**Machine Learning Project**

Goal of this project is to practice what we have discussed so far. We will define a new credit card strategy (based on a new ML model), and compare it with the existing strategy.

1. Download data from <https://www.kaggle.com/competitions/amex-default-prediction/data>. We will work with “train\_data.csv” and “train\_labels.csv”.

**Project and Data Explanation:**

***I suggest to read “Chapter 11 – I am a Data Scientist 2”, before continuing.***

Business needs a Default Risk model. The model will be used in Credit Approval Decisioning; i.e. to decide whether to approve an application for a Credit Product.

Modeling team will build the model. Strategy team will use this model’s output to design a Credit Approval strategy. I this project, we will build this model and design a strategy in this project.

Chapter 11 discussed the following steps for a modeling project. Red steps are already done in this project. We first need to know what went on there. Make sure you clearly understand these steps. You will receive questions on them.

1. Model Design
   1. Target Definition
   2. Sample Definition
2. Data Collection
3. Data Cleaning
   1. Feature Exclusion
   2. Observation Exclusion
4. Data Processing
   1. One-Hot Encoding
   2. Outlier Treatment
   3. Feature Scaling
   4. Missing Value Imputation
5. Feature Reduction
6. Model Training
   1. Grid Search (Hyper-parameter Tuning)
   2. Bias/Variance Analysis and Finalizing the Model

The first step is to define target variable. Target is 0/1, with 0 indicating no default and 1 indicating default. How is default defined? We don’t have that information. We just know that “train\_labels.csv” shows target variable (default / no default) for some of bank’s customers as of April 2018. For example, if default is defined as “Missing 2 payments in the next 1 year”, train\_labels shows whether applicant defaulted in the next one year; i.e. May 2018 to April 2019.

The second step is to define the modeling sample. Modeling team will use this sample to develop the model; i.e. the sample will be used for Test(s) and Train samples. This step is also done. Modeling team has decided to use “April 2018 Originations” to build the model. These are the customers who received a loan in April 2018, and we have enough historical data for them to calculate the target variable. For example, if like above default is defined as “Missing 2 payments in the next 1 year”, then we should have one year of data for these customers, so we can calculate whether this customer defaulted in the next 1 year or not. In other words, we need to have data for these customers from May 2018 to April 2019. So these are the cases who have been our customer, at least for 1 year.

Why do we use only April 2018? Why don’t we use other cohorts (like May 2018 originations, or like a period, like 2021, …). That is a decision that is made in the design phase, and as mentioned is already done.

**Q.** What criteria to consider when defining the dev sample? Answer: Quality and Quantity of data. Read the very important “Chapter 7 - I am Data… Bias/Variance and Sample Bias”.

So we discussed model design and its two steps:

1. Target Definition
2. Sample Definition (we didn’t discuss Test/Train split, will discuss it later)

Next step is data collection and data cleaning. These time-consuming steps are also already done. But be ready for them as one of the first tasks that will be assigned to you as a data guy.

We have data on target variable, now we need data for independent features. “train\_data.csv” shows data available for these customers as of April 2018. Data is from April 2017 to April 2018; i.e. 13 months of data. So when the customer applied for a loan in April 2018, we had this information about the customer (13 months of historical data from April 2017 to April 2018). Modeling team has decided to use this data to define features.

**Note** that we don’t have 13 months of data for all customers. For some we have less. They provided us with less months of information, for any reason.

We may have more than 13 months of data. Why do we use 13 months to define features?

This question is similar to how we define Target. The decision would be made in the design phase. Maybe modelers think more than 13 months is very old, and 13 months of data is the best to define default in the next 12 months (like how we defined target variable).

As mentioned, feature exclusion and observation exclusion steps are also already done. Some observations may have been removed, like to mitigate sample bias.

**Exam/Project Question.** Think of possible sources of sample bias for this project. You can come up with a story for model’s application, and think of some sources of sample bias.

Feature exclusion is also already done, and data is clean. Even several steps of data processing has been done. All features are scaled, and probably before this step, outliers are removed. Also I think Missing values are imputed, but I am not sure! So you may need to do that part.

Now that we understood what has been done, let’s start the project.

1. The data might be too large, and you may get memory error while doing the project; so we will use only 20% of observations. Randomly choose 20% of observations from the “train\_labels.csv”. Merge this sample with “train\_data.csv” to have features for these applicants. This will be our development sample. **Save this data**, so in the future you don’t have to read the original large file again.
2. Explore the data. Data Size, data type of features, a snapshot of data, …
3. Perform One-Hot encoding on categorical variables.
4. Next we want to define some features. As mentioned, we have historical data for up to 13 months for each applicant. For some applicants less than that. We need to aggregate these up to 13 months.

For **Numerical** features, aggregation can be done by: Average, Sum, Min, Max,… Also I suggest you include feature’s value as of April 2018, which is the most recent value.

Here are some examples for some aggregated features based on feature X\_1:

* X\_1\_Ave\_6: Average X\_1 in the last 6 months
* X\_1\_Ave\_12: Average X\_1 in the last 12 months
* X\_1\_Min\_6: Minimum X\_1 in the last 6 months
* X\_1\_Max\_9: Maximum X\_1 in the last 9 months
* X\_1\_Sum\_3: You know
* X\_1\_Apr\_2018
* You name it: (X\_1\_Apr\_2018 – X\_1\_Apr\_2017)/ X\_1\_Apr\_2017
* …

As you can see, you can define many many features. Do that, Model will choose for you, the ones that have real predictive power. Try to come up with a feature that adds to the model.

**Note:** For some observations you have less months of data. So the above features may be calculated with less months. For example, for an application with 4 months of data, X\_1\_Ave\_6 will be calculated based on average of X\_1 in the last 4 months.

Sometimes people make some decisions for these cases. For example, you may decide that if there is less than 2 months of data for an observation, then X\_1\_Ave\_6 would be recorded as missing. I don’t suggest that.

For **Categorical** features, some examples for aggregation are as following. Note that you have already done one-hot-encoding and your categorical features are binary (0/1). In fact they the features, are categories of categorical features, one-hot encoded.

* X\_1\_Response\_Rate\_6: Percentage of times X\_1 equals 1 in the last 6 months.
* X\_1\_Ever\_Response\_12: Whether X\_1 is response at least once in the last 12 months
* X\_1\_April\_2018
* …

1. Split data into 70% as Train sample, 15% as Test1, and 15% as Test2.
2. Next we want to reduce number of features, and keep only features which have high predictive power. To do so we build an XGB model and will keep features with Feature Importance higher than 0.5%.

**Make sure all missing values are stored as NaN, so XGBoost can work with them.**

1. Run an XGBoost model on the train sample, with default parameters. Don’t forget to drop unnecessary columns if any. Calculate feature importance and save the feature importance as a CSV file.
2. Run another XGBoost model, which has 300 trees, 0.5 as learning rate, maximum depth of trees is 4, uses 50% of observation to build each tree, uses 50% of features to build each tree, and assigns a weight of 5 to default observations. Save the feature importance as a CSV file.
3. Keep features that have feature importance of higher 0.5% in any of the two models. We will use only these features after this.
4. Next we run Grid Search for the XGBoost model (using only features we chose in step 10). Use the following combinations in the grid search:
   * Number of trees: 50, 100, and 300
   * Learning Rate: 0.01, 0.1
   * Percentage of observations used in each tree: 50%, 80%
   * Percentage of features used in each tree: 50%, 100%
   * Weight of default observations: 1, 5, 10

Create the following table. **Update the table after each iteration of grid search and save the table**, so in case you got memory error or any other issues, you don’t need to re-run that part of Grid.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # Trees | LR | Subsample | % Features | Weight of Default | AUC Train | AUC Test 1 | AUC Test 2 |
| 50 | 0.01 | 50% | 50% | 1 | … | … | … |
| … | … | … | … | … | … | … | … |

**Note, optimum would be to use all features in the grid search. Also optimum is to test all the possible parameters for grid search! But we sacrifice a little in model’s performance, but gain a lot in computational efficiency. At the end, the sacrifice has minor impact on model’s performance, and even lower impact on the strategy and business results.**

1. Choose the best model, based on bias and variance. Re-run the model with optimum parameters, and save the final XGB model.
2. Next, grid search for Neural Network. We first need to process the data. We have already done one-hot encoding. We need to do Missing Value Imputation, Outlier Treatment, and Normalization. **We will use only features that we chose in step 10**. As mentioned, probably there is no need for outlier treatment and feature scaling; but to practice, cap and floor observations at 1 and 99 percentiles. Use StandardScaler for normalization (standardization). Replace missing values with 0.

**As you know, you should get values for 1 and 99 percentiles, as well as Mean and Standard Deviation values for scaling, only based on the Train sample. Later you should apply the same value to Test samples (or any other sample). In other words, for each observation in the test sample (or any other sample), you should first do outlier treatment based on 1 and 99 percentiles of the train sample, and Standardize it based on Mean and Standard Deviation from the (capped and floored) train sample.**

1. Next we run Grid Search for the Neural Network model. Use the following combinations in the grid search:
   * Number of hidden layers: 2, 4
   * # nodes in each hidden layer: 4, 6
   * Activation function for hidden layers: ReLu, Tanh
   * Dropout regularization for hidden layers: 50%, 100% (no dropout)
   * Batch size: 100, 10000

Use Adam for optimizer, Cross Entropy for Loss function, and 20 for number of Epochs. For everything else, use default parameters.

**Note you would need to run separate For Loops for different number of Hidden Layers.**

Create the following table. **Update the table after each iteration of grid search and save the table**, so in case you got memory error or any other issues, you don’t need to re-run that part of Grid.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # HL | # Node | Activeation Function | Dropout | Batch Size | AUC Train | AUC Test 1 | AUC Test 2 |
| 2 | 4 | ReLu | 50% | 100 | … | … | … |
| … | … | … | … | … | … | … | … |

1. Choose the best model, based on bias and variance. Re-run the model with optimum parameters, and save the final NN model.
2. Choose the best model among NN and XGB (models of step 11 and step 14)

**Strategy:**

Next, you want to define two strategies: a conservative and an aggressive. For each strategy, you define a threshold to accept/reject applicants based on the model’s output. Applicants with probability of default (model’s output) lower than threshold, will be accepted, and those with PD higher than threshold will be rejected. The conservative strategy has a lower threshold compared with the aggressive one; hence accepts less applicants.

We will estimate Portfolio’s default rate, and Revenue based on each strategy, show it to management, and let them decide which strategy is better.

Estimate Portfolio’s Default Rate: You already know how to calculate default rate for a strategy; you just need to calculate default rate among applications that will be accepted based on the strategy, i.e. those with PD less than threshold.

Estimate Portfolio’s Revenue: Revenue on a credit card depends on two factors: how much the customer spends, and how much of monthly balance the customer does not pay (roll over to the next month). Credit Card companies, charge a small amount for each dollar you spend. Also they charge an interest rate on the remaining monthly balance that you do not pay (revolving balance).

For example, assume a CC charges 0.1% on each dollar spent, and charges 24% (annually) on balances. If a customer spend $1000 in a month, company’s revenue from spend of this customer will be 1000×0.001=$1. If customer pays back $200 out of $1000, company will charge 2% monthly interest on the remaining $800, which means $16 interest revenue in that month.

**Note:** As you know interest rates on CC balances are very high, so don’t be manipulated by banks; i.e. don’t spend too much. You are most attractive with a cheap, healthy life, with a lot of exercise. Also Never Default on your Debt; i.e. never miss the minimum monthly payment.

So, to estimate revenue, you need to have a measure of Spend and Balance, in the next few months. In other words, just like default that we checked payments in say 12 months after origination, we need to have information on spend and balance in say 12 months after origination. If we have that information, we may be able to build ML models for spend and balance. For example, Spend model in this case estimates “Expected Spend in the next 12 months Conditional on Independent Variables.”

However in this data we have no information on spend and balance after origination. Note that the only information we have about after-origination period is 0/1 indicator in the train\_labels.csv. Since we don’t have spend and balance data, we use historical data on balance and spend for each customer. Basically we are assuming historical spend and balance is a good predictor of spend and balance in future.

In the data, features that start with S\_ are spend variables, and features that start with B\_ are balance variables. Choose one spend and one balance feature (any feature of your choice). Calculate average of these two features for the last 6 months (i.e. November 2017 to April 2018). If we show these two averages with S\_Ave and B\_Ave, monthly revenue for a customer would be calculated as:

And Expected Revenue in the next 12 months would be 12 multiplied by the above value.

To estimate portfolio’s expected revenue based on a strategy, calculate sum of the above revenue among customers who are accepted based on the strategy. Assume a revenue of 0 for those who default.

**Note**: To estimate Balance and Spend you have built a model. It is a very simple model, which is just the average.

1. Write a function that calculates default rate and revenue based on a threshold. Function gets sixe inputs:
   * Data with four columns: Target Variable (Default indicator), Default model’s output (PD), Estimated Monthly Balance, Estimated Monthly Spend
   * Name of Target Variable (as a text/string)
   * Name of default model’s output (as a text/string)
   * Name of Estimated Monthly Balance variable (as a text/string)
   * Name of Estimated Monthly Spend variable (as a text/string)
   * Threshold (a number between 0 and 1)

And will return two outputs: portfolio’s default rate, and portfolio’s expected revenue.

Use only train sample to try a few thresholds, and choose one conservative and one aggressive strategy. It is up to you how to choose the thresholds. Imagine you want to present it to senior management and want to impress them with your work/results. The only constraint is that company does not want the default rate to be higher than 10%.

**Prepare the presentation:**

**General Guidelines:**

1. Create pretty slides
2. Don’t use any background
3. Format numbers, use 1000 separators. Decimal numbers with 2 decimal places (in case of very small numbers with 3 decimal places)
4. Don’t use small fonts that can not be seen
5. Don’t put too much material in a slide
6. Each slide should be self explanatory. While you don’t want to put too much material, put enough material that explain the stuff in the slide
7. Format tables. Assign appropriate titles to tables and figures
8. Don’t copy paste from your code
9. Have a good story to tell
10. Format everything. Standard fonts …
11. Use colors, but don’t overuse

In general, remember a presentation is like presenting a product. Both packaging and functionality matter. **You need to wrap your good model in a pretty package.**

**Note 1:** In the following steps, feel free to change the format of tables to make the slides easier to follow and understand.

**Note 2:** I have proposed the minimum items to be included in the slides. Feel free to add additional explanations, …

**Note 3:** Due to computational constraints, you may need to simplify the project, read less rows and observations, … Adjust the following tables based on your final sample.

Fill the attached deck with your results.



**Slide #1. Executive Summary**. This is where you sell your model. Show the results of your strategies, and add any explanation that can attract people. In this slide imagine you are a seller. Include the following table. Talk about the project, project’s goal, why this project is important, how it helps the company, and anything that might be interesting (like these days people get excited when they hear AI …)

Propose the strategy that you think help the company better. Explain why you think this is a better strategy.



**Slide #2. Data.** Explain your data (data of step 3). Explain why you chose April 2018 originations (come up with a story). Include the following table, explain why you decided to use this data, explain your target variable (you can generate a story for what default means), …



**Slide #3. Features.** Talk about categories of independent variables used in the development process (data of step 3). Use raw features; i.e. features as they are in the raw data, and before defining new features in step 5.



**Slide #4. Feature Engineering.** Talk about type of features you have created (step 5). You can talk about categories, such as Average, Median, Min, Max, …

Add a table like table of slide 3, this time not for raw features, but for features you have defined based on raw features.



Also show summary statistics for the top 5 features with highest SHAP values in the best XGBoost model (Step 12). Note that at this point you don’t need to talk about the XGBoost model and SHAP. You can just mention that based on your analyses these are among the most important attributes.



**Slide #5. Data Processing / One-Hot Encoding.** Show the categorical variables, and show how you treated them. Show the results after One-Hot Encoding. Include your code to do one-hot encoding.

**Slide #6. Feature Selection.** Add a graph that explains your feature selection process (steps 7 to 10). Create a pretty graph. Attach an excel file with results of feature importance for two models (steps 8 and 89 Add a column to table of slide 4, that shows # features selected from each category to be used in grid search.



**Slide #7. XGBoost - Grid Search.** Include your grid search code. Explain why you chose these parameters (don’t say because you said …). Talk about your experience with grid search, how many models you trained, any lessons learned, …

**Slide #8. XGBoost - Grid Search.** In this slide, we create scatter plots for models of grid search, and will choose the best model based on the scatter plot. For each of the models of grid search, calculate average and standard deviation of AUC across three samples (train and tests). Then include 2 scatter plots in the slide:

* In the first one, X\_Axis shows Average AUC, and Y-Axis shows Standard Deviation of AUC.
* In the second one, X-Axis is AUC of train sample and Y-Axis is AUC of Test 2 sample.

Explain which model you would choose based on each scatter plot.

**Slide #9. XGBoost – Final Model.** Show the parameters of the final model, also AUC of model on each sample. Also show how model Rank Orders on each of the three samples. Check the last part of XGBoost sample code, for rank ordering. Note you need to define score bins based on the train sample, and apply the same thresholds to test samples. Show rank orderings in a Bar-Chart, where each sample is one series in Bar Chart, X-Axis shows score bins (intervals), and Y-Axis shows default rate in each bin.

**Slide #10. XGBoost – SHAP Analysis.** Show Beeswarm Graph for the final model, based on Test 2 sample. Add some explanation of your choice. You can talk about ranking of attributes, correlation between attribute and the output, …

**Slide #11. XGBoost – SHAP Analysis.** Show Waterfall Graph for the final model, based on one observation in Test 2 sample. Add some explanation of your choice. You can talk about which attributes are driving the score, how to improve the score, …

**Slide #12. Neural Network – Data Processing.** Explain your data processing for Neural Network. Feel free to add code, tables, … Format this slide, so it is easy to follow and understand.

**Slide #13. Neural Network - Grid Search.** Include your grid search code. Explain why you chose these parameters (don’t say because you said …). Talk about your experience with grid search, how many models you trained, any lessons learned, …

**Slide #14. Neural Network - Grid Search.** In this slide, we create scatter plots for models of grid search, and will choose the best model based on the scatter plot. For each of the models of grid search, calculate average and standard deviation of AUC across three samples (train and tests). Then include 2 scatter plots in the slide:

* In the first one, X\_Axis shows Average AUC, and Y-Axis shows Standard Deviation of AUC.
* In the second one, X-Axis is AUC of train sample and Y-Axis is AUC of Test 2 sample.

Explain which model you would choose based on each scatter plot.

**Slide #15. Neural Network – Final Model.** Show the parameters of the final model, also AUC of model on each sample. Also show how model Rank Orders on each of the three samples. Check the last part of XGBoost sample code, for rank ordering. Note you need to define score bins based on the train sample, and apply the same thresholds to test samples. Show rank orderings in a Bar-Chart, where each sample is one series in Bar Chart, X-Axis shows score bins (intervals), and Y-Axis shows default rate in each bin.

**Slide #16. Final Model.** Talk about the final model (XGBoost or Neural Net), and why you chose this one. Add tables or graphs from previous steps to support your reasoning …

**Slide #17. Strategy.** Include the function you have written in step 17. Also include the following table. Explain what thresholds you chose for conservative and aggressive strategy, and explain your rationale.

