Loan Predictions

Osher Boudara, Allison Paxton - Team 3

Outline

Abstract

- In this project, we will implement XGBoost for Loan Predictions as specified by a UCLA article.
    - We will improve on this by using two different data balancing techniques: K-Fold Cross Validation and Stratified K-Fold Cross Validation.
- We will also implement a more standard version of XGBoost with the new data balancing techniques and observe performance.
- Random Forest Classifier and Logistic Regression will also be implemented with the new data balancing techniques and compare performance.

Introduction

Decision

- We chose to implement a loan prediction algorithm using XGBoost over license plate image recognition using Long Short-Term Memory (LSTM) because:
  - Loan prediction algorithm had a much better and more efficient performance.
  - We misunderstood what the LSTM model could accomplish and it did not perform well with license plate recognition.

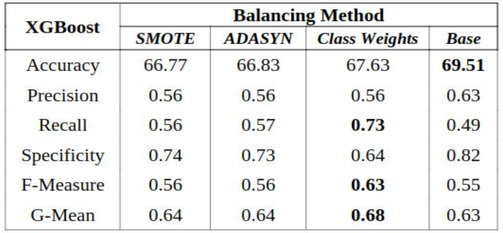
Previous Work

- Both logistic regression and random forests are common ways of predicting loans.[12]
- Many papers that implemented loan prediction algorithms needed to use balancing techniques to balance their data.[1] [11] [13]
- Papers that used XGBoost for loan predictions or other implementations used intricate parameters to set up their model.[1] [11] [13]

UCLA Article

Reference [21]

- We found a paper from UCLA that tackles loan predictions using XGBoost. But, there was an imbalance in the distribution of the target features.[1]
- Further, they used different data balancing techniques to assess their performance of XGBoost.
- UCLA's XGBoost algorithm performed the overall best using the class weights data balancing method as demonstrated in the table below:



XGBoost	Balancing Method			
	SMOTE	*ADASYN*	*Class Weights*	*Base*
Accuracy	66.77	66.83	67.63	**69.51**
Precision	0.56	0.56	0.56	0.63
Recall	0.56	0.57	**0.73**	0.49
Specificity	0.74	0.73	0.64	0.82
F-Measure	0.56	0.56	**0.63**	0.55
G-Mean	0.64	0.64	**0.68**	0.63

Reference [2]

Plan

- Predict whether someone will be approved for a loan, or not.
- To predict loan approval, we will implement two variations of XGBoost, Logistic Regression and a Random Forest Classifier.
- Use two data balancing methods (Stratified K-Fold and K-Fold Cross Validations) on each of the algorithms to improve performance and compare against UCLA's data balancing methods.
- Compare results of all algorithms.

Reference [18]

Hypothesis

- We expect that our implementation of UCLA’s XGBoost using K-Fold Cross Validation and Stratified K-Fold Cross Validation will perform better than that of the article.
- Further, we will implement other common algorithms for Loan Predictions such as standard XGBoost, Random Forest and Logistic Regression to determine the best performing algorithm.
- The purpose of doing this is to improve upon or find the best algorithm for Loan Predictions. The UCLA article stated it is XGBoost with class weights but we felt that the metrics were too low to be the best performing algorithm.

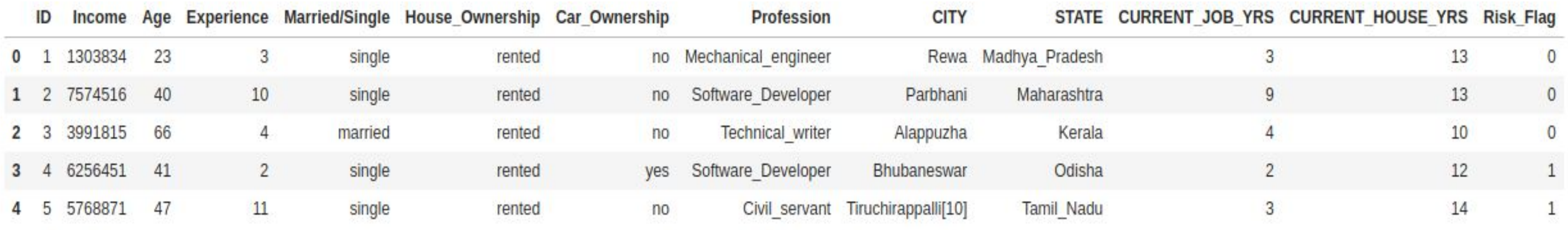
Dataset

Dataset

- We chose a much larger dataset with more descriptive features than the UCLA article in order to improve results.
- Source of dataset came from Kaggle with three separate files. [10]
    - Three CSV files - training dataset, test dataset with no target feature, and the target feature for test dataset.
- We combined the two test data CSV files into one so that the descriptive features and the target feature are in the same file.
- There were 252,000 samples in the training dataset and 28,000 samples in the test dataset.

Dataset Features

- There are 11 descriptive features:
  - Income, exact age, experience, current job years, and current house year are numerical data.
  - Marriage status, house ownership, car ownership, profession, city, and state are categorical data.
- Risk flag is a binary target feature, where 0 means a loan was not approved and 1 means a loan was approved. Below are the first 5 rows of the dataset we are using:



	ID	Income	Age	Experience	Married/Single	House_Ownership	Car_Ownership	Profession	CITY	STATE	CURRENT_JOB_YRS	CURRENT_HOUSE_YRS	Risk_Flag
0	1	1303834	23	3	single	rented	no	Mechanical_engineer	Rewa	Madhya_Pradesh	3	13	0
1	2	7574516	40	10	single	rented	no	Software_Developer	Parbhani	Maharashtra	9	13	0
2	3	3991815	66	4	married	rented	no	Technical_writer	Alappuzha	Kerala	4	10	0
3	4	6256451	41	2	single	rented	yes	Software_Developer	Bhubaneswar	Odisha	2	12	1
4	5	5768871	47	11	single	rented	no	Civil_servant	Tiruchirappalli[10]	Tamil_Nadu	3	14	1

Reference [10]

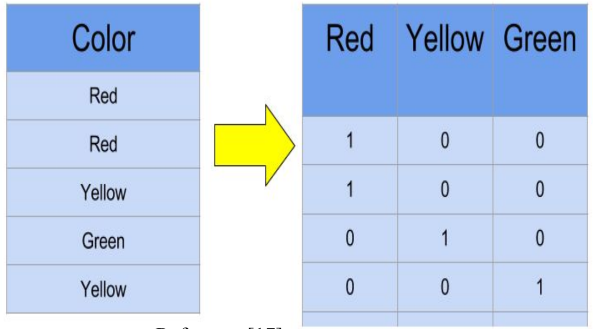
Methodology

Data Preparations

- Firstly, we combined the training and test data into one huge dataset. This is done because stratified K-Fold and regular K-fold cross validations will split the data into training and test sets for us.
- Further, we had to check if there were any null values and there were not any.
- We also dropped the ID column because we will not be learning from it.

Data Preprocessing: Scaling and Encoding

- We scaled all our numeric data using the Min Max Scaler.
  - The Min Max Scaler will normalize values between the range 0 and 1. [8][9]
- The next step in our data preprocessing is to handle our categorical data with One Hot Encoding. [16]
  - One Hot Encoding will change our categorical data in a binary vector representation.



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Reference [17]

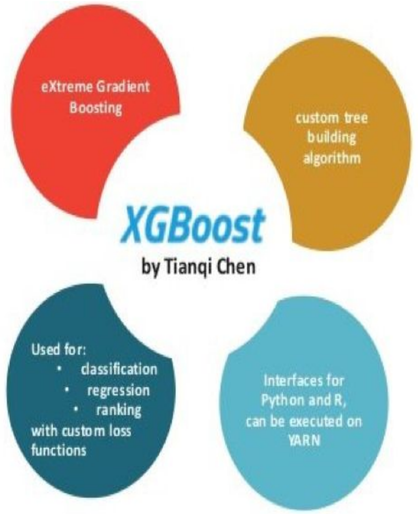
$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

Data Balancing Techniques

- Cross-validation is a statistical method that uses resampling techniques to evaluate a model. [4]
- It will partition the data into multiple groups and run the model on each group and then it will store the evaluation metrics. The model is relearned for each partitioned group.[7]
- K-Fold cross validation uses random sampling and Stratified K-Fold Cross Validation uses stratified sampling. [4][6][14]
- Both algorithms will be ran 3 times where K=3, K=5 and K=10 because researchers have stated that these are the optimal values to use. [5]

What is XGBoost?

- Extreme Gradient Boosting or XGBoost is a type of boosting algorithm created by Tianqi Chen which improves upon the Gradient Boosting Algorithm. [2]
- Boosting algorithms aim to build stronger models using an ensemble of weak learners.
- In our case, since this is a classification problem, XGBoost is an ensemble of decision trees.
- XGBoost tends to work much quicker than other algorithms because it utilizes newer technologies such as parallel computing. [1]


eXtreme Gradient Boosting
custom tree building algorithm
XGBoost
by Tianqi Chen
Used for:
• classification
• regression
• ranking
with custom loss functions
Interfaces for Python and R, can be executed on YARN

Reference [15]

XGBoost Implementations

- We implemented XGBoost in two different ways:
  - The first implementation was the way the UCLA paper implemented theirs. They used various parameters such as n_estimators = 300, max_depth = 3 and a learning rate of 0.5. [1]
  - The second implementation was a more standard implementation of XGBoost. The above parameters are optional so we did not use them in this implementation. [3]

UCLA Article Limitations Fix

- Both of our XGBoost algorithms (Standard and UCLA's) used the parameter scale_pos_weight because the actual distribution of target features was imbalanced in the dataset we chose. [3]
    - This is one of the limitations of the UCLA paper that we had to account for. [1]
    - We had 7.1 times more negative results than positive results in our target feature so setting that parameter equal to 7.1 fixes this issue.

Logistic Regression

- Logistic regression was implemented to compare with XGBoost, since it is a common model used to predict loans. [12][22]
- Logistic regression is a classification algorithm where a logistic function is used to model probabilities of possible outcomes. [24]
- The parameters used were class_weight = "balanced" and max_iter = 400.

Random Forest Classifier

- This algorithm was implemented because it is also commonly used in loan predictions. [12][22]
- The random forest classifier algorithm is a classification algorithm which uses an ensemble of decision trees to make predictions and utilizes the averaging method to improve its accuracy. [23]
- This algorithm was implemented in its most standard form, thus no additional parameters were needed.

Summary of Tools/Algorithms/Formulas

- Jupyter Notebooks
  - Integrated Development Environment
- Python
- Python Libraries:
  - Pillow
    - Load Images
  - Pandas
    - Data Manipulation
  - Sci-Kit Learn
    - Machine Learning related functions
  - Re
    - Regex library
  - XGBoost
    - XGBoost implementation library
  - Numpy
    - Array/Matrix Operations

- XGBoost Classifier
  - 2 variations:
    - UCLA Paper
    - Standard XGBoost
- Random Forest Classifier
- Logistic Regression
- Stratified K-Fold Cross Validation
- K-Fold Cross Validation
- MinMaxScaler
- One Hot Encoding

Reference [19]

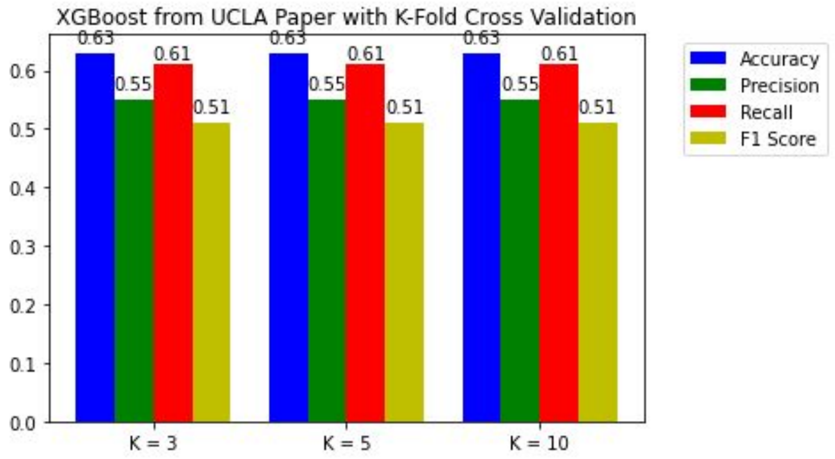
Reference [20]

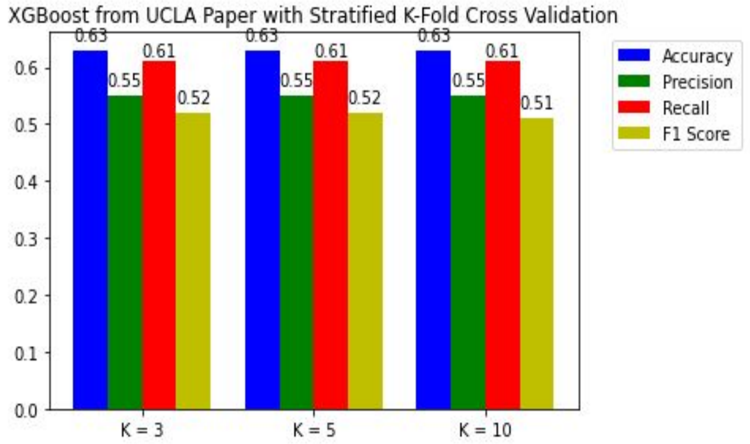
Results

Note: Precision, Recall and F1 measures have been averaged out for each K trial.

UCLA XGBoost Algorithm: K-Fold vs Stratified

- These are the results from our implementation of UCLA's XGBoost algorithm using K-Fold Cross Validation and Stratified K-Fold Cross Validation:

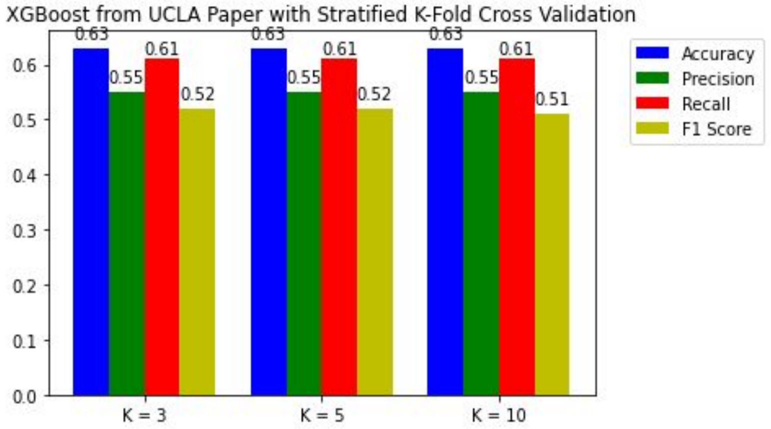


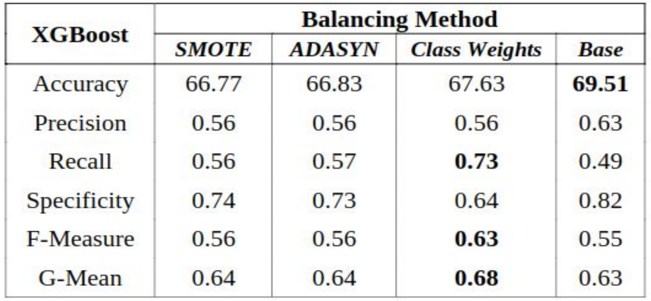


- As can be observed from the graph above, K-Fold Cross Validation and Stratified K-Fold Cross Validation are about the same in terms of performance (the variation of the metrics for the different trials is off by at most 0.01 percent). We will use the algorithm that performed slightly better from now when comparing algorithms.

UCLA XGBoost Algorithm: Article vs Ours

- To compare performance, we chose Stratified K-Fold Cross Validation which performed minutely better than K-Fold Cross Validation.





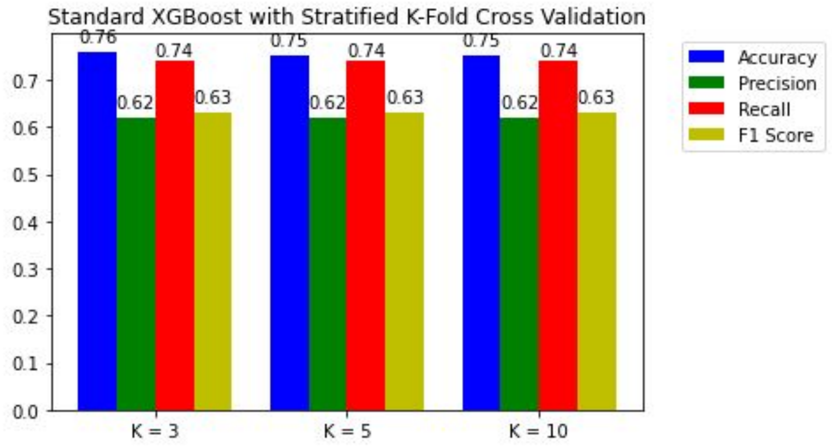
XGBoost	Balancing Method			
	SMOTE	*ADASYN*	*Class Weights*	*Base*
Accuracy	66.77	66.83	67.63	**69.51**
Precision	0.56	0.56	0.56	0.63
Recall	0.56	0.57	**0.73**	0.49
Specificity	0.74	0.73	0.64	0.82
F-Measure	0.56	0.56	**0.63**	0.55
G-Mean	0.64	0.64	**0.68**	0.63

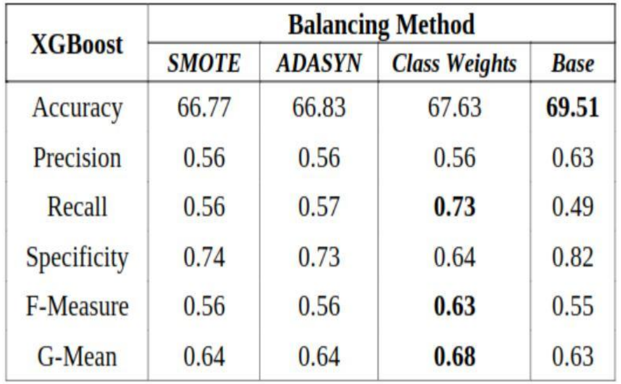
Reference [2]

- Overall, UCLA's with class weights stills performs the best. Our algorithm did better than base, SMOTE and ADASYN in terms of recall but otherwise, it did not perform as well.

Standard XGBoost Performance

- Because our implementation of UCLA's algorithm using cross validation did not perform as well, we decide to take a more standard approach and omit the parameters used by UCLA in their implementation of XGBoost while still using Cross Validation. We will compare this version against class weights which is currently the best performing algorithm.





XGBoost	Balancing Method			
	SMOTE	*ADASYN*	*Class Weights*	*Base*
Accuracy	66.77	66.83	67.63	**69.51**
Precision	0.56	0.56	0.56	0.63
Recall	0.56	0.57	**0.73**	0.49
Specificity	0.74	0.73	0.64	0.82
F-Measure	0.56	0.56	**0.63**	0.55
G-Mean	0.64	0.64	**0.68**	0.63

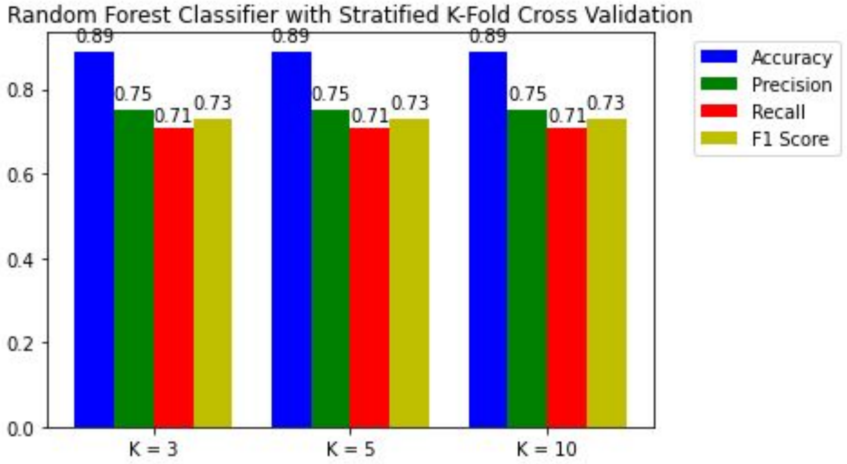
Reference [2]

- As can be observed, Standard XGBoost with Cross Validation performed much better in terms of accuracy and precision than XGBoost with class weights. Recall is a bit higher in class weights but F1-measure is slightly better in the standard XGBoost algorithm.

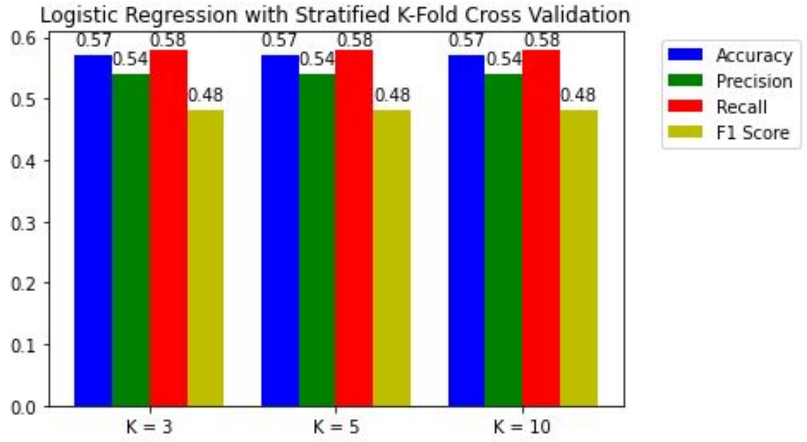
Performance Comparisons

- Because Standard XGBoost using cross validations performed better than UCLA’s XGBoost with class weights, we decided to implement other commonly used with Loan Predictions: Random Forest and Logistic Regression. Below we will display best and worst performing algorithms.

Best Performing



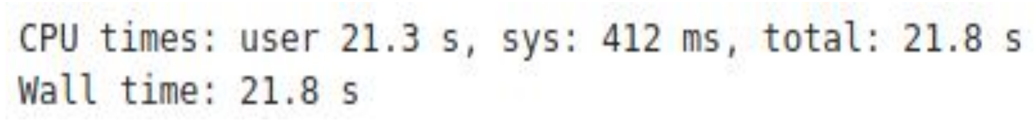
Worst Performing



- As can be observed, Random Forest performs exceedingly better than all the other algorithms, XGBoost implementations included. Logistic Regression performs worse than all the other implemented algorithms.

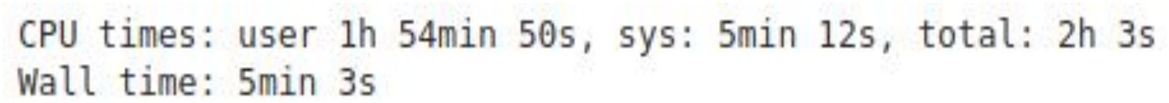
Runtime Comparisons

- To further compare the algorithms, we also compared each of their runtimes to find the best and worst runtimes. We ran all algorithms using an AMD Ryzen 5900x CPU.
  - The quickest: Logistic Regression using stratified 3-fold.



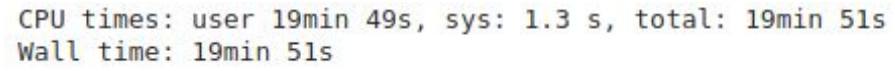
```
CPU times: user 21.3 s, sys: 412 ms, total: 21.8 s
Wall time: 21.8 s
```

  - The slowest: UCLA XGBoost using 10-fold.



```
CPU times: user 1h 54min 50s, sys: 5min 12s, total: 2h 3s
Wall time: 5min 3s
```

Conclusion

- Because the class weights UCLA algorithm performs better than the UCLA algorithm with cross validation, we will reject our first part of the hypothesis.
- Further, we wanted to determine the best performing algorithm. Although standard XGBoost performed the best out of the XGBoost implementations, Random Forest Classifier using Stratified K-Fold Cross Validation is the best algorithm.
- Random Forest Classifier evaluation metrics:
  - Accuracy: about 89%, Recall: about 71%, Precision: about 75%, F1-Measure: about 73%
  - Runtime: 

```
CPU times: user 19min 49s, sys: 1.3 s, total: 19min 51s
Wall time: 19min 51s
```

Contributions

New Perspectives

- From the previous works that needed to balance their data, none used stratified K-fold cross validation or K-fold cross validation.
  - We observed very slight variation between implementations of K-Fold and Stratified K-Fold.
- Since many works used elaborate parameters for their XGBoost model, we implemented a simpler XGBoost algorithm alongside the UCLA one to compare them.
- Because the UCLA article had an issue of imbalanced data, we used the scale_pos_weights parameter in our XGBoost algorithm. Further, we used a dataset with more features/rows to obtain better results.
- We found that Random Forest Classifier with Stratified K-Fold Cross Validation performs the best, even though the UCLA article concluded that XGBoost with class weights performed the best.

Future Works

- Fine tune Standard XGBoost parameters in order to try to obtain better performance than Random Forest Classifier.
- We did not implement any of the balancing techniques from the UCLA paper ourselves so implementing the standard XGBoost with Class Weights would be beneficial to assess.
- Since Logistic Regression performed the worst out of all the algorithms, we will try to obtain better performance.
- Implement another commonly used Loan Prediction algorithm and compare it with the two best algorithms we found - Standard XGBoost and Random Forest Classifier.

Division of Work

Osher Boudara

- XGBoost source code implementation and visualization
- Random Forest source code implementation and visualization
- LSTM source code debugging
- Found dataset and performed data preparation/preprocessing
- Wrote introduction, result, and methodology sections

Allison Paxton

- XGBoost source code debugging
- LSTM source code implementation
- Logistic Regression source code implementation and visualization
- Wrote dataset, contributions, and reference list sections

Reference List

[1] "Classification Example with XGBClassifier in Python," *DataTechNotes*, Jul. 4, 2019. [Online]. Available: link. [Accessed: Oct. 3, 2021]

[2] L. Zhu, "Predictive Modelling of Loan Defaults," *UCLA Electronic Theses and Dissertations*, 2019. [Online]. Available: link. [Accessed: Sept. 21, 2021].

[3] “Python API reference,” *Python API Reference - xgboost 1.6.0-dev documentation*, 2021. [Online]. Available: link. [Accessed: Oct. 4, 2021].

[4] J. Bownlee, ”A Gentle Introduction to k-fold Cross-Validation,” *Machine Learning Mastery*, May 23, 2018. [Online]. Available: link. [Accessed: Oct. 9, 2021].

[5] J. Bownlee, “How to Configure k-Fold Cross-Validation,” *Machine Learning Mastery*, Jul. 21, 2020. [Online]. Available: link. [Accessed: Oct. 9, 2021].

[6] “sklearn.model_selection.StratifiedKFold,” *scikit-learn.org*. [Online]. Available: link. [Accessed: Oct. 13, 2021].

[7] “Stratified K Fold Cross Validation,” *GeeksforGeeks*, May 29, 2021. [Online]. Available: link. [Accessed: Oct. 10, 2021].

[8] “sklearn.preprocessing.MinMaxScaler,” *scikit-learn.org*. [Online]. Available: link. [Accessed: Oct. 12, 2021].

[9] “StandardScaler, MinMaxScaler and RobustScaler techniques – ML,” *GeeksforGeeks,* Aug. 20, 2021. [Online]. Available: link. [Accessed: Oct. 15, 2021].

[10] A. Vayani, *Loan Predictor- EDA, GML, Voting Classifiers*, Kaggle, Aug. 2021. [Online]. Available: link.[Accessed: Oct. 10, 2021].

[11] X. Yu, “Machine learning application in online lending risk prediction,” *arXiv.org*, Jul. 17, 2021. [Online]. Available: link. [Accessed: Sept. 23, 2021].

[12]M. Medaan, A. Kumas, C. Keshri, R, Jain and P. Nagrath, “Loan default prediction using decision trees and random forest: A comparative study,” *IOP Conference Series: Materials Science and Engineering*, 2020. [Online]. Available: link. [Accessed: Sept. 22, 2021].

[13]Y. Lao, F. Qi, J. Zhou and X. Fang, “A Prediction Method Based on Extreme Gradient Boosting Tree Model and its Application,” *Journal of Physics: Conference Series*, 2021. [Online]. Available: link. [Accessed: Sept. 23, 2021].

Reference List

[14] S. Nickolas, “How stratified random sampling works,” *Investopedia*, May 19, 2021. [Online]. Available: link. [Accessed: Sept. 30, 2021].

[15] I. Reinstein, “XGBoost, a top machine learning method on Kaggle, explained,” *KDnuggets*. [Online]. Available: link. [Accessed: Oct. 2, 2021].

[16] J. Brownlee, “Why one-hot encode data in machine learning?,” *Machine Learning Mastery*, Jun 30, 2020. [Online]. Available: link. [Accessed: Oct 2, 2021].

[17] Dansbecker, “Using categorical data with one hot encoding,” *Kaggle*, Jan. 22, 2018. [Online]. Available: link. [Accessed: Oct. 4, 2021].

[18] A. Crowe, “A simple guide to 5 popular types of loans,” *BadCredit.org*, Jun. 23, 2020. [Online]. Available: link. [Accessed: Oct. 23, 2021].

[19] P. Pandey, “Jupyter Lab: Evolution of the jupyter notebook,” *Medium*, Jul. 3, 2019. [Online]. Available: link. [Accessed: Oct. 19, 2021].

[20] “The python logo,” *Python.org*. [Online]. Available: link. [Accessed: Oct. 21, 2021].

[21]“Historical marks,” *Brand Guidelines | Identity | Logos and Marks | Historical Logos and Marks*. [Online]. Available: link. [Accessed: Oct. 20, 2021].

[22] R. Mukherjee, “Loan prediction Project Termpaper,” *Deepnote*, 16-Jun-2021. [Online]. Available: link. [Accessed: 04-Nov-2021].

[23] “sklearn.ensemble.RandomForestClassifier,” *scikit-learn.org*. [Online]. Available: link. [Accessed: Nov. 5, 2021].

[24] “sklearn.linear_model.LogisticRegression,” *scikit-learn.org*. [Online]. Available: link. [Accessed: Nov. 3, 2021].

Q&A