A

Project Proposal

on

Home Credit Default Risk (HCDR)

Developed at

Luddy School of Informatics, Computing, and Engineering

Developed by

Aarushi Dua (aarudua@iu.edu)

Sai Teja Burla (saburla@iu.edu)

Lakshay Madaan (lmadaan@iu.edu)

Shyam Makwana (smakwana@iu.edu)

Guided By

James Shanahan (jshanah@indiana.edu)

700 N Woodlawn Ave, Bloomington, IN 47408

Spring - 2023

Group Name: FP_Group22

Team Members:

1. Aarushi Dua (aarudua@iu.edu)
2. Sai Teja Burla (saburla@iu.edu)
3. Lakshay Madaan (lmadaan@iu.edu)
4. Shyam Makwana (smakwana@iu.edu)

Team Photo:

Abstract

The objective of the Home Credit Default Risk project is to assess a client's ability to repay a loan based on a collection of features present in the dataset.

In phase 1, we identified the problem as a classification task and selected suitable ML models. Phase 2 involved EDA and creating baseline models with four different algorithms. In Phase 3, we conducted additional feature engineering to identify the most effective features and used GridSearch to further refine the accuracy of our models. We observed that the XGBoost produced a Kaggle score of 0.736.

During this phase, our aim was to implement Multilayer Perceptron architectures to analyze our data and evaluate the performance based on the binary cross-entropy loss function. We utilized the ReLU activation function for the hidden layer and sigmoid for the output layer with the Adam optimizer and performed 50 epochs using 3 hidden layers and 1 output feature. Additionally, we utilized TensorBoard to visualize our accuracy and loss. As a result, we achieved a test ROC of 0.76, with a Kaggle submission public score of 0.728.

Data Description

1. application_{train|test}.csv

The primary table consists of the necessary information regarding home credit loan applications.

This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET). Static data for all applications. One row represents one loan in our data sample.

```
application_train: shape is (307511, 122)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
None
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY
0	100002	1	Cash loans	M	N	Y
1	100003	0	Cash loans	F	N	N
2	100004	0	Revolving loans	M	Y	Y
3	100006	0	Cash loans	F	N	Y
4	100007	0	Cash loans	M	N	Y

5 rows × 122 columns

```
application_test: shape is (48744, 121)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN
0	100001	Cash loans	F	N	Y	0
1	100005	Cash loans	M	N	Y	0
2	100013	Cash loans	M	Y	Y	0
3	100028	Cash loans	F	N	Y	2
4	100038	Cash loans	M	Y	N	1

5 rows × 121 columns

2. bureau.csv

This is a record of how well the client has paid back loans and borrowed money in the past, as reported by other banks or lenders. All client's previous credits provided by other financial institutions were reported to the Credit Bureau (for clients who have a loan in our sample). For every loan in our sample, there are as many rows as the number of credits the client had in the Credit Bureau before the application date.

```
bureau: shape is (1716428, 17)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_CURR              int64
 1   SK_ID_BUREAU            int64
 2   CREDIT_ACTIVE           object
 3   CREDIT_CURRENCY         object
 4   DAYS_CREDIT             int64
 5   CREDIT_DAY_OVERDUE      int64
 6   DAYS_CREDIT_ENDDATE     float64
 7   DAYS_ENDDATE_FACT       float64
 8   AMT_CREDIT_MAX_OVERDUE  float64
 9   CNT_CREDIT_PROLONG      int64
 10  AMT_CREDIT_SUM          float64
 11  AMT_CREDIT_SUM_DEBT     float64
 12  AMT_CREDIT_SUM_LIMIT    float64
 13  AMT_CREDIT_SUM_OVERDUE  float64
 14  CREDIT_TYPE             object
 15  DAYS_CREDIT_UPDATE      int64
 16  AMT_ANNUITY             float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
None
```

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVERDUE
0	215354	5714462	Closed	currency 1	-497	0
1	215354	5714463	Active	currency 1	-208	0
2	215354	5714464	Active	currency 1	-203	0
3	215354	5714465	Active	currency 1	-203	0
4	215354	5714466	Active	currency 1	-629	0

3. bureau_balance.csv

This has information about the amounts you owed and paid back each month in the past, which is recorded by a company that keeps track of people's credit histories. This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.

```
bureau_balance: shape is (27299925, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Dtype
---  ------          -----
 0   SK_ID_BUREAU    int64
 1   MONTHS_BALANCE  int64
 2   STATUS          object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
None
```

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

4. POS_CASH_balance.csv

This table shows how much money the client owes to Home Credit every month from their previous purchases made using credit or cash loans. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.

```
POS_CASH_balance: shape is (10001358, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                 Dtype
---  ------                 -----
 0   SK_ID_PREV             int64
 1   SK_ID_CURR             int64
 2   MONTHS_BALANCE         int64
 3   CNT_INSTALMENT         float64
 4   CNT_INSTALMENT_FUTURE  float64
 5   NAME_CONTRACT_STATUS   object
 6   SK_DPD                 int64
 7   SK_DPD_DEF             int64
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTURE
0	1803195	182943	-31	48.0	45.0
1	1715348	367990	-33	36.0	35.0
2	1784872	397406	-32	12.0	9.0
3	1903291	269225	-35	48.0	42.0
4	2341044	334279	-35	36.0	35.0

5. credit_card_balance.csv

This table shows how much money the applicant owed on their Home Credit credit cards in past months. This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credit cards * # of months where we have some history observable for the previous credit card) rows.

```
credit_card_balance: shape is (3840312, 23)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                      Dtype
---  ------                      -----
 0   SK_ID_PREV                  int64
 1   SK_ID_CURR                  int64
 2   MONTHS_BALANCE              int64
 3   AMT_BALANCE                 float64
 4   AMT_CREDIT_LIMIT_ACTUAL     int64
 5   AMT_DRAWINGS_ATM_CURRENT    float64
 6   AMT_DRAWINGS_CURRENT        float64
 7   AMT_DRAWINGS_OTHER_CURRENT  float64
 8   AMT_DRAWINGS_POS_CURRENT    float64
 9   AMT_INST_MIN_REGULARITY     float64
 10  AMT_PAYMENT_CURRENT         float64
 11  AMT_PAYMENT_TOTAL_CURRENT   float64
 12  AMT_RECEIVABLE_PRINCIPAL    float64
 13  AMT_RECIVABLE               float64
 14  AMT_TOTAL_RECEIVABLE        float64
 15  CNT_DRAWINGS_ATM_CURRENT    float64
 16  CNT_DRAWINGS_CURRENT        int64
 17  CNT_DRAWINGS_OTHER_CURRENT  float64
 18  CNT_DRAWINGS_POS_CURRENT    float64
 19  CNT_INSTALMENT_MATURE_CUM   float64
 20  NAME_CONTRACT_STATUS        object
 21  SK_DPD                      int64
 22  SK_DPD_DEF                  int64
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL
0	2562384	378907	-6	56.970	135000
1	2582071	363914	-1	63975.555	45000
2	1740877	371185	-7	31815.225	450000
3	1389973	337855	-4	236572.110	225000
4	1891521	126868	-1	453919.455	450000

5 rows × 23 columns

6. previous_application.csv

All previous applications for Home Credit loans of clients who have loans in our sample. There is one row for each previous application related to loans in our data sample.

```
previous_application: shape is (1670214, 37)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0

5 rows × 37 columns

7. installments_payments.csv

This shows whether or not you've paid back loans you got from Home Credit in the past. There is a) one row for every payment that was made plus b) one row each for missed payment. One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.

```
installments_payments: shape is (13605401, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13605401 entries, 0 to 13605400
Data columns (total 8 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_PREV              int64
 1   SK_ID_CURR              int64
 2   NUM_INSTALMENT_VERSION  float64
 3   NUM_INSTALMENT_NUMBER   int64
 4   DAYS_INSTALMENT         float64
 5   DAYS_ENTRY_PAYMENT      float64
 6   AMT_INSTALMENT          float64
 7   AMT_PAYMENT             float64
dtypes: float64(5), int64(3)
memory usage: 830.4 MB
None
```

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTALMENT
0	1054186	161674	1.0	6	-1180.0
1	1330831	151639	0.0	34	-2156.0
2	2085231	193053	2.0	1	-63.0
3	2452527	199697	1.0	3	-2418.0
4	2714724	167756	1.0	2	-1383.0

Tasks to be Tackled

We handled the following task in this phase:

1. Firstly, we implemented a MultiLayer Perceptron (MLP) model, which is a neural network architecture consisting of an input layer, one or more hidden layers, and an output layer.
2. We then performed experiments using various optimizer functions and activation functions to optimize the performance of our MLP model.
3. Furthermore, we experimented with different numbers of hidden layers and neurons in each layer to find the most appropriate architecture for our dataset.
4. After that, we evaluated the accuracy of each model and identified the highest-performing one. By performing these tasks, we were able to build a robust MLP model that accurately predicted the binary classification of our dataset.

Workflow Diagram

Here's a brief timeline of the project shown below along with the status of each task as green represents its completed, blue ongoing, and orange yet to start.

Name	Start ...	End Date
Problem Understanding	Mar 18, 2023	Mar 19, 2023
Understanding Data	Mar 20, 2023	Mar 21, 2023
Environment Setup	Mar 21, 2023	Mar 21, 2023
Data Preparation	Mar 23, 2023	Mar 29, 2023
Data Visualization and EDA	Mar 30, 2023	Apr 05, 2023
Phase 1: Report Writing	Apr 02, 2023	Apr 03, 2023
Choose Algorithms and Evaluation Met…	Apr 01, 2023	Apr 02, 2023
Phase 1: Report Submission	Apr 04, 2023	Apr 04, 2023
Feature Engineering	Apr 04, 2023	Apr 13, 2023
Pipeline Creation	Apr 04, 2023	Apr 06, 2023
Baseline Model Training	Apr 07, 2023	Apr 08, 2023

Name	Start ...	End Date
Data Visualization and EDA	Mar 30, 2023	Apr 05, 2023
Phase 1: Report Writing	Apr 02, 2023	Apr 03, 2023
Choose Algorithms and Evaluation Met…	Apr 01, 2023	Apr 02, 2023
Phase 1: Report Submission	Apr 04, 2023	Apr 04, 2023
Feature Engineering	Apr 04, 2023	Apr 13, 2023
Pipeline Creation	Apr 04, 2023	Apr 06, 2023
Baseline Model Training	Apr 07, 2023	Apr 08, 2023
Baseline Model Evaluation	Apr 08, 2023	Apr 09, 2023
Hyperparameter Tuning	Apr 09, 2023	Apr 14, 2023
Phase 2: Report writing/ Video creation	Apr 10, 2023	Apr 11, 2023
Phase 2: Report Submission	Apr 11, 2023	Apr 11, 2023

Name	Start ...	End Date
Baseline Model Training	Apr 07, 2023	Apr 08, 2023
Baseline Model Evaluation	Apr 08, 2023	Apr 09, 2023
Hyperparameter Tuning	Apr 09, 2023	Apr 14, 2023
Phase 2: Report writing/ Video creation	Apr 10, 2023	Apr 11, 2023
Phase 2: Report Submission	Apr 11, 2023	Apr 11, 2023
Compare and Visualize Results	Apr 13, 2023	Apr 16, 2023
Phase 3: Report Writing/ Video Creation	Apr 17, 2023	Apr 17, 2023
Phase 3: Report Submission	Apr 18, 2023	Apr 18, 2023
Advance Model(Pytorch)	Apr 19, 2023	Apr 22, 2023
Phase 4: Report Writing/ Video Creation	Apr 23, 2023	Apr 24, 2023
Final Submission	Apr 25, 2023	Apr 25, 2023

Multi-Layer Perceptron (MLP)

- Multi-Layer Perceptron (MLP) is being utilized to perform binary classification analysis. It is composed of an input layer, one or more hidden layers, and an output layer. The data is fed through the input layer, and the hidden layers perform nonlinear transformations to extract essential features, while the output layer generates the final output of the model.
- The data to be processed is received by the input layer, and then it is processed through a sequence of nonlinear transformations in the hidden layers in order to extract important features. Each hidden layer is composed of a set of neurons, where every neuron accepts inputs from the prior layer and generates an output based on an activation function. The output layer generates the ultimate output of the model.
- Here, we are creating a linear layer that takes the number of neurons in the previous layer as input and number of neurons in the current layer as output. This linear layer applies a linear transformation to the input and produces an output tensor by multiplying the input tensor with a weight matrix and adding a bias vector.
- ReLU stands for Rectified Linear Unit, and it is an activation function used in deep learning models. It applies a non-linear function to the output of each neuron in a neural network. The ReLU function returns the input value if it is positive, and returns zero if the input value is negative. In other words, ReLU introduces non-linearity to the network, helping it to learn more complex patterns in the data. Here, we are adding a ReLU activation function after each linear layer.

$$max(0.0, x)$$

- Batch normalization (BatchNorm1d) is a technique to normalize the inputs of each layer, which is usually done between linear and non-linear transformations. It rescales the activations and stabilizes the learning process by reducing the internal covariate shift. In this implementation, we are applying BatchNorm1d to the output of each linear layer to standardize the inputs to the activation function of the next layer, which helps improve the training speed and stability of the model.
- Dropout is a regularization technique that is used to prevent overfitting in neural networks. During training, some random neurons are turned off or “dropped out” with a specified probability. This helps in preventing the network from relying too much on a few neurons and promotes the learning of more robust features. We set the dropout rate to 0.5, meaning that 50% of the neurons are dropped out at each iteration.
- Sigmoid activation function is used in neural networks to introduce non-linearity in the output of a neuron. It maps any input value to a value between 0 and 1, which can be interpreted as a probability. It is commonly used in binary classification problems, where the output is interpreted as the probability of belonging to the positive class.

$$1.0/(1.0 + e^{-}x)$$

- Binary cross entropy is a commonly used loss function in Multilayer Perceptron (MLP) models for binary classification tasks. It measures the difference between predicted and true binary labels and provides a measure of how well the model is performing in terms of accuracy.

$$BinaryCrossEntropy = H_p(q) = -\frac{1}{N}\sum_{i=1}^{n} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) + \frac{\lambda}{2}||\theta||^2$$

Data Leakage

- It is important to make sure that the data is free from any mistakes or assumptions that can impact the EDA, our interpretation of the model, and the actual model that we create. Leakage happens when data is present during training but not during testing and can be a concern when dealing with missing values. But we were able to handle the missing values during feature engineering.
- There are many cardinal sins of machine learning, such as poor data quality, improper data splitting leading to overfitting, relying on only one scoring metric, overinterpreting results, and confusion due to excessive cross-validation. To ensure that these concerns were addressed, we made sure that the data was well structured and missing values were handled properly; overfitting was avoided by properly splitting the data; multiple scoring metrics were used for evaluation; we tabulated our results to avoid over-interpretation; and multiple cross-validations were used to avoid confusion.
- In addition to the above points, we ensured that proper documentation and version control was done to avoid any mistakes in reproducing the results, the right model and hyperparameters were chosen, and the model was validated on diverse datasets to further improve the robustness of the model.

Neural Network

The below architecture is used primarily to experiment with different batch sizes, epochs, and the number of nodes in hidden layers.

	Architecture string	Optimizer	Epochs	Train AUC	Valid Auc	Test AUC	Total experiment time
0	199-relu-32-relu-16-relu-8-sigmoid-1	adam	50	76.2%	67.10000000000001%	73.14872195693694%	487.11 secs
1	199-relu-64-relu-32-relu-16-sigmoid-1	adam	50	62.8%	55.400000000000006%	71.84095564070823%	433.81 secs
2	199-relu-64-relu-32-relu-16-dropout-sigmoid-1	adam	10	56.3%	55.00000000000001%	76.73255313226835%	203.19 secs

Machine Learning Models and Metrics

Modeling Pipeline

Machine Learning algorithms:

The model that we are aiming to build should have the capability to classify the target variable into two classes, i.e. 1 or 0 which are the numeric representation for whether the person who took out a loan will repay it or not based on all the data provided to us in the dataset. For achieving this with accurate results we are planning on trying the following algorithms to see which one provides us with the best outcome:

1. **Logistic Regression** - This is a form of statistical model that is used to examine the correlation between a binary or categorical dependent variable and one or more independent variables. It employs a logistic function to simulate the likelihood that the dependent variable will fall into a certain category. The logistic function converts any input to a number between 0 and 1, which represents the likelihood of the binary result. Logistic regression makes use of Log Loss.

$$\text{Log Loss } = -\frac{1}{N}\sum_{i=1}^{n} y_i * log(p(y_i)) + (1-y_i) * log(1-p(y_i)) + \frac{\lambda}{2}||\theta||^2$$

2. **Gaussian Naive Bayes** - It is a variant of the Naive Bayes algorithm for classification tasks in machine learning. It is predicated on the Bayes theorem and the supposition of data feature independence. The input characteristics in Gaussian Naive Bayes are thought to have a Gaussian or normal distribution. With the input data, the algorithm determines the likelihood of each class, and then it chooses the class with the highest probability as the anticipated output.

3. **Decision Tree Classification** - In this model, based on the characteristics of the input data, a tree-like model of decisions and their potential outcomes is built. The method begins with the complete dataset and chooses the optimal feature to divide the data according to a certain criterion, such Gini Impurity or Information Gain. The data is then divided into subsets depending on the selected feature and the procedure is then performed iteratively for each subset until all the data in each subset belong to the same class, or a stopping requirement is satisfied.

4. **Random Forest Classification** - This model builds numerous decision trees using subsets of the input data and randomly choosing features for each split. Each decision tree only sees a subset of the data since they are trained on a bootstrapped sample of the data. The decision trees' majority vote determines the algorithm's final result.

5. **Neural Networks** - Neural networks are a class of models that learn by adjusting the weights of the input features to generate a desired output. The input features are fed into a network of interconnected neurons, which transform the input through a series of mathematical operations. These transformations are governed by a set of weights that are learned during training. The output is then compared to the actual target output and the weights are updated using a loss function, such as cross-entropy loss.

$$BinaryCrossEntropy = H_p(q) = -\frac{1}{N}\sum_{i=1}^{n} y_i * log(p(y_i)) + (1 - y_i) * log(1 - p(y_i)) + \frac{\lambda}{2}||\theta||^2$$

Description of metrics and analysis:

The evaluation metrics that we want to use to check whether our model is performing well or not are as follows:

1. **Confusion Matrix** - A confusion matrix is a table that is frequently used to assess the effectiveness of a classification model. It is a matrix that compares the predicted labels of the model with the actual labels of the data. The confusion matrix consists of the following cells:
    a. True Positive(TP) - The number of rows that are correctly predicted as positive by the model.
    b. True Negative(TN) - The number of rows that are correctly predicted as negative by the model.
    c. False Positive(FP) - The number of rows that are incorrectly predicted as positive by the model.
    d. False Negative(FN) - The number of rows that are incorrectly predicted as negative by the model.

2. **Classification Report** - An evaluation of a classification model's performance is summarized in a classification report, which is commonly produced using metrics derived from a confusion matrix. A classification report generally includes the following metrics:
    a. **Precision** - The ratio of true positives among all the instances that the model classified as positive.

$$Precision = \frac{TP}{TP + FP}$$

    b. **Recall** - The ratio of true positives among all the instances that actually belong to the positive class of the model.

$$Recall = \frac{TP}{TP + FN}$$

c. **F1-Score** - The harmonic mean of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN}$$

3. **Accuracy Score** - A typical performance metric in machine learning for assessing the accuracy of a classification model is the accuracy score. It is defined as the ratio of correctly classified instances out of the total number of instances in the dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

4. **ROC Curve** - An ROC curve is a graphical representation of the performance of a binary classification model at different classification thresholds. In machine learning, it is frequently used to assess and contrast the effectiveness of various categorization models. The True Positive Rate and False Positive Rate are plotted on the y-axis and x-axis respectively.

Machine Learning Pipelines:

For solving this problem statement we plan on following the below explained pipeline:

1. **Problem Statement and Requirements** - The problem statement in a Machine Learning (ML) project specifies the particular aim or task that the ML model is meant to address which in this case would be to determine if a client will be able to repay a loan or not. The requirements of the project is to specify what is necessary to create a powerful ML model that solves the given problem. This contains any performance measures that will be monitored, the necessary data, the features to be used, the model architecture, and the method to be used. It should also outline the necessary hardware and software for developing and deploying the ML model.

2. **Data Acquisition** - The data needed for solving this problem statement is available on kaggle as Home Credit Default Risk. The data is divided into 7 csv files that we can use to come up with a solution.

3. **Data Visualization and EDA** - The graphical depiction of data and information is known as data visualization. In order to help illustrate patterns, trends, and correlations within the data, visualizations like charts, graphs, and maps are made. EDA on the other hand is the process of applying statistical and graphical tools to analyze and summarize a dataset's key features. EDA uses a number of approaches, including outlier detection, correlation analysis, and summary statistics. EDA would play a key role in ensuring our data's accuracy and quality and is a crucial stage in creating a machine learning model that works for our problem statement.

4. **Feature Engineering** - Implementing techniques for feature engineering that include combining existing features to create new ones, converting features using mathematical operations like exponential or logarithmic transformations, and normalizing or standardizing features to guarantee that they are of the same scale. Additional methods include handling missing data, encoding categorical variables, and principal component analysis for dimensionality reduction.

5. **ML Model and Pipeline Development** - The aim in this section is to build various ML models like; logistic regression, support vector classification, gaussian naive bayes, random forest classification, decision tree classification and to build the following three pipelines:
    a. **Numeric Pipeline** - This pipeline works with the numeric part of our data to process it. We plan on making use of SimpleImputer to take care of missing values and we also plan on including StandardScaler transformation for standardizing the data.
    b. **Categoric Pipeline** - This pipeline works with the categorical part of our data to help process it. In this pipeline we plan on making use of SimpleImputer for dealing with missing values by using the concept of mode. We make use of OneHotEncoder to convert categorical data into numeric values. We also make use of scaler to transform the present data.
    c. **Column Transformer** - This is the resultant pipeline that is a combination of the above two numeric and categorical pipelines.

6. **Model Evaluation** - In this step we fit the above made models and pipelines together and train the various models on the training data. We then use various evaluation metrics as previously mentioned like; confusion matrix, classification report, accuracy score and ROC curve to determine the best model among the ones made.

7. **Model Deployment** - The process of integrating a trained ML model into a production environment so that it may be used to generate predictions or categorize new data is known as model deployment.

Phase Leader Plan

Phase Number	Phase Leader	Tasks
Phase 1	Shyam Makwana	Project proposal, Layout, Notebook, Credit assignment plan, Gantt chart, Dataset
Phase 2	Aarushi Dua	Grab data, EDA, Metrics, Baseline models, Baseline pipeline, Brief report, Presentation and script
Phase 3	Sai Teja Burla	Feature engineering, Hyperparameter tuning, Additional feature selection, ensemble methods, Presentation and script
Phase 4	Lakshay Madaan	Neural Network, Advanced model architectures, Loss functions, Final project presentation and report

Credit Assignment Plan

Phase	Member Name	Task Name	Description
Phase 1	Aarushi Dua Sai Teja Burla Shyam Makwana Lakshay Madaan	Problem Understanding	Comprehending the problem and understanding the provided data
	Shyam Makwana	Assigning Tasks	Assigning tasks to each of the team members
		Phase Planning	Planning all the phases
		Data Description	Describing what information each of the given csv files have with a sample of their content
	Sai Teja Burla	Abstract	Brief description of the problem statement at hand
		Researched ML models and metric	Description of ML Models and Evaluation Metrics
		Block Diagram	Visual representation of the pipeline
	Aarushi Dua	Model Pipeline	Explained the steps to perform during the course of the project
		Gantt Chart	Summarize the timeline of the project using gantt charts
	Aarushi Dua Sai Teja Burla	Credit Assignment Plan	Design the four phases and describe the task required.
	Lakshay Madaan	Environmental Setup	Connected Data with docker
		Data Preparation	Checked Dataset are loading and performing basic data visualization.
Phase 2	Aarushi Dua Sai Teja Burla	Data Visualization and EDA	Visualize the dataset to gather insights and perform different data pre-processing techniques.
	Lakshay Madaan Shyam Makwana	Feature Engineering and Pipeline Creation	Data Transformation is required and will add/remove necessary features. Along with creating a pipeline for numeric and categorical features.
	Aarushi Dua	Model Training	Fit the training data on various ML algorithms using pipelines

	Sai Teja Burla	Model Evaluation	Apply different evaluation metrics to get the accuracy and loss
	Shyam Makwana	Research about different Hyperparameters	Gather hyperparameters of each model applied into the GridsearchCV
	Lakshay Madaan	Comparing Baseline Models	Compare the results of each baseline model.
Phase 3	Sai Teja Burla	Feature Importance	Figure out the importance of each features used
	Aarushi Dua	Tuning Hyperparameters	Hyperparameter tuning is performed again to improve the performance.
	Shyam Makhwana	Visualizing different algorithm's results	Plot bars and charts to display the results of each model
	Lakshay Madaan	Feature Engineering	Any further manipulation in features if required
Phase 4	Lakshay Madaan	Final Model Evaluation	Perform final model evaluation on the best model achieved.
	Shyam Makhwana	Report Writing and Video Editing	Write report for final submission and record the video presentation
	Sai Teja Burla Aarushi Dua	Adding Multilayer Perceptron	Introduce advance ML model; i.e Pytorch model for further improvement in model's accuracy

Experimental Results and Kaggle Submission

Featured Prediction Competition

Home Credit Default Risk

Can you predict how capable each applicant is of repaying a loan?

Home Credit Group · 7,176 teams · 5 years ago

$70,000
Prize Money

Overview Data Code Discussion Leaderboard Rules Team Submissions **Late Submission**

Submissions

0/2

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

Submissions evaluated for final score

All Successful Selected Errors Recent

Submission and Description	Private Score	Public Score	Selected
MLP_submission5.csv Complete (after deadline) · 40m ago · MLP_submission5	0.72903	0.72821	☐
MLP_submission4.csv Complete (after deadline) · 1h ago · MLP_submission4	0.67462	0.6734	☐
MLP_submission3.csv Complete (after deadline) · 1h ago · MLP_submission3	0.69961	0.6952	☐
MLP_submission2.csv Complete (after deadline) · 2h ago · MLP_submission2	0.73287	0.73602	☐
MLP_submission1.csv Complete (after deadline) · 6h ago · MLP_submission1	0.71744	0.72172	☐
MLP_submission.csv Complete (after deadline) · 8h ago · MLP_submission	0.70128	0.71193	☐

Leaderboard

Raw Data Refresh

YOUR RECENT SUBMISSION

MLP_submission5.csv
Submitted by AaruDua · Submitted 41 minutes ago

Score: 0.72903
Public score: 0.72821

Jump to your leaderboard position

Discussion

While building the MLP, we had the following discussion about its architecture before deciding on the best model.

1. As a preprocessing technique, we implemented both Random UnderSampling as well as Smote for oversampling to see which could help make a better model but noticed that both of these were not helping with making a better model than the ones we already have so we decided to go with a simple pipeline that we implemented before for our regular models.
2. We used BCELoss as the loss function since we are dealing with binary target variables.
3. For the optimizers we made use of SGD and Adam but as a part of our initial experiment phase we saw that SGD was not helping at all so we ended up showcasing only Adam as our main optimizer.
4. We also experimented with learning rate values 0.01 and 0.001 and noticed that 0.001 always gives us better results.
5. We used different batch sizes to experiment and noticed that lesser batch sizes is giving us better results.
6. Coming to the architecture, we experimented with 2 or 3 hidden layers in each of our experiments.
7. Weight decay was also experimented with but the accuracy kept remaining the same throughout the epochs due to which we decided not to use it.

Gap Analysis

When we looked into the leaderboard we observed that the other groups have implemented LGBM which might have played a key role in their public kaggle score being higher than ours. We also saw that they made batch normalization layers in their MLP model which could have also helped with the same. Added to this their initial number of features are more than ours. There could also be a difference in our preprocessing methodologies.

Conclusion

In conclusion, our project focused on predicting the risk of loan defaults for Home Credit using machine learning techniques. This is an important problem as loan defaults can have a significant impact on financial institutions and their clients.

In phase 2, we conducted EDA to identify significant features and created ML pipelines using various classifiers. Underfitting was observed in Gaussian Naive Bayes and overfitting in Decision Tree and Random Forest. In the next phase, advanced feature engineering and hyperparameter tuning will be implemented to enhance model evaluation metrics.

In phase 3, we explored, cleaned, and engineered features for both categorical and numerical data, trained multiple models, and optimized their hyperparameters. Our best model achieved a ROC AUC score of 0.78 on the test data, outperforming the baseline. Important features for predicting loan defaults were past payment history, income, credit bureau reports, and external sources. In the future, we plan to implement a new multitask loss function in PyTorch to predict both the class and length of time before defaulting. Our findings and model could have practical implications for mitigating the risk of loan defaults in financial institutions.

In this phase, we built a neural network with 199 input features, 3 hidden layers (64, 32, 16 neurons), and 1 output features, using the ReLU activation function on the hidden layers and the Sigmoid activation function on the output layer and Adam Optimizer. Cross Entropy Loss Function was used for evaluation. The MLP model had a test roc score of 0.76, and a Kaggle submission public score of 0.728. However, the AUC score of the non-deep learning models (Logistic Regression, Random Forest, and XGBoost) was better. The MLP model can be improved using embeddings or advanced classification models like boosting.

Bibliography

Matplotlib — Visualization with Python, https://matplotlib.org. Accessed 11 April 2023.

seaborn: statistical data visualization — seaborn 0.12.2 documentation, https://seaborn.pydata.org. Accessed 11 April 2023.

"Home Credit Default Risk." *Kaggle*, https://www.kaggle.com/c/home-credit-default-risk. Accessed 11 April 2023.