accurate information for some reason. For example, I found users who claimed to have 5000 alcoholic drinks per week.

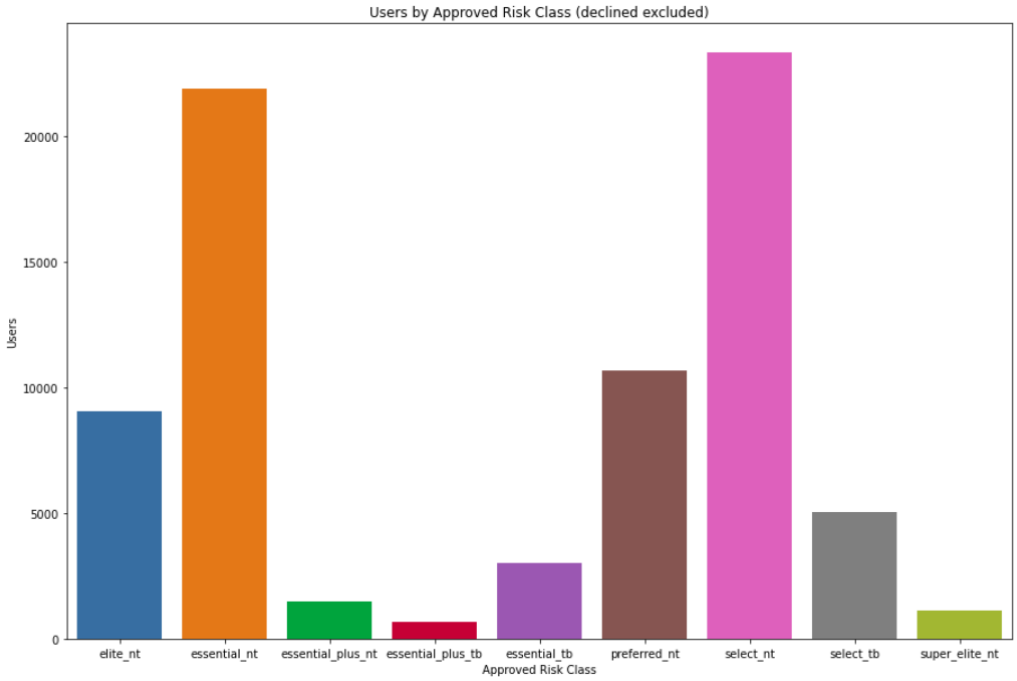
2. We have revised our application a few times, and have retained historical data for all users, so there are duplicate fields, or fields which have similar information (but not exactly the same). For example, I found six fields related to alcohol consumption, and this was not an isolated incident (I 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.

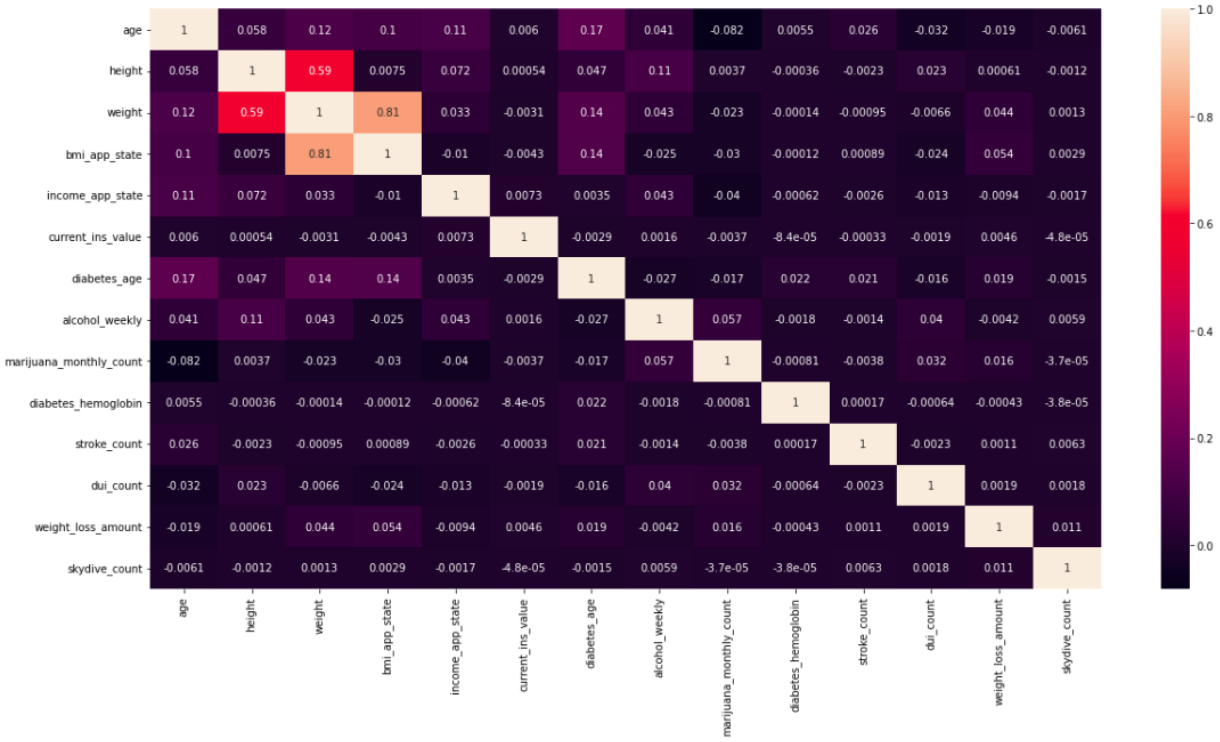
Once I had the data in Jupyter, I used pandas profiling for EDA, which is an incredible tool for generating the summaries needed to understand my data and identify issues. Data cleaning steps:

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 the initial Random Forest Model, I decided that I needed to separate declined users from the approved risk classes, since 59% of users were declined in this dataset, and that lack of balance was making it difficult to a model that was accurate for approved risk classes. I tried two approaches:

1. First create a binary classification model for approved vs. declined users, and then create a multiclass classification model for approved risk classes, which are much more balanced if the declined users are removed.
2. Assign a numerical value to each risk class using the mortality rate associated with each class, and convert the problem to a regression problem. After getting predictions, we can convert the numerical values back to risk classes.

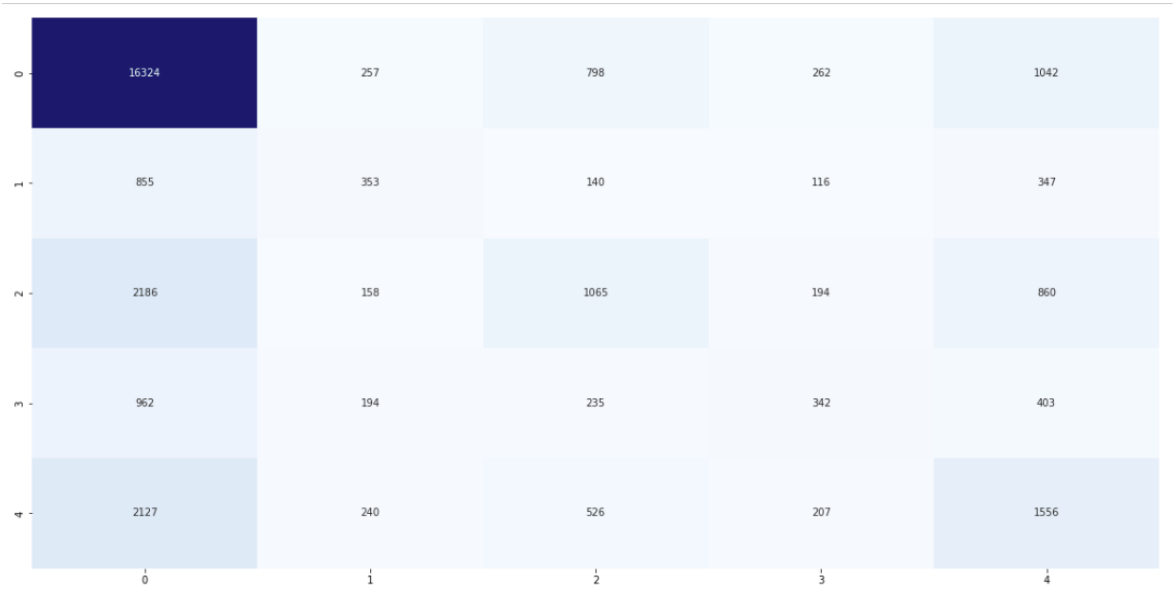
For the first method, I tried Random Forest, CatBoost, and XGBoost for both the binary and multiclass classification models. For the regression model, I tried Random Forest, CatBoost, and Ridge regression algorithms.

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%.

Updated Analysis

I tried a number of models, but ultimately found the CatBoost models to perform the best for this problem, with XGBoost a close second. I tried tuning the parameters for the Random Forest model a few times and the numbers below represent the best for that model for this problem.

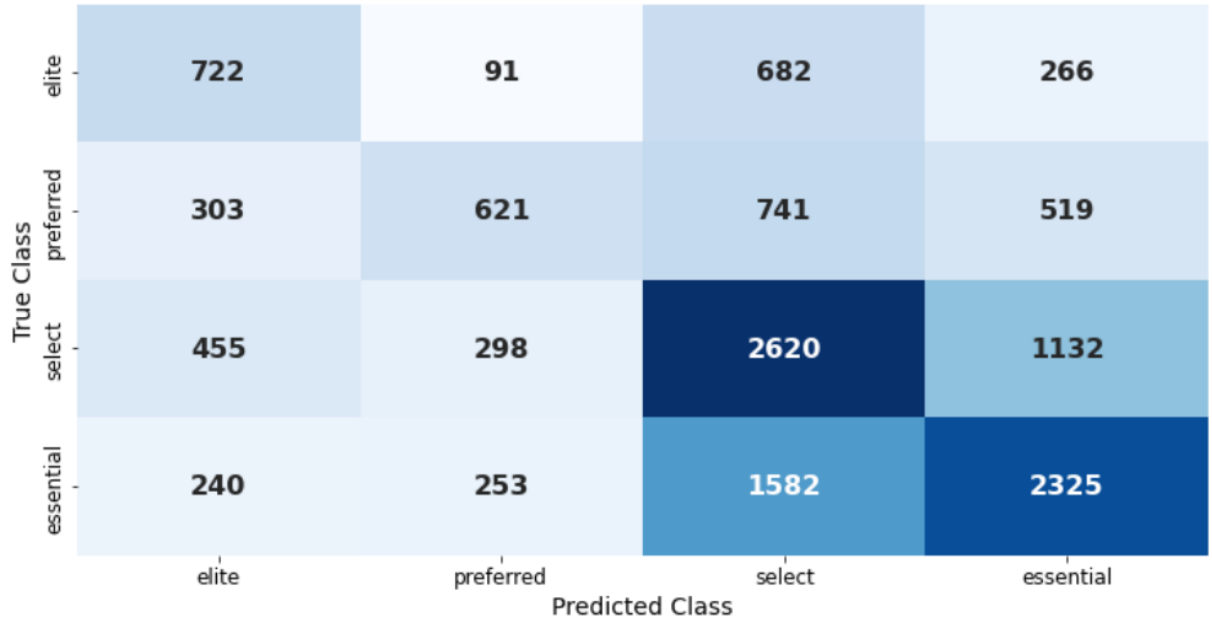
Classification Model	Binary Accuracy	Multiclass Accuracy
Random Forest	0.807	0.457
CatBoost	0.892	0.489
XGBoost	0.868	0.492

Comparison of Classification Models Accuracy (Approved/Declined)

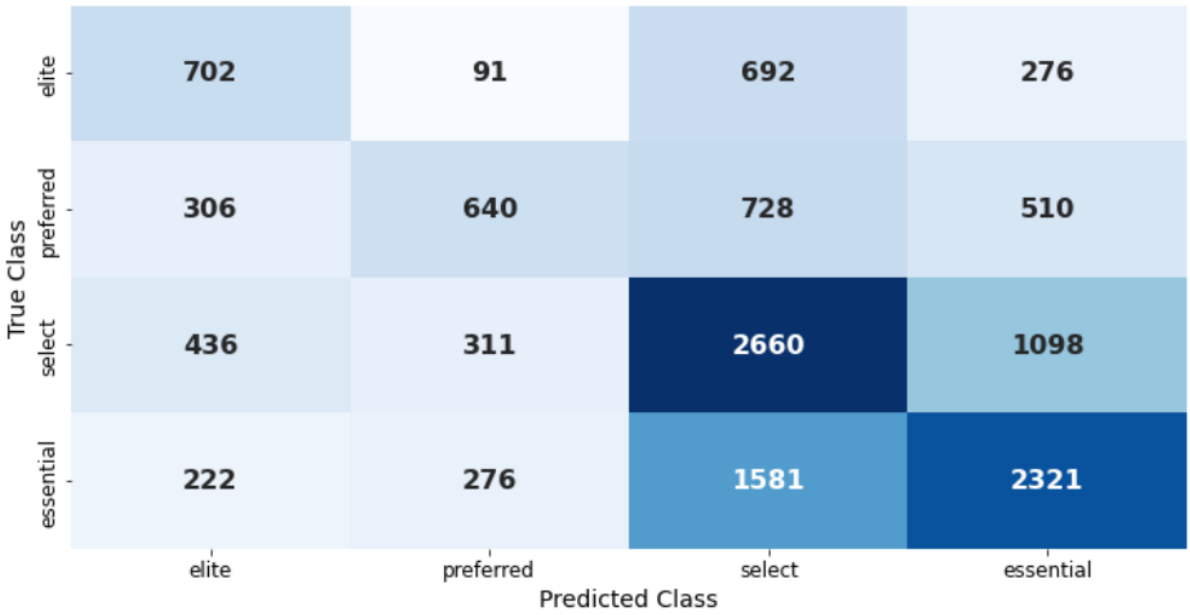
Classification Model	Class	Binary Precision	Binary Recall
Random Forest	Declined	0.90	0.75
	Approved	0.72	0.88
CatBoost	Declined	0.95	0.86
	Approved	0.83	0.94
XGBoost	Declined	0.91	0.85
	Approved	0.81	0.89

Comparison of Binary Classification Models Precision and Recall

The CatBoost Binary classifier was the clear winner, while the multiclass classifier was very close between CatBoost and XGBoost. In fact, XGBoot had slightly better accuracy, but ultimately the CatBoost model performed better in terms of the distribution of errors, as shown in comparing the confusion matrix for each model below. The risk classes are listed in order from best to worst, or from least to most risky. The riskier risk classes are higher priced, and the issue that we are trying to solve for is having most users put into the elite risk class when receiving a quote, and getting approved in a riskier risk class and seeing a higher price after being approved than they saw when they received a quote. We recently A/B tested a 24% across the board increase of the quoted price shown to users, which was successful and which we currently show to all users now (this is the solution we are trying to improve upon). What this experiment showed is that users are not very price sensitive at quote, as we saw only a slight decrease in users starting an application after receiving a higher quote than those who did not see the larger quote price. That being the case, my goal was to minimize the number of users below the diagonal in the confusion matrix, as these represent users who were predicted to be in a better risk class than they ended up in at approved, and thus were shown a better price at quote than approved. It seems that our users can better tolerate seeing a higher price at quote than approved, so I was looking for the model that had more users above the diagonal than below. Given that the diagonal was very similar between the two models as evidenced by the similarity of the accuracies of the two models (48.9% for CatBoost vs. 49.2% for XGBoost), the CatBoost model tended to have more users above the diagonal than below relative to the XGBoost model. However, the two models were so close that it's really a toss-up – the CatBoost model only had 29 fewer users below the diagonal, while the XGBoost model had 35 more users on the diagonal, i.e. users who were correctly predicted. These differences are small enough to be due to chance, i.e, a different train/test split could easily swing these numbers in opposite directions. Ultimately, more work needs to be done to tune these models before making a decision.



CatBoost MultiClass Confusion Matrix



XGBoost Multiclass Confusion Matrix

Conclusion (Updated)

Use the CatBoost Binary classifier to predict approved or declined:

- Users predicted to be declined will be shown the price for essential_nt, which are the highest prices. This is advantageous to us, as users who are declined incur underwriting costs without resulting in any revenue. Showing them the highest prices at quote may influence them to not start an application and therefore save us those costs. Furthermore, if they are predicted to be declined but are not, they are likely to be in one of our riskier risk classes, so this is also likely to be the price closest to what they will see if approved.
- Users predicted to be approved will be input into the multiclass model to determine the approved risk class. This means that some users who may end up being declined will be fed to the model and could potentially degrade the performance of that model somewhat. In the CatBoost binary classifier, about 14% of declined users were predicted to be approved and would be labeled as being in the lowest risk class (essential) for model training purposes.

Use either the CatBoost or XGBoost multiclass classifier to predict approved risk class. Below is the distribution of Quoted and Approved Risk Class, i.e., the number of users who were quoted in each class and which risk class they ended up in after being approved. Note when comparing with the confusion matrices above, there are many more users in the visualization below since the confusion matrices only depict the 20% of users in the test group.

	Quoted Risk Class				
Approved Risk Class	elite_nt	preferred_nt	select_nt	essential_nt	Grand Total
elite_nt	8,996	34	7		9,037
preferred_nt	7,621	2,703	26	5	10,355
select_nt	13,302	2,290	6,202	18	21,812
essential_nt	11,520	2,318	3,117	2,483	19,438
Grand Total	41,439	7,345	9,352	2,506	60,642

Distribution of Quoted and Approved Risk Classes

We can think of Quoted Risk Class as our current "prediction" of what risk class a user will be in if approved, and thus I visualized this distribution in a way that is comparable to the confusion matrices shown above. The first thing to notice is that almost all the users who were not correctly predicted, i.e., quoted risk class = approved risk class, were below the diagonal, and thus were shown a price at quote that was lower than what they saw after being approved. In fact, 66% of users ended up in a riskier risk class after being approved than they were in when receiving a quote. Another insight is that approximately 2/3 of our users are put into the best risk class at quote, i.e., elite_nt, which is a big part of the problem. Finally, 33.6% of users are in the same risk class at quote and approved, while only 0.1% of users are put in a riskier class at quote than approved (and thus would see a better price at approved). In general, this distribution encourages users to start an application and sets unrealistic expectations about what price they can expect to pay for a policy.

Comparing these numbers to the CatBoost model, 24% of users will see a lower price at quote, 27% will get a higher quote, and 49% will see the same price. Given what we know about the price sensitivity of users at quote vs. approved, this is a much better prediction than what we currently have.

Next steps

1. Run the model(s) (binary and multiclass) again with the Top 10-20 features to test performance.
2. Tune the CatBoost and XGBoost models to decide which to use
3. Repeat this process using BLIC data to see if we have enough data to start using a model to predict approved risk class
4. Create a model for tobacco users
5. Perform A/B testing to determine:
    a. How adding questions to the quote experience influences user dropoff
    b. How implementing this model for predicting approved risk class affects funnel conversion rates
    c. The revenue/cost impact of implementing this model

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 no 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.

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 tend discriminate against women in terms of approval.

I found that the accuracy of the catboost model by gender was 49.1% for males and 48.7% for females, which seems negligible.

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.

This model accurately predicts about half of users, while being close to distributing the other 50% above and below the price that they'll see if approved. Thus, the issue potential issue of false advertising and/or unintentionally inducing a user to complete an application is improved with this model vs. the current state.

Ten Questions an audience could ask (with answers):

1. What key metric(s) and thresholds would need to be met to put the model into production?
    a. The primary metric is accuracy – as much as possible we would like to correctly predict the risk class would be in if they are approved.
    b. An important secondary metric is the proportion of users who are approved in a riskier risk class than they were quoted, i.e., we would like to minimize the number of users who are predicted to be in a better risk class than what they were actually approved for.
    c. Another secondary metric is the number of users who are predicted to be in a risk class that is not "adjacent" to their true risk class. The larger the difference between a user's true and predicted risk class, the larger the price discrepancy they will see.
2. Is the model accurate enough to be put into production? Why or why not?
    a. Using historical data as a baseline, the model is much more accurate than the current system of assigning risk class at quote which is entirely based on BMI. Both the improvements in terms of accuracy, as well as the improvement of the distribution of errors make this model a significant improvement over our current state. I would say that this model is accurate enough to warrant development, but not accurate enough to put into production.
3. If not, what improvements need to be made?
    a. The model described here is considered to be a first version of a model that could be put into production. Other work and/or improvements that are needed before a production-ready model would be include:
        i. Run the model(s) (binary and multiclass) again with the Top 10-20 features to test performance.
        ii. Tune the CatBoost and XGBoost models to decide which to use
        iii. Repeat this process using BLIC data to see if we have enough data to start using a model to predict approved risk class
        iv. Perform A/B testing to determine:

1. How adding questions to the quote experience influences user dropoff

v. How implementing this model for predicting approved risk class affects funnel conversion rates

vi. The revenue/cost impact of implementing this model

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?
    a. In order to save spending on underwriting, we are currently offramping users to a third party who is likely to have a life insurance product that the user is eligible for if they we expect the user to be declined. However, we are currently making that determination based on income alone, as we found income to be highly correlated with eligibility. This model is likely more accurate than using income alone, and would be an improvement over our current state in that is less likely to be discriminatory.
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?
    a. Answering this question is one of the next steps.
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?
    a. Answering this question is one of the next steps.
7. How does the model increase accuracy by risk class, i.e., overall 33% 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?

Risk Class	Current State	Model	Proportion of Users
Elite	99.5%	41.0%	14.9%
Preferred	26.1%	28.4%	17.1%
Select	28.4%	58.2%	36.0%
Essential	12.8%	52.8%	32.1%

The current state has fantastic accuracy for the Elite risk class, but at the cost of putting of 2/3 of our users into that risk class and misclassifying nearly 80% of them. The true distribution shows how our approved risk classes are distributed in terms of proportion of users, and that the model achieves the improvement over the current state by being more accurate with our larger risk classes.

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.
    a. By putting these users into the lowest (or riskiest) risk class, we show them the highest prices at quote and therefore potentially save underwriting costs by deterring users who are likely to be declined to not complete an application.

9. Is there enough data to build a similar model for tobacco users? If not, what do we do with them?

    a. This is a problem I have not had time to tackle yet. In fact, I don't even currently know if the tobacco risk classes are more generally more accurate than the risk classes analyzed and predicted in this project. The first step would be to do that analysis – given the much smaller number of users in the tobacco risk classes, if the accuracy of these risk classes under the current system is better, it may be sufficient to continue to use the current system 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.

Feature Importance

The top 20 features (in order of importance) for the CatBoost model are shown below:

1. **BMI**
2. **Age**
3. Occupation Description
4. Income
5. **State**
6. **Weight**
7. Marijuana monthly count
8. Medical conditions
9. **Height**
10. Marijuana use (yes/no)
11. Weekly alcohol consumption
12. Current insurance value
13. Mental health diagnosis
14. Employment status
15. **Gender**
16. Valid driver's license
17. Travel (what countries the user plans to travel to in the near future)
18. Weight loss amount (for users who report having recently lost weight)
19. Citizen (U.S. citizen yes/no)
20. Criminal history

Although it's not realistic to move all these questions to the quote section of the user experience, it also may not be realistic to move all the Top 10 depending on the sensitivity of the questions. The more questions we ask, and the more sensitive the questions are, the more likely we will see users not generate a quote or continue through the process. The good news is that six of the Top 20 questions are already asked as part of current quote experience (height, weight, state, age, and gender, and BMI is generated from the first two). Thus, it seems that adding 4-5 questions to the quote experience is not unrealistic. In fact, after tuning the model, it may be worth creating a baseline based on the data collected during the current quote experience and add one feature at a time to the model until we see performance level off.

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