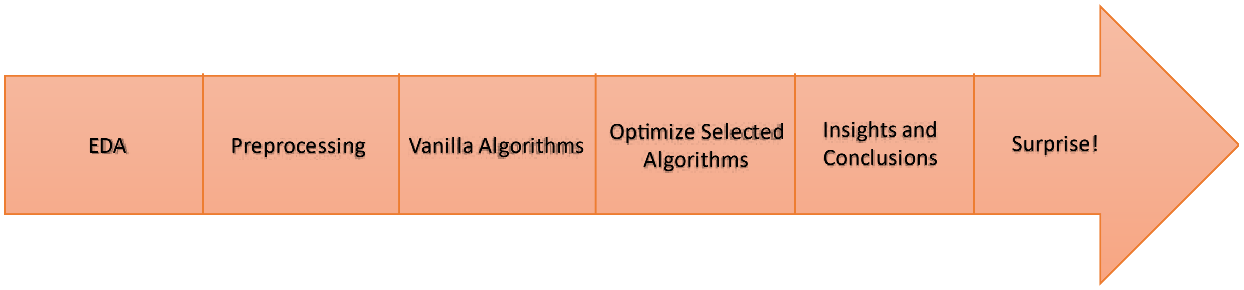
Marc Violides

To churn or not to churn, that is the question

I am sure we've all been there before, whether for a school or a work project, christmas or summer break:

The project in question is a churn prediction problem: we have to predict for DLM bank which customers will churn and which will stay put in their company.
What is the purpose? After running the prediction algorithm, the idea for the company is to contact customers that are expected to churn and convince them to stay, whether directly or indirectly.
Why is this useful? Customers will feel more wanted and more valued when the company checks in on them. This is brilliant because what usually happens is that the company in question convinces their customers to stay after that the latter has opted to leave, and is not very efficient.
Is it limited to banking? No, also in many other sectors: if you have ever gone to cancel a gym membership, you surely know how clingy the gym becomes, almost offering you their service for free.
What is the impact of our algorithm? It is obviously superior to contacting every single customer, which is not possible for a large bank. Additionally, contacting people who are not planning to churn but resulted as churning in our algorithm (False Positives) will not do any harm by making them fell more valued; however, the issue will be with customers which will churn but are not detected as such (False Negatives) because this will cause the company to lose some of its customers.
What is our goal? That being said, we will look for the best possible algorithm according to the metrics we chose, limiting False Negatives to an extent.
Where to start? First, we will explore the data, performing what is known as EDA (exploratory data analysis). This is the longest part of our report, but we personally believe that this is a crucial part of any machine learning problem, as it helps us to center the problem at stake and we believe that without this step, we would just be throwing ourselves into the lion's den.

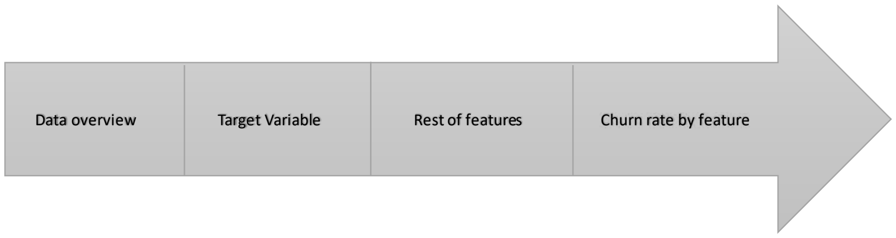
In summary, here are the main phases we undertook to design our project:



Sounds easy right? Well, yes and no. Let's see how we proceeded for each stage:

Phase 1: EDA: FOR THIS PHASE RUN "EDA.py" file

We are going to conduct an analysis of each of the following sections shown in the diagram below



Step 0: Data overview

From the overview function we defined, we see that we have 17 features, from which 11 are numerical and 6 are objects (we excluded CID). You were kind enough to split the features into categories: we have basic info; target feature that we want to predict; Demographic features and variables which both constitute the independent variables. The dataset is clean, having 0 missing values, so we can proceed to the next step.

Step 1: Target Variable (FOR THIS SECTION CHECK THE MICROSOFT EDGE FILE, OR WHATEVER BROWSER THAT APPEARS AFTER RUNNING "EDA.py")

Starting with the target variable (Attrited_Flag), we notice that we are dealing with a binary classification exercise with unbalanced target variable

- Attrited = 16.1%
- Existing = 83.9%

As a result, we should choose a metric that deals with imbalanced datasets, like F1 Score rather than accuracy, precision, recall or any of the other metrics we saw in class. The aforementioned metric is used to check the quality of the model predictions, being a harmonic mean of precision and recall.

Step 2: Rest of Features (CHECK SCIVIEW WINDOW OF PYTHON)

- ***Demographic features:***
  - ➢ Gender: we notice that there is a slightly higher proportion of females compared to males, but we do not believe that this will affect churn rate
  - ➢ Income: most people in the dataset earn less than $40k so not wealthy.

- Education, Graduates far outnumber other categories. We don't expect them to be rich since they spent all their money on college tuitions, which explains the preceding finding.
- Age: mid-aged people, especially around 49 and 50 years old. This perhaps hints that we are dealing with people who are supposed to be financially stable.
- Dependents: most people have 2 to 3 dependents, which is pretty standard.

- ***Product features:***
    - Card Category: majority of people have blue Cards and very few have silver, gold or Platinum. This leads us to believe that upgrading from blue is not for everyone in this company.
    - Marital Status: considerably more married couples in our dataset compared to other categories (maybe this is a message to get married?). We would expect
people to be financially stable here as well and therefore lean towards not churning.
    - Months on book: people in this dataset have been pretty loyal, or at least have stuck with this company for a long time, with a high number of people having spent 35 months.
    - Total relationship count: the number of individuals stabilizes at more than 3 relationships.
    - Number of months inactive: The peak is at 3 months. This surprised us because a lot can happen in 3 months without being active.
    - Number of contacts in the last 12 months: most people have had contacts 2 or 3 times. We must say, contacting the bank is not a good sign for the bank in general.
    - Total transaction amount: it comes as no surprise that the peak is towards the beginning (at around $4500) because we saw that the majority of people have an income of less than $40k (and who knows if it is with or without taxes).
    - Total number of transactions, the dataset is pretty balanced, with most people located towards the middle (from 65 to 85 transactions). This is a good sign for the company, as there are many transactions that are taking place, without overloading the system at the same time.
    - Credit limit: For the reason mentioned in the previous paragraph, it is low on average.
    - Average utilization ratio: a bit low in our opinion compared to the number of transactions made.

Now that we've had an idea about our data and we know where to redirect our thoughts, we can start to look at the churn rate feature by feature.

Step 3: Churn rate by feature (BACK TO MICROSOFT EDGE AND ALSO ATTACHED EXCEL FILE FOR VALUES, OR YOU CAN JUST TRUST US 😊)

- ***Demographic Features: Churn rate by:***
    - Gender: Not too much influence on churn rate as we presumed before.
    - Education Level: Highest churn rate is among individuals with doctorates (27%) which is logical, because of college debts. DLM could offer special offers for these people.
    - Marital Status: The people who are churning the most (21%) are those with unknown marital status. This is an issue for the company because it doesn't have a clear idea on how to improve the experience of these customers. To explain ourselves better, the bank could suggest discounts for married men related to group/family discounts, but since the marital status is unknown, it can do nothing about it. However, we believe that the percentage is too small to form a real issue.
    - Income Category: The two classes with the highest churn rates (21%) are the two extreme opposites: those with salary less than 40k and those with salary more than 120k. In this case, maybe it is best for the company to proceed as it is in order not to increase the societal class imbalances.
    - Customer Age: Customers with the highest churn rates are middle aged men (40 to 59) which is a bit surprising since you expect these people to be financially stable at this point, as we previously hinted. This is put in evidence even more with the boxplots of the diagram, with the first quartiles for both attrited and existing customers being respectively 36 and 36.75, and the third quartiles being 57 and 59.25, so most values are contained within the aforementioned range (40 to 59). In

this regard, the company should provide better services for this category of people (at that age they might be thinking about retirement, so why not provide them with a retirement plan, among other)

- Dependents: Stable ratio and the boxplots for existing and churned customers are quite similar, so for this matter we believe that there is not much to say and that the company should proceed as is.

Conclusion for Demographic features:

The company should look to improve the satisfaction of people with doctorates, middle-aged men but should also look to lower the number of unknown marital statuses.

- ***Product features: Churn rate by:***
  - Card Category: Highest churn rate with platinum card (33%, with the next one being 22%). It is surprising, because as we noticed, very few people reach platinum rank, so you would like to think that most people who arrive at this stage are very reluctant to back out. However, this doesn't seem to be the case at all. The company should definitely have a look in improving drastically the benefits of being a platinum card holder.
  - Months on book: Same reasoning as dependent count: the period of relationship with the bank doesn't seem to influence customer churn in a visible manner. To improve this, the bank could issue loyalty plans to make customers more loyal.
  - Total Relationship Count: The more products a customer holds, the lesser the churn rate. This means that the company should plan on getting those who have 0 to 1 product to buy more in order to almost guarantee them staying at their company.
  - Number of months inactive: People with 4 to 5 months of inactivity are more likely to churn than the rest. It comes as no surprise that those who churn the less are those who are inactive only from 0 to 1 month. The bank should constantly send emails and updates to their customers to solve this issue and push them to be active.
  - Number of contacts in last 12 months: The more contacts customers have with DLM, the more likely the customer is to churn. The bank needs to limit at a maximum the number of contacts by satisfying the needs of customers: a great start in this regard is the prediction problem we are undertaking where the bank can contact customers before them churning.
  - Total transaction amount: The higher the amount, the less likely the customer is to churn. The attrited boxplot has its first and third quartiles, as well as its median at the beginning, which suggest that most people that are churning are those with low transaction amounts. The bank can do some discounts on every transaction made (the more transactions, the less you pay). The same logic can be applied for credit limit which is left skewed (with a sudden peak in the 34.5k – 35k region). This logic can also be applied to the total transaction count which is closely linked to the transaction amount.
  - Average Utilization Ratio: From the attrited customer boxplot we notice that people who use their card the least are most likely to churn. The bank should provide additional (if any) benefits for using the card to encourage their customer to use it more.

Conclusion for Product features:

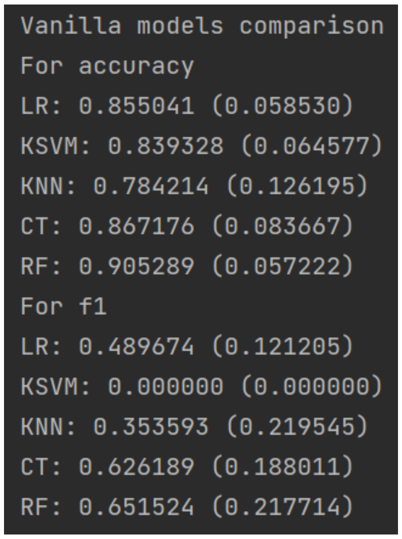
DLM should improve the experience of people with platinum cards, work on customer loyalty and encourage people to buy more products, which helps them stay more active. It must also lower number of contacts and do more discounts to increase transactions and credit limits. Lastly, they must improve the cards' benefits.

Phase 2: Data Preprocessing: YOU CAN NOW RUN THE “proj.py” FILE

1. The first step is to remove features that we will not use in our prediction and target, which are CID and CLIENTNUM.
2. As we know, features that we use to predict must all be numerical ones, so we start by transforming categorical features into numerical features. To do that, we split them into 2 categories:
   a. Those with 2 unique values: We encode them using LabelEncoder
   b. Those with more than 2 unique values: We encode them using OneHotEncoding or pd.get_dummies, which is basically the same thing.
3. We then chose to manually encode the target feature, as it was inverting the roles (giving existing customer as 1 and attrited customer as 0. This would cause a huge problem for our metrics and for our algorithm)
4. Then, we split the data into y and X, the target variable and the other features, respectively
5. Split data from previous phase into train and test sets (we chose 80%-20%)
6. Conduct feature scaling to avoid inefficiency in algorithm.

Phase 3: Vanilla Algorithms (excluding linear SVM because ineffective)

Now that we are done with preprocessing, we can start by testing some baseline algorithms, which we chose to be VANILLA algorithms. For the reasons previously mentioned, we chose Accuracy and F1 as the two metrics to evaluate our algorithms.

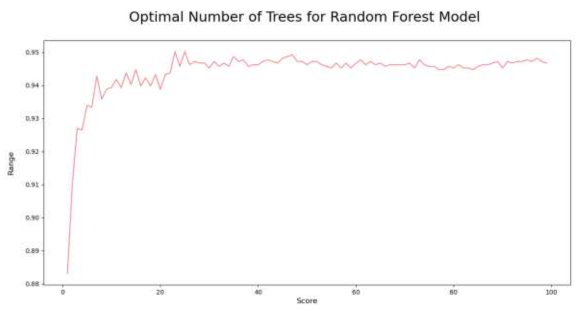


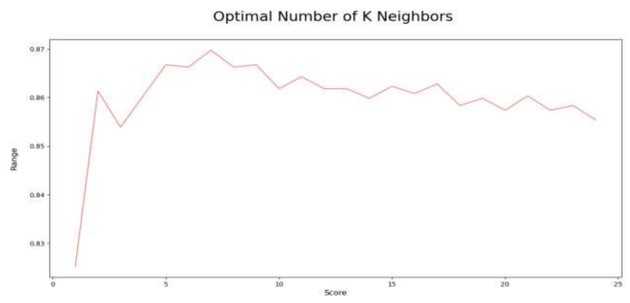
```
Vanilla models comparison
For accuracy
LR: 0.855041 (0.058530)
KSVM: 0.839328 (0.064577)
KNN: 0.784214 (0.126195)
CT: 0.867176 (0.083667)
RF: 0.905289 (0.057222)
For f1
LR: 0.489674 (0.121205)
KSVM: 0.000000 (0.000000)
KNN: 0.353593 (0.219545)
CT: 0.626189 (0.188011)
RF: 0.651524 (0.217714)
```

We found out that Random Forest and classification trees are the top 2 candidates but we learned in class that Random Forests are more accurate (if used with right parameters). It would therefore be a waste of time to study both, so we preferred to spend our time optimizing random forest, and also giving a chance to KNN, which, although it is 4th in our ranking, we believed had a higher potential to be improved compared to Logistic Regression (which is 3rd) because of the parameters we could tweak (mainly number of neighbors), while for logistic regression, the only way we could optimize it is by checking for multicollinearity between independent variables, which we believed were too few to cause any real issue.

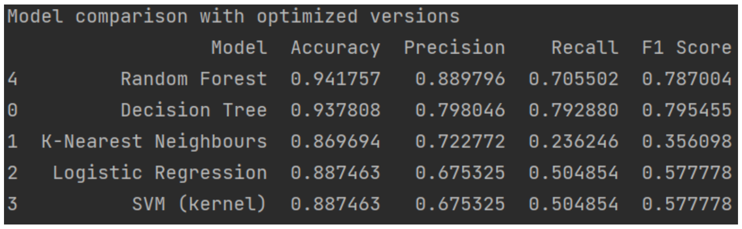
Phase 4: Optimizing Selected Algorithms: Random Forest ~~and KNN~~

- Our first idea to optimize was the way we saw in class, by drawing diagrams for optimal number of neighbors (KNN). You might ask: why don’t we also draw the diagram for the optimal number of trees (RF). That’s a good question, but RF having the accuracy and F1 that it has, will be taken for further optimization later; while we want to see whether the metrics will increase significantly with optimized KNN. We then compared optimized versions of these 2 algorithms and other algorithms, fitting them to our training set and evaluating the results.





We observe that optimal number is 27 and 7, respectively. However, for random forest this will not necessarily be the optimal one because we are putting an upper bound of 100, we will improve later.



```
Model comparison with optimized versions
                  Model  Accuracy  Precision    Recall  F1 Score
4         Random Forest  0.941757   0.889796  0.705502  0.787004
0         Decision Tree  0.937808   0.798046  0.792880  0.795455
1  K-Nearest Neighbours  0.869694   0.722772  0.236246  0.356098
2   Logistic Regression  0.887463   0.675325  0.504854  0.577778
3          SVM (kernel)  0.887463   0.675325  0.504854  0.577778
```

Even though Recall for KNN is the highest among the bunch, the improvement compared to the improved random forest is just negligeable when we look at the F1 difference and accuracy.

- So, KNN is eliminated and we are left with RF. We thought that we could reach a better result by hyperparameter tuning, but there was no chance that we were going to try every possible combination and get lucky, because knowing our luck, we would never reach the optimal results. So, we thought to ourselves: there must be some already created techniques to find optimal combinations, and lucky for us, there were.

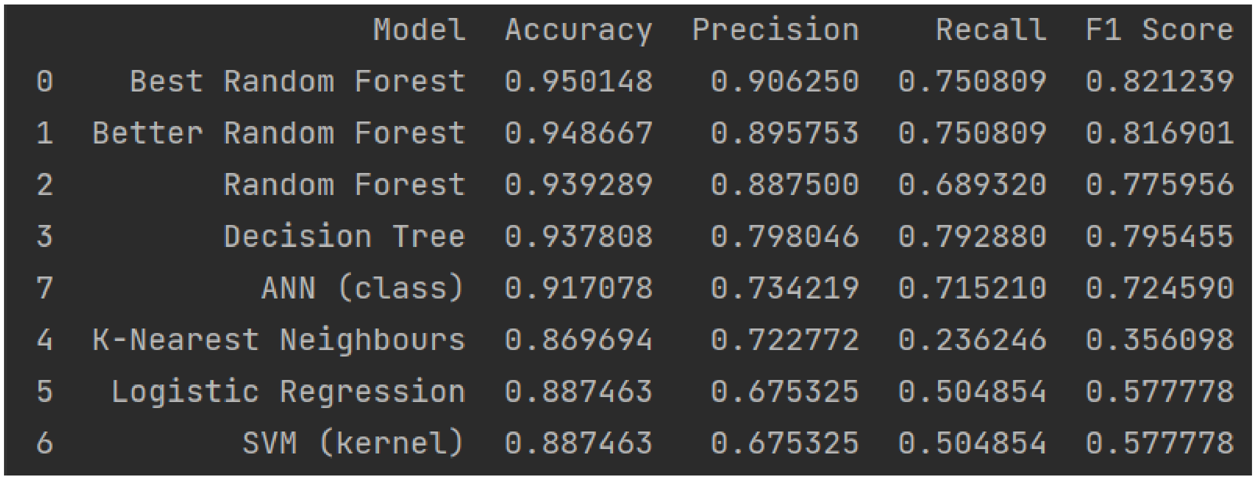
Trying random parameter combinations by hand
Using random search cv and grid search cv

- This is brilliant: but now we have another problem: WHICH TO USE?

- Why not do both? We decided to use Randomized Search to narrow down the possible hyperparameter values and then use Grid Search with the narrowed down parameters (maximum of 3 values per hyperparameter). Why's that? Because Grid Search is extremely accurate compared to Randomized Search but also super time consuming: by time consuming we are talking hours and maybe days to search between many hyperparameters. Even with the current restricted parameters we have, it is taking around 8 minutes for our project to fully run, but we hope that the end product is worth. This is because in Grid Search, we test out all possible combinations; while in Randomized Search CV, we take combinations at random and output the ones that gave the optimal accuracy/score. Plus, time, as long as it is not too exaggerated is something that the DLM bank could use as a tradeoff for better accuracy: better take more time but be more accurate to avoid losing customers. But also, it must not be too time consuming (hours or days), because we are losing on potential customers that we could retain.

Phase 5: Insights and Conclusions

We compared improved random forest models to be deceived and find out that the fully optimized versions only increased by a little bit. But then again, the algorithm per se performs well and anything that improves it makes it even better and a small change in accuracy at this stage can lead to a massive profit for the company in question.



```
                  Model  Accuracy  Precision    Recall  F1 Score
0    Best Random Forest  0.950148   0.906250  0.750809  0.821239
1  Better Random Forest  0.948667   0.895753  0.750809  0.816901
2         Random Forest  0.939289   0.887500  0.689320  0.775956
3         Decision Tree  0.937808   0.798046  0.792880  0.795455
7           ANN (class)  0.917078   0.734219  0.715210  0.724590
4  K-Nearest Neighbours  0.869694   0.722772  0.236246  0.356098
5   Logistic Regression  0.887463   0.675325  0.504854  0.577778
6          SVM (kernel)  0.887463   0.675325  0.504854  0.577778
```

Here is our final masterpiece: we decided to be inclusive and let everybody in even though some of them already lost. I know we already did many conclusions along the way but here is the final summarized one regarding our algorithms:

- Here, "Best random Forest", which is supposed to be the version we got with Grid Search after doing Randomized Search slightly outperforms "Better Random Forest" which is just the model with Randomized Search. However, this is Not always the case. It may happen that when you run our file, "Better RF" performs better than "Best RF". This doesn't mean that it performs worse, it is just that we have to run Randomized Search multiple times (around 100) and see which are the hyperparameters which repeat the most and then input them into Grid Search to make the latter perform at worst like the "Better RF". Additionally, if we increase the number of cv in our Grid Search, we would get a more accurate result with the already present hyperparameters. This is too time consuming however, and we thought that we would leave it as is to illustrate the tradeoffs that we must do (time vs performance).
- We also added the ANN model we did in class and saw that our model is far superior

- How to improve our model? We believe that the model we chose is the best possible from all models in class, but it can be improved by increasing the number of iterations and cv folds in Randomized Search CV and Grid Search.
- We believe that the company will love our model and will event think about hiring us full time.

Phase 6: Surprise! (NOW RUN "website.py")

We built a site/application where we pass in our file (churn.csv) and we get as an output the predictions that our optimal model (the best random forest model) gives us.

Making your life simple: To make our life easier, I took the best optimal parameters after running Grid Search multiple times, and created a model with these parameters, which will be used in our site to predict the target variable. Simply copy the link that appears in terminal

(It has this form: "streamlit run C:/Users/marcv/pythonProject1/websitepy" (this is for me, different for each person)) and paste it in a terminal: a Microsoft edge file will appear where you can browse the desired file to see the results of our prediction. Enjoy!

WARNING: notice that prediction in app starts from 0, in python CID starts from 1, so there is no decay error
If you are too lazy to run the app (I don't blame you), we provided 2 videos in the PowerPoint file

Bibliography
- ***https://learnpython.com/blog/python-customer-churn-prediction/***
- ***https://neptune.ai/blog/hyperparameter-tuning-in-python-a-complete-guide-2020***
- ***https://www.kdnuggets.com/2020/04/hyperparameter-tuning-python.html***
- ***https://www.geeksforgeeks.org/python-customer-churn-analysis-prediction/***
- ***https://www.kaggle.com/jsaguiar/churn-prediction-tutorial-with-sklearn***
- ***https://towardsdatascience.com/predict-customer-churn-in-python***
- ***https://towardsdatascience.com/hyperparameter-tuning-in-python-21a76794a1f7***
- ***https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/***
- ***https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm***
- ***https://towardsdatascience.com/exploratory-data-analysis-eda-python-87178e35b14***