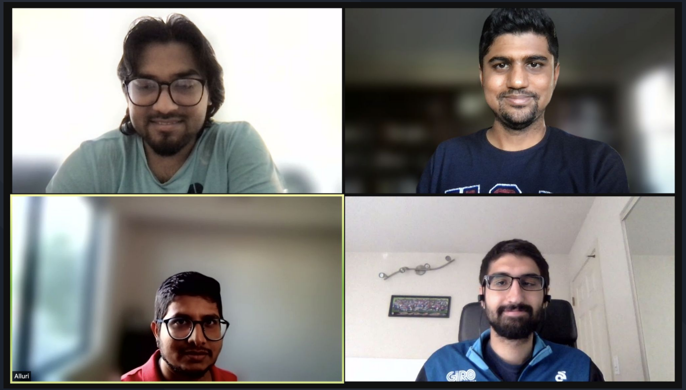
Predicting Home Credit Loan Default



Group 3 - Navy Seals
Pritam Vanmore pvanmore@iu.edu
Gautam Ashok gaashok@iu.edu
Sudheer Alluri naalluri@iu.edu
Raman Kahlon rakahlon@iu.edu

Outline

1. Background
2. Additional EDA
3. Pipeline Modeling
4. Results
5. Conclusion & Next Steps

Background

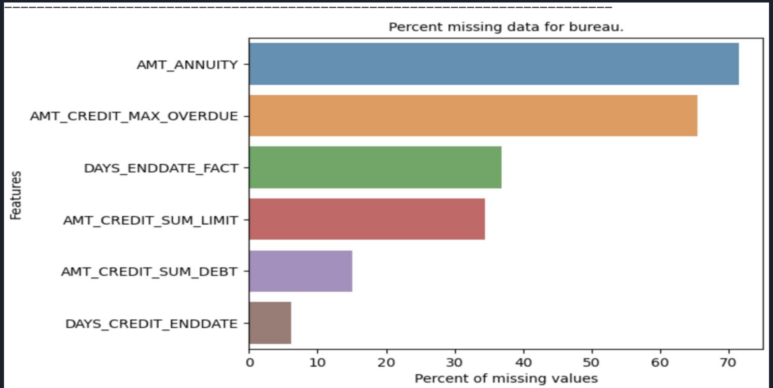
High-Level Goal: Build a machine learning model which can accurately predict whether the customer defaults on a loan or not.

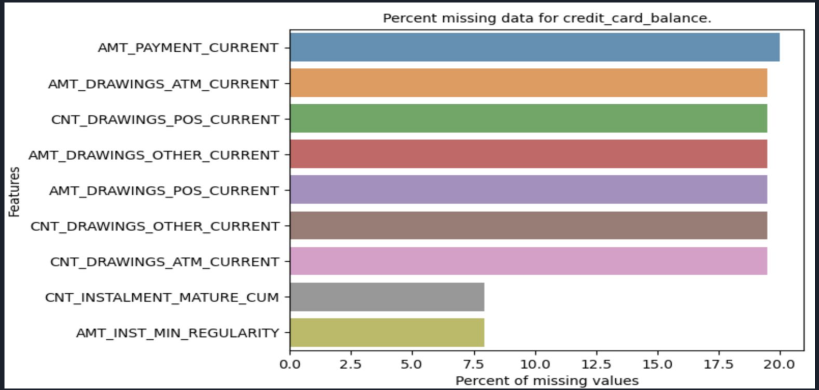
- **Phase 2 Objective:** compare six classification models' performance after hyperparameter tuning (along with additional feature engineering from Bureau dataset)

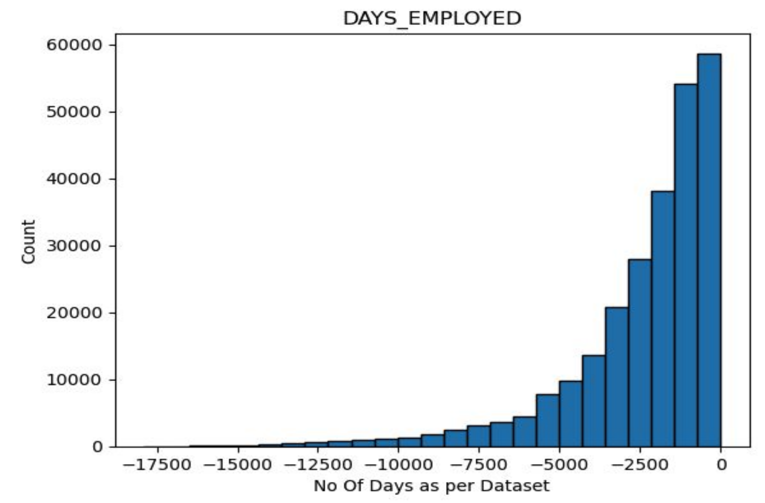
Completed Activities:

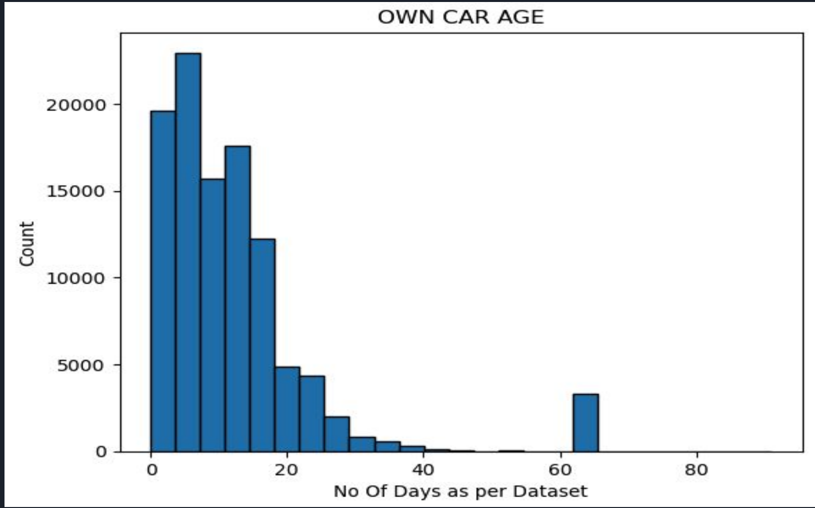
- Phase 0: Performed EDA on the datasets and identified algorithms and evaluation metrics
- Phase 1: Implemented baseline model with Logistic Regression and fine-tuned parameters

Additional EDA

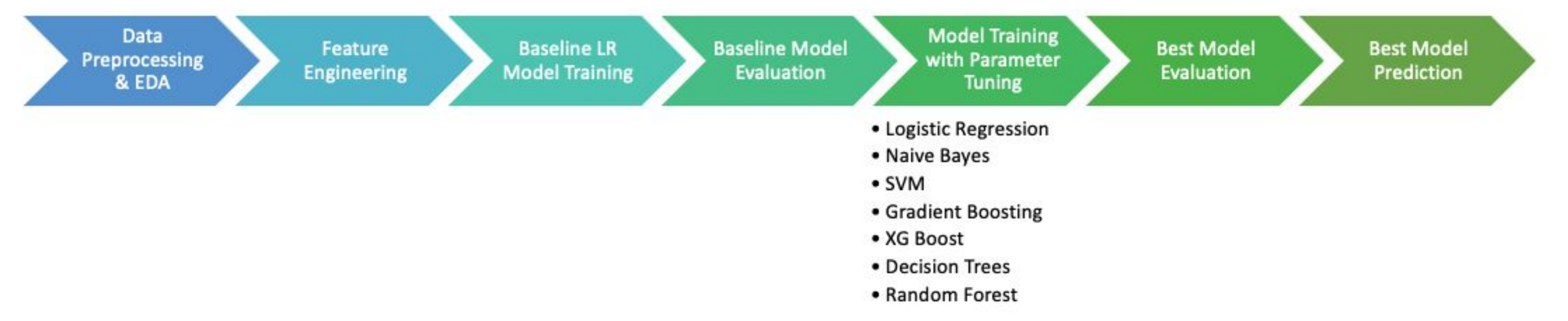








Pipeline Modeling

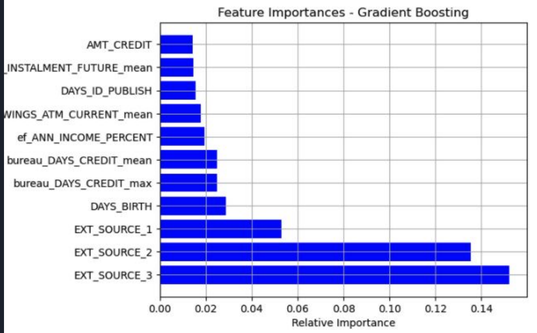


1. Pre-processed data and set up pipeline with highly correlated features
   a. **Calculated min, max, and mean for all numerical features**
   b. **Engineered new features (ratios) to better assess loan affordability**
   c. **Tuned hyperparameters for regularization and limit tree depth to avoid overfitting**
2. Implemented GridSearch to tune hyper-parameters with other classifier models
3. Performed GridSearch on each classifier and evaluated performance via accuracy score, AUC score, Log Loss, RMSE, MAE, p-value, confusion matrix, and ROC curve plots

Results & Discussion

Final experiment results:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train Precision	Valid Precision	Test Precision	Train Recall	Valid Recall	Test Recall	Train Log Loss	Valid Log Loss	Test Log Loss	P Score
0	Baseline_91_features	0.9188	0.9191	0.9182	0.7615	0.7504	0.7567	0.4976	0.5600	0.3895	0.0206	0.0160	0.0148	2.7957	2.8187	2.8269	0.0000
1	Logistic Regression	0.9191	0.9188	0.9185	0.7589	0.7490	0.7561	0.5592	0.5000	0.4286	0.0172	0.0114	0.0132	2.7889	2.8107	2.8157	0.0477
2	Naive Bayes	0.1960	0.1929	0.1944	0.6583	0.6522	0.6489	0.0867	0.0861	0.0862	0.9340	0.9302	0.9303	27.8160	27.8216	27.8264	0.0000
3	Gradient Boosting	0.9235	0.9197	0.9185	0.8244	0.7582	0.7580	0.8198	0.5926	0.4636	0.0735	0.0366	0.0280	2.6279	2.8179	2.8157	0.0000
4	XGBoost	0.9230	0.9194	0.9190	0.8563	0.7592	0.7619	0.9132	0.6000	0.5200	0.0574	0.0240	0.0208	2.6005	2.8024	2.7989	0.0000
5	DecisionTrees	0.9188	0.9188	0.9188	0.7059	0.6966	0.6896	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2.7960	2.8183	2.8034	0.8562
6	RandomForest	0.9193	0.9188	0.9188	0.8884	0.7454	0.7502	1.0000	0.0000	0.0000	0.0057	0.0000	0.0000	2.7792	2.8111	2.8034	0.0001



- Gradient Boosting and XGBoost - statistically significant improvement over baseline
  - Naive Bayes was worst performing model
  - No statistical significance with Decision Tree; Random Forest same performance as baseline
- Most predictive features: external source scores, days_birth, and days_credit
- Phase 2 Kaggle score (0.74779) vs Phase 1 (0.74306)

Name	Submitted	Wait time	Execution time	Score
submission_P2.csv	38 minutes ago	1 seconds	1 seconds	0.74779

Complete

Conclusions & Next Steps

- Refined baseline model, engineered new features, and test against other classification models
- XGBoost and Gradient Boosting were best performing models

Project Challenges:

- Determine appropriate features to engineer and deal with missing data.
- **Memory issues** - needed to increase to 7 CPU's and 11GB memory in Docker to run the model. Running SVM crashed our kernels; also XGBoost kept running and hence couldn't submit for Kaggle.

Next Steps:

- For Phase 3, run PyTorch model in IU Red.
- Attempt SVM in new environment
- Examine feature engineering process