**Team 21:**

Code Flow:

* Add all the Data-Files to the Datasets folder before running the next codes.
* The First File to Execute is the “1.Deliquency.ipynb” file.

1. Here, we perform data pipelining

* The Second File to execute is the “2.Pre-Processing XGB Model.ipynb.”

Here, I pre-processed the data and applied

XGB model on it. Here You will also see some Extra Models (Note: This Model is not considered.)

* The Third File to execute is “3.Final\_Model.ipynb.” This file contains the Neural Network Model, and its prediction is used to predict the outcomes. There is some Extra Model like NN and XGboost.

Data Pipeline:

* We started with the “UTD\_DPD” and “UTD\_Outcome\_Payment” files to check the Delinquency Log of Every Record.
* Following is the snapshot for Visualizing Log for that account.

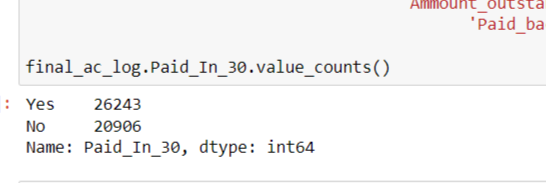
A screenshot of a computer

Description automatically generated

* We then formulated a Function that would capture the two scenarios. The first one is “if the payment was done by the customer within 30 days or not”.

The second is, “If the payment was not made in the first 30 days, it was repaid within the following 30 dates.”

* This would give us a brief Idea about who went Delinquent and paid within the next 30-day span, and if not, then did the customer payback to the bank in the future or not.
* We then create an “Accounts\_that\_went\_deliquent log” containing the information about the customer who went Delinquent at some point.
* Now, consider the account which has wholly gone delinquent for the 60 days straight after observing the 90 days log for that customer. We would consider it a **Negative Class.**
* While those customers who didn’t fail to pay within 30 days and took their recent reading, we consider them **Positive Class.**
* This information is stored in two files in CSV format, namely “Delinquency” and “Final\_account\_log.”
* Following is the snapshot of Positive Class (“Yes”) and Negative Class(“No”).



With the help of this information, We can train a Machine Learning Model.

Models

* Before applying the model, we handle the null and missing values.
* We also dropped variables that had same piece of information or very high missing values and didn’t add any value to the model.
* We dropped variables with high correlation and necessarily depicted the same information.
* Variables like number of calls made, contact made, etc. having missing values were replaced by 0 because no contact was made.
* We  handled the variables like today minus time contacted, today minus contact with no promises, which had “NaN” values with -1 because those “NaN” just meant they were not reached, and replacing them with zero would change the meaning as they were recently contacted or contacts just today or yesterday

If the column has more than 30% of missing data, we drop those columns.

* We then checked the Highly Correlated columns, and after dropping those, we are ready to apply ML on them.

Variables

* We have stored the variables which we have used to train our model in the folder named Selected\_variables.

Final Model

* Our model estimates how likely the account was to pay us back
* (For example, if the model outputs high probability, the account will pay us back within 30 days of becoming delinquent, and if it outputs low probability, then the account will not pay us back.)
* We used a Neural Network model and One hot encoded output class.
* Final layer of the model has two neurons, 1st neuron outputs if the probability of the account belonging to the Negative, i.e., probability of not paying us back
* The 2nd neuron outputs the probability of the account belonging to positive, i.e., probability of paying us back.
* We will take these probabilities to calculate our AUC score as shown.

Graphical user interface, text, application, Word

Description automatically generated

* The model we submitted is not in pickle format because we used the TensorFlow library, which has a .save method built in.
* We have also provided a method of loading the model if required.
* We also provided two alternative models, namely “NN1” and “XGB” as they were our 2nd and 3rd best performing models.