Final Submission Report

Loan Default Prediction

Mikael Friederich

December 15th, 2023

Executive summary

We have proposed a decision tree model to the bank to predict the likelihood of their customers defaulting on their loans. This will help them switch to a more automated process free of human biases, that reaches a 74% *recall rate* (see Appendix 1 for definitions), which means the bank will be able to detect if a customer will default successfully almost 7.5 times out of 10. Our model is subject to limitations such as data quality issues, dependency to variables with many missing values, and was chosen purposely versus another better performing model for its interpretability and visualization strengths, which is key to the bank's federal regulation compliancy requirements. If implemented our model will create an increment to the bank's net revenue[1] of ~$50M under assumptions, an almost 300% increase, while also reducing the percentage amount of missed revenue over net revenue of ~100%. We recommend as next steps to implement our solution and continue improving our decision tree by performing cross-validation, and hypothesis testing on our performance metrics to stabilize the features of importance and strengthen our model's reliability over time.

Problem summary

AI [2] is quickly transforming the way mortgage lenders interact with borrowers and manage resources. Following a Forbes report published in 2020, 56% of mortgage lenders agree that AI revolutionizes key functions in their field. AI can find ways to predict borrowers' behavior and practices thus helping lenders earn more. The real estate lending market size is very large. Businesses in the U.S. sell over 10 million commercial properties and houses every year. According to Mordor Intelligence, USA Home Loan Market is valued at USD 4,400 billion and is expected to register a CAGR [3] of 18% during the forecasted period of 2023 to 2028!

With the advent of data science and AI, the focus is shifting from lengthy and manual processes prone to wrong judgment due to human errors and biases to building machine learning models that can look through several criteria more efficiently than humans, also learning from its previous mistakes and free of human biases that can happen in human interactions, or wrong judgment. The discussed analysis and model proposed will provide a highly accurate and automated prediction of loan defaulters, focused on interpretability and transparency of reasons of rejection to comply with banking regulations.

Solution Design

Model approach and comparison

A number of machine learning models were tried as part of the solution design, including a logistic regression that we had to discard for poor performance, and a random forest that we discarded for lack of interpretability and overfitting (see Table 1 for the overview of all our best models' performances sorted by descending performance from left to right). Based on that comparison, the best model in terms

[1] Net Revenue = Revenue net of defaulters losses and model's investments costs (implementation and running)
[2] Artificial Intelligence
[3] Compound Annual Growth Rate

of performance that we tried is the XG Boost classifier that yielded the best recall score at 87% while showing no signs of overfitting.

We've decided to choose the decision tree model instead because it also yielded a good recall score at almost 75%, an overall accuracy of 87%, increased consistency on the features of importance and a good generalization from train to test datasets. We also chose it because of its interpretability compared to the XG boost that is much harder to explain and builds a lot more randomness into it through the selection of features (see next section for the details on interpretability).

Below is displayed the *confusion matrix* of our chosen model on 1788 borrowers which composes our test dataset, a technical tool that allows us to compare the actual predictions from our model from the real outcome stored in our dataset. We see how the false negatives at 50 are much lower than the false positives at 255. We will use this confusion matrix for the business case (see costs/benefits section)

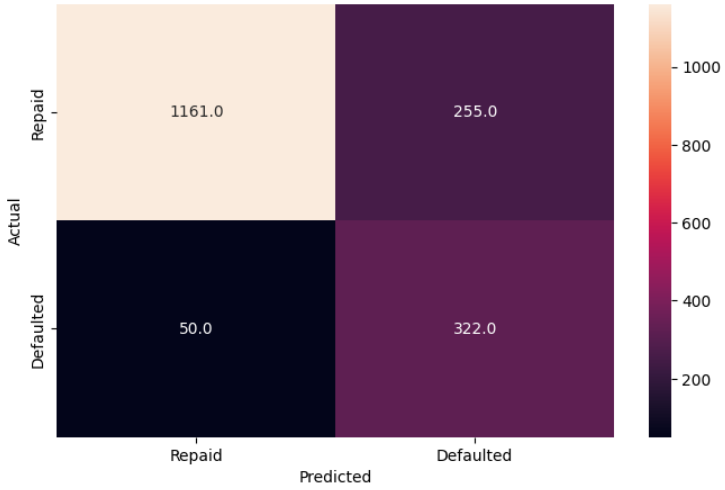

Figure 1 - displaying the confusion matrix of our chosen model applied to a test dataset of 1788 borrowers

Hence, even though we have a difference of 12 pts in the recall score with our single decision tree, that in turn gives us interpretability in the results and clarity to our bank stakeholders and customers on the reasons of acceptance or rejection.

Table 1 - Comparing machine learning models' performances and overfitting scores

	XG Boost Tree	Decision Tree	Logistic Regression	Random Forest
F1 score	68%	70%	52%	78%
Recall	**87%**	**74%**	**64%**	75%
Accuracy	83%	87%	75%	91%
Δ_f1	2.0 pts	1.0 pts	0.0 pts	17.0 pts
Δ_recall	**5.0 pts**	**4.0 pts**	1.0 pts	**24.0 pts**
Δ_accuracy	2.0 pts	0.0 pts	1.0 pts	7.0 pts

Note: In the table above, the *Δ* sign preceding each performance metric stands for *delta*, it calculates the difference of the metric value between the training and test datasets. Below 5 points difference in recall and other metrics, we reach our target threshold which means we have a robust model.

Model specifications

Our dataset was heavily imbalanced at 80% towards repayors so we were careful in splitting our dataset into test and train datasets. It came out that the test/train split of 70%/30% was a good choice, not impacting further the imbalance of the dataset in each split dataset.

To obtain consistency of performance on new unseen data points (i.e. *overfitting*), we used a concept called *pruning,* that helps reducing the size of our tree so it doesn't grow freely too deep and complex, and only allowed our performance metrics to be within a 5 points threshold difference between test and train datasets. We iterated with following parameters:

- **Weighting**: we rebalanced the weighting of our decision tree towards the underrepresented class the defaulting one, so they account for 80% of the weighting.
- **Maximum depth**: we set a maximum depth for our tree of 6 to ensure simplicity of the model and good generalization of results on new unseen data (we add an appendix for why we chose that parameter).
- **Minimum samples per leaf**: we set a minimum of 20 individuals per leaf for the same reasons as above. To find the right number, we used a process called *tuning*, that consists in iterating with different values to find the best performance – 5, 10 and 25 were tested as well.
- **Minimum split per sample**: we set the default parameter of 2 here. This indicates the minimum number of individuals required in a leaf or node before a split is allowed – 5, 7 and 10 were tested as well.

Model interpretability

Interpretability here is obtained thanks to a very clear plot of our model that can be seen in Figure 2. Decision trees are very understandable as they split into different nodes and provide values that separate our population into homogeneous subgroups.

Interpretability is also obtained thanks to a transparent list of features of importance that can be presented to regulators in case of audit or our clients if they asked for reasons of rejection (see Figure 3 - features of importance of our decision tree model).

Not surprisingly, we found that the debt-to-income ratio is the most critical feature of importance in all our models at 80%, when its impact is still the most important in our non-retained boosted model but more balanced at around a third of importance, followed by delinquent credit lines that was consistently a factor increasing the likelihood of defaulting at 10% of importance.

Other credit score related features such as the *number of recent credit inquiries* or the *number of credit lines* did provide some insights and impact on the likelihood of defaulting, at a lower level than the previously mentioned variables though.

Features that didn't impact our model's outcome were: the number of *derogatory reports* (even though it appeared to be impacting our model in the boosted decision tree which is worth monitoring in the future), the *job position* and *job longevity* even though these variables can be good proxies for job stability, financial strength and security (we added more comments on the *JOB* variable later in this report).

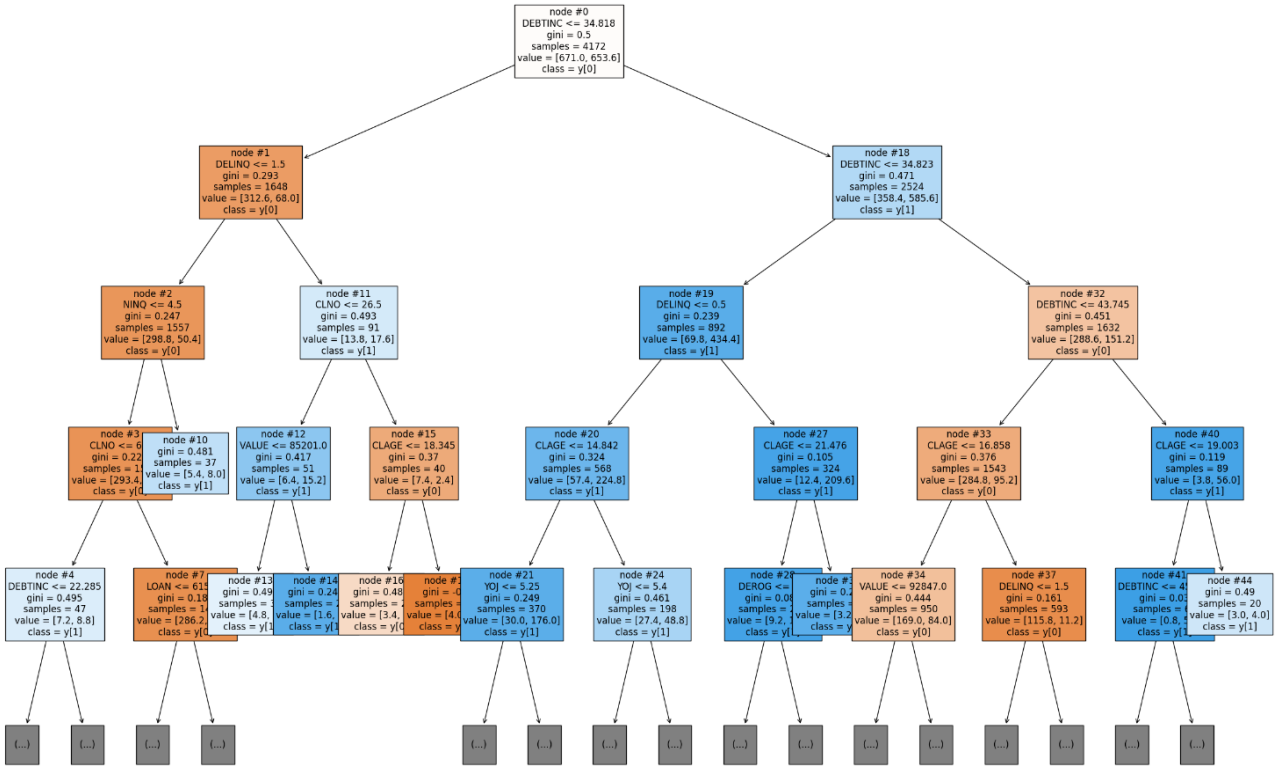

Figure 2 - decision tree model (4 levels of depth displayed here out of 5 for ease of display), Blue leaves represent the defaulters

On the loan characteristics, the value didn't seem at first to have a significant impact on the chances of defaulting (i.e., in the explanatory data analysis, when analyzed on the overall population of borrowers), but it consistently came back as one of the features of importance to our model when applied to smaller subgroups of the population, at a lesser degree though compared to previously mentioned stronger features of importance. We didn't see a meaningful impact of the type of loan (i.e., debt consolidation, home improvements) on the outcome of defaulting.

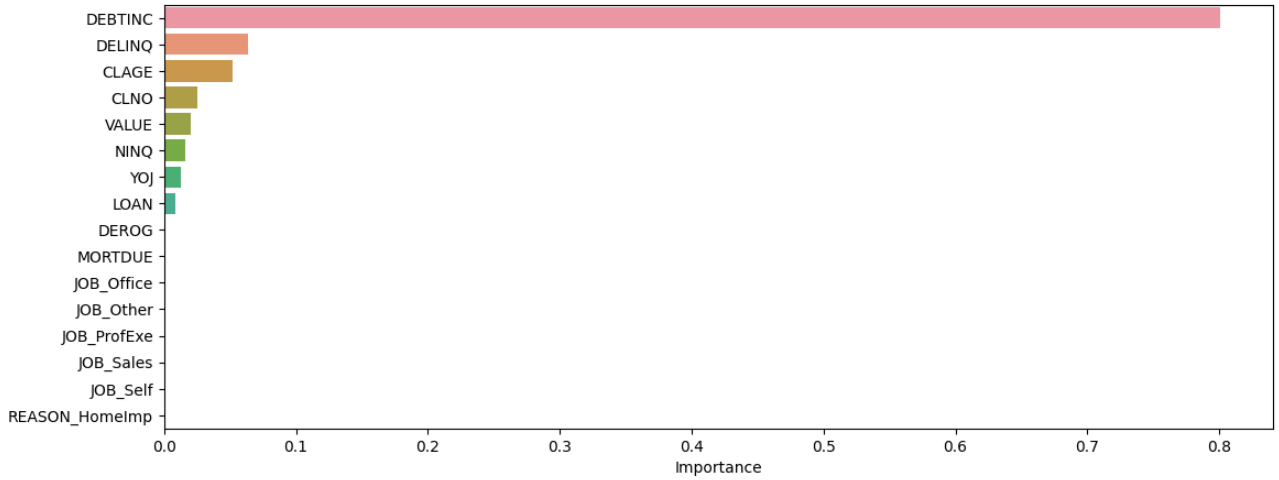

Figure 3 - features of importance of our decision tree model

Model strengths and challenges

Solution	Cost/benefits for the bank
Simplicity of our model	• The model is straightforward to understand, it's easy to present to the federal government if we get audited or to customers if they ask. • The data doesn't need to be pre-processed with scaling or synthetic creation of data points (SMOTE), it's almost ready to use (some data prep still involved).
Ease of interpretability	• Our model is straightforward in providing features of importance with exact percentages of impact on the prediction • It can also be visualized with plotting of all decision nodes impacting the prediction outcome
Computational efficiency	In terms of the cost of computation related to running our model once in deployment, we're low compared to many other more complex models out there including deep learning and random forest models.
Robustness	• Our model is not affected by outliers and missing values to the extent that our logistic regression would be (or other clustering techniques). • The model generalized very well from the training to the test dataset. We expect to perform well when used by the bank in production • It's also more robust than random forests and boosting decision trees as it involves less randomness in the selection of features (there's still a slight element of randomness in the permutation of features)
Feature selection	• Our decision tree inherently performed feature selection, using the most informative features first. • If we get more data over time, with bigger datasets and many more columns, this benefit would prove even more impactful
Cost of bias reduced	Compared to the old-fashioned manual process conducted by humans, we're eliminating bias in human decisions (if our provided dataset is itself unbiased)
Time efficiency	By running this model for our bank instead of having our previous manual process, we're gaining in operational efficiency by running the process faster (gain in work efficiency)

Solution	Challenges of the solution proposed
Performance	By being more conservative and focused on interpretability, we are also losing 12 points in the recall rate compared to the XG Boost, so performance here is definitely lowered. That drop in performance is equivalent to >1 person out of 10 we won't be able to detect as future defaulters because of our choice.
Reliance on the debt-to-income	• We rely at 80% of feature importance in our model on the debt-to-income ratio, when the XG Boost was more balanced between the different features regarding their impact on the model. • 20% of the values for that variable were missing, and we found data quality issues for that variable (outliers that weren't possible). Both issues may convey that it's hard to get this information correct in a consistent amount of situations, which puts at risk the validity of our model

Bias towards features with more levels	Decision trees can be biased towards variables with more levels. Features with more unique values or categories may be favored over others, potentially leading to suboptimal trees.
Threshold for overfitting	Even though we decided to accept a model that would stay within 5 pts of the training data performance (the dataset being very imbalanced, we assume this is a fair threshold), we are also accepting a model that doesn't have an exact same performance on test and train datasets. That could potentially present some risks in the future to not perform as well on unseen data

Recommendations for implementation

Cost/benefit analysis of deploying the model

Now that the model is built, the next step for the stakeholders is to decide if they want to deploy our model in production. This is a whole project that will require changes in the organization and investments. Let's make some assumptions and estimate the incremental revenue net of defaulters losses and investments costs for our stakeholders if they implement our solution versus not doing anything (we disregard all other costs or revenues).

Let's assume we are working for a small local US bank that provides 200k loans by year at an average value of $20k by loan. For this project, we would hire four data scientists with visualization skills at $150k a year, four machine learning data engineers at the same salary that can deploy our model and provide access to it. Let's say we choose a competitive cloud contract that would only cost us $100 by year for storage and computation, and same price for Tableau licenses to monitor results and present them to stakeholders. Let's say it will cost $100k to train underwriters from the field to the new process and deploy the interface in all branches. The price of renting yearly our dataset from the Home Equity dataset (HMEQ) would be $100k and would be needed to keep monitoring our model.

Now let's assume our decision tree model helped improve the recall by 8 points on the defaulting class compared to the old manual underwriting process that the bank currently has, and helped improve by 4 points the precision. Also because the underwriting process becomes automated, for the same number of underwriters (the bank doesn't plan to fire anyone), we improve by 10% the number of loans we can handle within a year. On the other hand, the advantage of the old manual process is to not incur any of the model implementation expenses described above (I.e., recruitment, cloud costs, training).

Finally, let's say that a false negative will represent a loss of $10k for the bank as they won't be able to get the full loan amount back plus losses on lawyer fees, and that the bank charges a flat 6% interest rate annually on its loans.

Our calculations show that the incremental net revenue [4] with our new model is of $49.6M thanks to the increased performance of the model and the efficiency increase in the underwriting process due to the automation (see detailed business case). Our missed revenues from rejected customers (that turned out

[4] Net revenue = revenue net of defaulters losses and model's implementation and running costs

to be valid future repayors) over our net revenue now dropped 96% thanks to the implementation of our solution. Thus we strongly recommend the stakeholders to implement our solution.

Key actionable items and risks for stakeholders

Business opportunity:

The bank is only providing two types of loans, home improvement and debt consolidation. Overall, it would make sense to ask the bank if they plan to provide more type of loans in the future, there are many opportunities out there. If we already have data on customers, it could also make sense to get into providing mortgages that are more lucrative. Now that we have a quick automated way to detect with higher accuracy defaulters, there might be more business opportunities for us to grow.

Missing values (see appendix 4 for more details):

21% of the values for the debt-to-income were missing, which is too high. We will raise that concern to the bank and ensure we have a process in place to get as close to 100% collection as possible for that feature, as it's so important to the validity of our predictions. Knowing that 80% of our model depends on that variable, we can't afford to have missing values here.

Debt-to-income ratio was not the only features with missing values, *DELINQ*, the number of delinquent credit lines that is our 2nd most important variable in our model, had around 10% of values missing. We absolutely need to ask the bank to fix this to ensure consistency and accuracy of our model over time

Data Quality issues:

For the *debt-to-income ratio* even if values between 50 and 100 are technically possible, we seriously doubt any bank would provide loans to such risky profiles. So maybe that means our values are incorrect, which in turn impacts the quality of our model. The bank will probably be able to respond to that and clear our doubts or fix it.

We also found data quality issues on the variable *CLAGE*, that provides the number of years of credit history. Maybe data quality issues come from the fact the data collection process is made through questionnaires and customers are lying or making mistakes when providing that information. We need to review the process with the bank and improve data quality.

Less relevant variables:

The *JOB* variable had 45% of its values that were equivalent to missing, as they refer to 'Other' jobs. In addition categories of values were not mutually exclusive making it confusing. For instance we have an *Office* category and an *Executive position*, one refers to the type of job, the other one to the seniority. Maybe a better collected variable could help us improve our model. As a reminder this variable had very little to no influence in all our models.

Also the *home property value* variable and the *outstanding mortgage due* are very correlated, and so together don't add more value to our problem. If the bank has the ability to swap one of these variables for new ones that are not correlated for the same price, such as a socio-economic criteria on the borrower (e.g., salary, age, address), that could help improve the model. As the dataset comes from the Home Equity dataset (HMEQ), that is a discussion to have with them.

Further analysis

- We will monitor consistency of our recall performance on new unseen data which will require to get fresh unseen data. We will also run hypothesis testing on this performance metric to ensure it is robust and significant.
- We will run cross-validation to ensure consistency and stabilization of features of importance over time on different training datasets which is key to the bank.
- We also might be able to get more data points or new features in the future with our yearly budget of $100k, that would likely help improving the quality of the model.
- We didn't try a clustering method on that dataset due to the many outliers that would potentially decrease significantly the quality of our clusters. There are great clustering techniques though that we could still try.
- There are other complex decision tree alternatives such as the Adaboost or neural network decision trees that we could try. These would be interesting to compare to our more sophisticated XG Boost model even though their interpretability will still be a challenge for implementation at the bank.
- We corrected the imbalance of the dataset with weighting in all our models. There are other techniques that we could try such as SMOTE algorithm, which stands for *Synthetic Minority Over-sampling Technique*, that is able to "synthetize" data points thanks to closest neighbors features. This eventually balances the dataset artificially and lets the model learn on a perfectly balanced dataset.

Bibliography

- Forbes Insights, July 7, 2020. Key Takeaways On the Rise of AI in the Mortgage Industry.
  Key Takeaways On the Rise of AI in the Mortgage Industry - FORBES
- Mordor Intelligence, 2019 - USA Home Loan Market Size & Share Analysis - Growth Trends & Forecasts (2023 - 2028).
  USA Home Loan Market Size & Share Analysis - Growth Trends & Forecasts (2023 - 2028)
- AWS, Type of offers and top product categories
  AWS, Type of offers and top product categories

Appendix

Appendix 1: Definitions

- **Accuracy** is the fraction of total ***observations*** that are predicted correctly by the model. It is suitable for cases where the data is balanced among the existing classes
- **Classifier:** type of machine learning model that tries to predict to which class an individual belongs. **Confusion matrix:** summary table that compares the actual predictions from our model from the real outcome from our dataset
- **Dataset imbalance:** Machine learning term that refers to a sample being over represented with one class vs. the other one (same thing with more than 2 classes which is not our case here)
- **F1-score** is used when ***precision*** and ***recall*** both seem to be equally important, or any one of them may not be preferred over the other
- **Feature/variable:** columns of our provided dataset (e.g., debt-to-income ratio, number of credit lines) that feeds our machine learning model in order to make predictions.
- **Overfitting:** when the model's performance usually good cannot be replicated on the test dataset.
- **Precision** is the fraction of total positive predictions that are actually correct. In other words, out of all the values predicted as defaulting, how many are correctly predicted.
- **Prediction:** Machine learning term that refers to the outcome provided by the model, i.e., the customer will default or not.
- **Recall** is derived from the confusion matrix and is the fraction of total actually positive cases that are predicted correctly. In other words, out of all the actual positive cases, how many were correctly classified.
- **Train & test datasets:** the dataset is provided split into training and test datasets. The training dataset will be used to train the machine learning model, the test dataset will be used to validate the performance of the model.

Appendix 2: maximum depth parameter choice

Our tuned decision tree with all the parameters we presented in the model specifications were reaching an optimal value for the recall score. Our only concern was about the recall score gap between our train dataset and the test dataset, that was above our set threshold of 5 points (at 7 points).

We decided to further explore ***the bias-variance tradeoff***, that consists in finding an optimal balance between the ***bias***, which measures the quality of our prediction, and the ***variance***, that focuses on the sensitivity of small fluctuations in the training set, that basically may lead to overfitting.

To do so, we calculate our main performance metrics for difference values of the *maximum depth* parameter mentioned in the model specifications:

```
# Defining different set of max depths
md_range = np.arange(1, 20, 1) # mdr stands for max depth range

# Initiating different series to fill later with values
perf_metrics = ['f1', 'recall', 'acc'] # stands for f1 score, recall, and accuracy
test_d_tree_mdr_f1, test_d_tree_mdr_recall, test_d_tree_mdr_acc = ( [] , [] , [] ) # mdr stands for max depth range
train_d_tree_mdr_f1, train_d_tree_mdr_recall, train_d_tree_mdr_acc = ( [] , [] , [] )

for md in md_range:
    d_tree_mdr = DecisionTreeClassifier(class_weight={0: 0.2, 1: 0.8}, max_depth=md, min_samples_leaf=20, random_state=1, criterion='gini', min_samples_split = 2)
    d_tree_mdr.fit(X_train, y_train)

    # Calculating f1-score, recall and accuracy for each train and test dataset
    d_tree_mdr_train=model_performance_classification(d_tree_mdr, X_train, y_train)
    d_tree_mdr_test=model_performance_classification(d_tree_mdr, X_test, y_test)

    # Storing these metrics values in our respective series
    for metric, col in zip(perf_metrics, d_tree_mdr_train.columns): eval(f'train_d_tree_mdr_{metric}').append(d_tree_mdr_train[col][0])
    for metric, col in zip(perf_metrics, d_tree_mdr_test.columns): eval(f'test_d_tree_mdr_{metric}').append(d_tree_mdr_test[col][0])
```

And plot it thanks to following python code:

```
y_labels = d_tree_mdr_train.columns.to_list()

fig, axes = plt.subplots(
        ncols = 3,  # Number of columns of the subplot grid
        nrows= 1, # Number of rows of the subplot grid
        figsize=(20, 4)
    )

# set the spacing between subplots
plt.subplots_adjust(wspace=0.4,
                    hspace=0.4)

# add the countplots to the subplots grid
for metric, axes, y_labels in zip(perf_metrics, axes.reshape(-1), y_labels): # iterating over columns and axes
  axes.plot(md_range, eval(f'train_d_tree_mdr_{metric}'), c='black', label='Train')
```

```
16.   axes.plot(md_range, eval(f'test_d_tree_mdr_{metric}'), c=my_color, label='Test')
17.   axes.set_title(y_labels)
18.   axes.set_xlabel('Tree depth')
19.   axes.set_ylabel(metric)
20.   axes.axvline(x=5, color='red', linestyle='--', linewidth=0.5) # Optimal solution to
    our problem represented by a red line
```

That gives us following graphs (only showing recall and accuracy for ease of display):

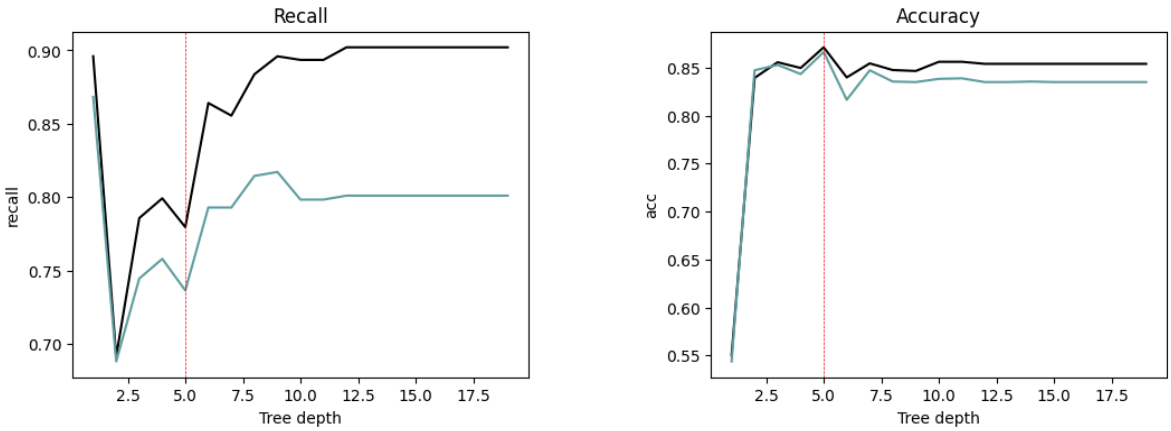

Figure 4 - Bias-variance tradeoff for different values of maximum tree depth

Our optimal solution was found by constraining the depth of our decision tree a little bit more at 5, instead of 6, which in turn reduces slightly our recall performance (i.e., bias), to ensure consistency of results in the future (i.e. variance).

Appendix 3: business case for implementing our new solution

Expenses	by year	*Revenues*	by year
Data scientists (x4)	($600,000)	Loans handled	200,000
ML engineers (x4)	($600,000)	Average Loan value	$20,000
Cloud contract	($100)	Interest	6%
Tableau licenses	($100)	Loan default loss	($10,000)
Home Equity dataset	($100,000)		
Training & deployment	($100,000)		

Decision tree model performance	
Precision	66%
Recall	74%
Defaulted (predict)	15%
Repaid (predict)	71%
False Negative	5%
False Positive	8%
Loan increase	10%
Scenario 1: deploy our decision tree model	
Model implementation	($1,400,200)
Loans handled	220,000
Defaulted (actual)	12,058
Defaulted losses	($120,581,655)
Declined	33,714
Repaid (actual)	156,879
Revenue loans	$188,255,034
Missed loans	17,349
missed revenue	$20,818,792
Net revenue	**$66,273,178**
Missed revenue	**$20,818,792**
Incremental Net Revenue	**$49,551,925**
Decrease % Missed Rev.	**-96%**

Old manual process performance	
Precision	62%
Recall	66%
Defaulted (predict)	14%
Repaid (predict)	69%
False Negative	7%
False Positive	9%
Loan increase	-
Scenario 2: keep our old process	
Model implementation	$0
Loans handled	200,000
Defaulted (actual)	14,962
Defaulted losses	($149,619,687)
Declined	30,649
Repaid (actual)	138,617
Revenue loans	$166,340,940
Missed loans	17,772
missed revenue	$21,326,174
Net revenue	**$16,721,253**
Missed revenue	**$21,326,174**

* Assuming here that the cost of time is $0, as the project will likely take a few months to be completed before we start seeing the performance and efficiency benefits, even though expenses will be deducted.

Also assuming one-year loans to simplify the revenue calculations.

Appendix 4: Comments on the dataset provided

- The dataset has 5960 entries and 13 columns**. Our dataset was imbalanced towards the defaulting category as the population of defaulters is only representing 20% of the total population**. An imbalanced dataset is an additional challenge to overcome in our machine learning journey, as it makes the most important class rarer in the dataset and therefore harder for the machine to learn.
- **On data quality, we found out that the variable representing debt-to-income ratio (*DEBTINC*) and the one representing the number of years of credit history (*CLAGE*) had outliers that didn't seem valid**. We clipped reasonable values for these variables when the debt-to-income ratio was above 100, and the years of credit history above 70 years. Note that outliers are not necessarily good or bad in the dataset, they just are worth checking to ensure there is not a data quality issue that would negatively impact our model
- **Missing values were a challenge in this dataset, as most features were missing values**, and important features such as the debt-to-income ratio, or the number of delinquent credit lines were missing at above 20% (see below the full report on missing values). These missing values were replaced by the mean for numerical variables, and the mode (i.e., most frequent value) for categorical variables such as *loan reason*, or *job position*.

Table 2 - Missing values in our dataset (total value and percentage of total)

index	missing	Missing (%)
DEBTINC	1,267	21%
DEROG	708	12%
DELINQ	580	10%
MORTDUE	518	9%
YOJ	515	9%
NINQ	510	9%
CLAGE	308	5%
JOB	279	5%
REASON	252	4%
CLNO	222	4%
VALUE	112	2%