Good afternoon everyone, we’re team 3. Today, we will have a presentation about our final project.

These are the contents.

Before we start our presentation, we’d like to briefly introduce our model. We applied the CatBoost model and achieved an F1 score of 0.5007 in our final results. Now we’ll present the process of how we achieved this score.

**//**

**<Data exploration, visualization>**

We utilized two key strategies in our analysis. Firstly, we create a 'year' derivative variable. and eliminate the 'euribor3m' and 'nr.employed' columns. Secondly, we addressed potential data bias by preprocessing duplicate columns after excluding 'id' and 'y', and retaining only rows with a 'yes' value.

We noticed that the given data set was highly imbalanced from the beginning. As shown in the graph below, a high proportion (around 90%) of the 'y' values are 'no'. Therefore, we put a lot of effort into data preprocessing and sampling.

As we have already covered the visualization aspect in our previous proposal presentation, we will be skipping it in today’s presentation.

We used df.describe() for data examination, both train and test datasets. This was done on the advice of a professor to compare the distributions of the test and train sets. All of the figures show very similar statistics, which leads us to believe that data providers probably split the train and test sets in a stratified way.

Also, we used the information found by matching the economic indicator (consumer confidence index) of Portugal with the cons.conf.idx column one by one. By doing this, we could create the “year” derivative variable and delete the ‘euribor3m’ and ‘nr.employed’ columns as we checked the multicollinearity in the heatmap.

We figured out that dimensionality reduction techniques such as Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are another way to solve the multicollinearity problem. So, we tried both PCA and t-SNE, and then visualized the results.

As the visualization shows, we couldn’t find any meaningful results from dimensionality reduction. Instead of using these techniques, we decided to delete the ‘euribor3m’ and ‘nr.employed’ columns as we initially did. It was less computationally expensive but still able to cope with the multicollinearity problem.

We compared every row with all the values of all columns except the ‘id’ and ‘y’ columns. We found there were some rows where only their 'y' values were different. So, we assumed that in some cases, the consumer's intent to subscribe to the bank campaign changed from ‘no’ to ‘yes’. If the dataset included the time series, we might have been able to analyze it in more detail, but we couldn’t as there were no time series. Thus, we preprocessed the duplicate columns, excluding the ‘id’ and ‘y’ columns, leaving only rows with a y value of ‘yes’. Through the aforementioned processing, we also aimed to mitigate data bias.

We also replaced all unknown missing values with their mode values in columns ‘job', 'marital’ and ‘loan’. In addition, the ‘pdays’ column, was originally planned to replace all missing values with zeros, but during the model training process, it was found that the performance of the model decreased when the column was processed or removed, so we proceeded without processing missing values and deleting the column. Also, after creating the dummy variable, there were no rows in the test set with a ‘default’ column value of ‘yes’, so we deleted it from the train set as well.

Lastly, we scaled the dataset with dummy variables in it using MinMaxScaler, and moved on to the next data sampling steps.

**<Data preprocessing - sampling>**

We especially focused on sampling to address the data imbalance problem. We tried various sampling methods, including oversampling, undersampling, and a combination of both. Each sampling method was paired with the CatBoostClassifier.

First, we used oversampling. We applied ADASYN, which addresses class imbalance in datasets by generating synthetic samples of the minority class. The model sampled with ADASYN achieved an F1 score of 0.94. However, such a high F1 score could be the result of overfitting, so we decided to explore other sampling methods.

Next, we used two hybrid sampling techniques, SMOTEENN, which is a combination of SMOTE and ENN. This method synthesizes minority oversampling technology combined with edited nearest neighbors. This approach also resulted in a high F1 score of 0.97, leading us to conclude that it might also be the result of overfitting.

The F1 score using SMOTETomek, which combines SMOTE and Tomek Links, was lower than those of the other models and was not useful.

Finally, we tried undersampling techniques, including Tomek Links, One-Sided Selection, and NCR.

And these are the results for each sampling method.

Tomek Links helps clean the dataset by removing specific instances, thereby improving the quality of the dataset for machine learning models.

One-Sided Selection addresses class imbalance by selecting a subset of the majority class while preserving minority class samples, thus enhancing classification performance by reducing noise.

NCR (Neighborhood Cleaning Rule) combines sampling techniques such as CNN and ENN. It operates based on the neighbors of each data point, removing unnecessary noise by examining the distribution of neighbor classes.

Considering the results of every method we tried, we concluded that NCR reliably increased F1 scores. Therefore, we decided to use only undersampling techniques in the model selection process. //

**<Model Selection>**

Next, let's talk about model selection. To establish a preliminary guide for model selection, we compared several models using the PyCaret library. PyCaret is an AutoML library that simplifies machine learning tasks, including data preprocessing, model training, and parameter tuning. We utilized the functions provided by PyCaret to train our models.

However, we encountered several issues. First, when we tuned the model and examined the pipeline, we found that its complexity could lead to overfitting. By evaluating other metrics, including the F1 score and the decision boundary, we confirmed our suspicion of overfitting. Additionally, the results from PyCaret did not align with those obtained from our manually coded model.

After careful consideration, we decided to base our model selection on the results from our own code, using PyCaret's results only to narrow down the models for training. To be more time-efficient at the start of the project, we initially trained and compared different models without cross-validation, focusing solely on preprocessing.

Here are the models we used and the results for each:

In conclusion, the different models did not significantly alter the F1 score. Among the models we tested—Decision Tree, Gradient Boosting Classifier, and CatBoost—those with relatively high F1 scores and other metrics stood out.

We ultimately chose CatBoost as our baseline model because it handles categorical data particularly well.

**<Finalized Model>**

To sum up, We did all of the preprocessing mentioned above and tried to solve the target imbalance problem by undersampling using NCR. In addition, we trained the catboost model by finding the appropriate parameters through grid search. We introduced k-fold cross-validation because we believed that it would improve the general performance of the model on the test set.

And this is the result.

There were a few aspects of this project that we found somewhat disappointing. First, we wanted to try LOOCV, a type of cross-validation but it wasn’t possible due to repeated runtime initialization, so we expect performance to improve once the computation issue is resolved.

since the Colab Notebook has a structure that shuts down if the user does not make any movements for 90 minutes, we improved the inconvenience by entering the JavaScript code in the console of the devtool so that the code continues to work after a certain period of time. We attempted several runs, but none were successful.

To conclude, when our model was evaluated on the test set, it achieved an F1 score of 0.648. Additionally, the model reached an accuracy of 0.894 and a ROC AUC of 0.857 on the training set. However, it was observed that the F1 score dropped to 0.5007 when applied to the training set.

This concludes our presentation. Thank you.