Applied Machine Learning Final Project : Home Credit Default Risk

Group 36

Yashvanth Guntupalli (yakguntu@iu.edu)

Srilekha Malraju (msrilekh@iu.edu)

Revanth Sai Chowdary Rayala (rerayala@iu.edu)

Vinay Chandra Makineni (vimakin@iu.edu)

Contents

Four P’s

- Past
  - The HCDR projects predicts whether the borrowers are defaulters or not based on various features
  - In the first phase, we performed EDA and performed baseline model with logistic regression
  - We got satisfactory results but with less AUC score
  - To improve the results, we tried balancing the data but it wasn’t very successful

- Present
  - We performed feature engineering, hyperparameter tuning, and modeling pipelines.
  - To find the best model, we implemented models using Logistic Regression, Decision Tree, Random Forest, XGBoost, Lasso Regression using Hyperparamter tuning(GridSearchCV)
  - We used a couple of metrics for each model to evaluate the models

Cont'd

- Planned
  - We will implement a Deep Learning Models
  - We will use models like Recurrent Neural Networks and Multilayer Perceptron using PyTorch
  - We will then submit our results on Kaggle.
- Problems:
  - We could not improve the accuracy and AUC at the same time
  - Our goal is to improve the both at the same time using Neural Networks

Feature Selection

- We added some new features to the dataset which we found to be meaningful which are as follows:
- 1) Annuity to Salary Ratio
- 2) Credit to Goods Ratio
- 3) Goods to Income Ratio
- 4) Credit to Income Ratio
- 5) Employed to Birth Ratio
- 6) Multiplication of EXT_SOURCE
- 7) New phone to birth ratio
- 8) Income Per Person
- 9) Annuity to Credit Ratio

Hyperparameter Tuning

- We tune our model to find the optimal parameters. We use GridSeachCV for achieving this. We implement multiple models such as Logistic Regression, Decision Trees,Random Forest, SVM, Lasso Regression, XGBoost.
- We tuned the hyperparameters with the help of "GridSearchCV"

Results and Discussion

- Logistic Regression :
  - We implemented Logistic Regression with Hyperparameter Tuning in the above cell, as we can see, We got an accuracy of 92% which is an improvement from Phase - 1 where we got an accuracy of 91.9%. The AUC score didn't improve too much. The improvement in the accuracy isn't very considerable, but it's still an improvement.
- Random Forest
  - We implemented Random Forest with GridSearchCV in the above cell, as we can see, We got an accuracy of 92%. The AUC score didn't improve too much for Random Forest as well. We haven't tested Random Forest with just the baseline model hence we can't comment on how well the model performed better with hyperparameter tuning.
- Decision Trees
  - We implemented Decision Trees with Hyperparameter Tuning in the above cell, as we can see, We got a very poor accuracy but a decent AUC score of 0.6. Hence, we can conclude that Decision Trees doesn't work well with the dataset.
- XGBoost
  - We tried using an ensemble method, XGBoost, with GridSearchCV to check how it's performing on our dataset. We got an accuracy of 64.98% with an AUC score of 0.5. Hence ,we can see that it's performance on our dataset isn't too great compared to Logistic Regression and Random Forest.
- Lasso Regression
  - Finally, we tried Lasso Regression using GridSearch and the model performed extremely poor, but with a decent AUC score. However, there's no point of a good AUC score with an extremely poor.

Conclusion and Next Steps

- Phase 2 :
  - We performed feature engineering, hyperparameter tuning, and modeling pipelines.
  - To find the best model, we implemented models using Logistic Regression, Decision Tree, Random Forest, XGBoost, Lasso Regression using Hyperparameter tuning(GridSearchCV)
  - We used a couple of metrics for each model to evaluate the models

- Phase 3 :
  - We will implement a Deep Learning Models
  - We will use models like Recurrent Neural Networks and Multilayer Perceptron using PyTorch
  - We will then submit our results on Kaggle.