

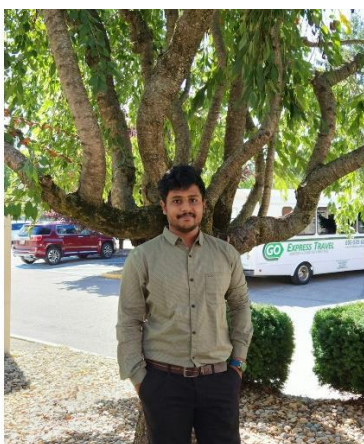
Home Credit Default Risk (HCDR)

Group 11

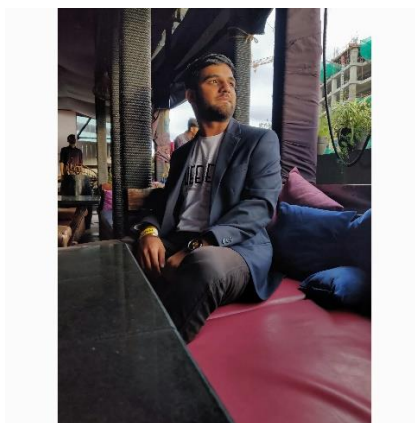
Anuj Mahajan
Siddhant Patil
Shashwati Diware
Shubham Jambhale

Team Members:

Shubham Jambhale
sjambhal@iu.edu



Siddhant Patil
sidpatil@iu.edu



Anuj Mahajan
anujmaha@iu.edu



Shashwati Diware
sdiware@iu.edu



Phase Leader Plan

Phase	Contributor	Contribution Description
Phase 1: Project Planning	Anuj Mahajan	Download Data, go through data, and load libraries. Create a pipeline diagram and describe the pipeline design. Describe Preprocessing,
Phase 1: Project Planning	Shashwati Diware	Project Abstract, ML Algorithm Names, and describe Metrics.
Phase 1: Project Planning	Shubham Jambhale(Phase Leader)	Understanding the problem statement, and writing table descriptions. Schedule meetings, coordinate tasks, plan phase
Phase 1: Project Planning	Siddhant Patil	Machine Learning Pipeline Steps and describes pipeline components.
Phase 2: Base Line Modelling and EDA	Anuj Mahajan (Phase Leader)	Creating Block Diagram EDA and one slide of the presentation. Schedule meetings, coordinate tasks, plan phase
Phase 2: Base Line Modelling and EDA	Shashwati Diware	Result Analysis EDA and one slide of the presentation.
Phase 2: Base Line Modelling and EDA	Shubham jambhale	Result Analysis and two slides of the presentation
Phase 2: Base Line Modelling and EDA	Siddhant Patil	Result Analysis and two slides of the presentation
Phase 3: Hyperparameter Tuning	Shashwati Diware (Phase Leader)	Testing Accuracy matrix and Schedule meetings, coordinating tasks, the planning phase
Phase 3: Hyperparameter Tuning	Siddhant Patil	Create and develop code for Hyperparameter tuning
Phase 3: Hyperparameter Tuning	Shubham Jambhale	Run and create analysis by testing the confusion / AUC matrix. Coordinate Tasks and one slide of the presentation
Phase 3: Hyperparameter Tuning	Anuj Mahajan	Run and analyze Lasso and ridge regression losses. Coordinate tasks and one slide of the presentation
Phase 4: Final Report Generation	Siddhant Patil (Phase Leader)	Plan Phase Schedule Meetings and Coordinate Tasks, analyze and go through the final results
Phase 4: Final Report Generation	Anuj Mahajan	Rearrange everything and go through the final documentation, list down the final recordings
Phase 4: Final Report Generation	Shashwati Diware	Prepare the final presentation
Phase 4: Final Report Generation	Shubham Jambhale	Check everything and submit the assignment before the deadline
Phase 4: Final Report Generation	Group	Build a Multi-Layer Perceptron using PyTorch.

Credit Assignment Plan

Phase 1:

Task	Task Description	Hours spent	Assigned to	Start	End
Understanding problem statement	Go through the problem statement to understand the requirements	6	Shubham	11/05/22	11/07/22
Data Exploration	Explore and analyze the data for a better understanding	6	Anuj	11/07/22	11/09/22
Project Proposal	Creating the project proposal and preparing a basic report with Abstract, ML models, and Gantt diagram	20	Group	11/09/22	11/14/22

Phase 2:

Task	Task Description	Hours Spent	Assigned to	Start	End
Creating Block Diagram	Creating the block diagram of the basic flow of execution.	5	Anuj	11/13/22	11/15/22
Creating Pipeline Diagram	Creating the pipeline diagram of the machine learning model from analyzing the data till the result analysis	5	Shashwati	11/13/22	11/15/22
Result Analysis	Analyzing the Result	10	Group	11/26/22	11/29/22
PowerPoint Presentation	Simultaneously prepare the PowerPoint presentation and add the analyzed data into it as per need	10	Group	11/20/22	11/29/22

Phase 3:

Task	Task Description	Hours spent	Assigned to	Start	End
Create and develop code for hyperparameter tuning	Design and develop python helper function for hyperparameter tuning	16	Siddhant	11/20/22	11/25/22
Result Analysis	Analysis of Obtained Result	2	Group	12/02/22	12/03/22
Testing Accuracy matrix	Analyzing accuracy using accuracy matrix	2	Shashwati	12/03/22	12/04/22
Analyzing the Loss and AUC	Analyzing Loss and AUC matrix	2	Shubham	12/03/22	12/04/22

Creating Powerpoint presentation	Simultaneously prepare the PowerPoint presentation and add the analyzed data into it as per need	2	Anuj	12/03/22	12/04/22
----------------------------------	--	---	------	----------	----------

Phase 4:

Task	Task Description	Hours Spent	Assigned To	Start	End
Build MLP using Pytorch	Build a neural network model using Pytorch	10	Group	12/03/22	12/08/22
Final Documentation	Rearrange everything and go through the final documentation, list down the final recordings	10	Anuj	12/03/22	12/13/22
Final Results	Analyze final results obtained after the final testing	6	Siddhant	12/05/22	12/12/22
Final Presentation	Prepare the final presentation	4	Shashwati	12/06/22	12/13/22
Assignment Submission	Check everything and submit the assignment before the deadline	1	Shubham	12/08/22	12/13/22

Abstract

The main objective of this project is to create the best machine learning model that can determine whether a loan application will be able to repay the loan. We examined the data in Phase 1 and completed our initial baseline models. We expanded our work from Phase 1 to Phase 2 and applied fundamental feature engineering, where we considered probable characteristics from other tables and constructed base models. In terms of performance on these data, logistic regression performed the best and decision tree and random forest came in second and third, respectively.

In Phase 3, we carefully chose the features from the generated features, analyzed the significance of the features, and applied hyper parameter tuning. We improved upon our baseline model using feature engineering, yielding 4 more characteristics which were Salary-to Credit Ratio, Total External Source AMT Credit to Annuity Ratio, Annuity to Salary Ratio. On baseline models, we performed hyper parameter adjustment, and Lasso and Ridge were also used, and we achieved the test accuracy of 92.13% for Decision Tree with AUC 71.8% for Lasso and Ridge. Our Kaggle Score were 0.71673(private) and 0.73086(public)

In Phase 4, we ran a deep learning algorithm to forecast the project's eventual aim. The Multi-Layer Perceptron was the Deep Learning algorithm that was employed. Implementing MLP and displaying the training model on TensorBoard were the primary objectives of this phase. Our accuracy increased to 92.4% along with Testing AUC of 60.13%. We found out that there is slight leakage in the pipeline and we think that is the reason our Kaggle Score was decreased to 0.504 even after receiving the testing AUC of 0.6013

Data and Task Description

Data source

We are planning to use the existing datasets provided by Kaggle.

Source: <https://www.kaggle.com/c/home-credit-default-risk/data>

POS_CASH_balance.csv

This dataset gives information about previous credit information such as contract status, the number of installments left to pay, DPD(days past due), etc. of the current application.

Table 1. POS_CASH_balance.csv

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

bureau.csv

This dataset gives information about the type of credit, debt, limit, overdue, maximum overdue, annuity, remaining days for previous credit, etc.

Table 2. Bureau.csv

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVERDUE	DAYS_CREDIT_ENDDATE
0	215354	5714462	Closed	currency 1	-497	0	-153.0
1	215354	5714463	Active	currency 1	-208	0	1075.0
2	215354	5714464	Active	currency 1	-203	0	528.0
3	215354	5714465	Active	currency 1	-203	0	NaN
4	215354	5714466	Active	currency 1	-629	0	1197.0

bureau_balance.csv

This dataset gives information about the Status of the Credit Bureau loan during the month, the Month of balance relative to the application date, Recoded ID of the Credit Bureau credit. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.

Table 3. bureau_balance.csv

	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

credit_card_balance.csv

This dataset gives information about financial transactions aggregated values such as amount received, drawings, number of transactions of previous credit, installments, etc. Each row is one month of a credit card balance, and a single credit card can have many rows.

Table 4. credit_card_balance.csv

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_DRAWINGS_ATM_CURRENT	AMT_DRAWINGS_CURRENT
0	2562384	378907	-6	56.970	135000	0.0	877.5
1	2582071	363914	-1	63975.555	45000	2250.0	2250.0
2	1740877	371185	-7	31815.225	450000	0.0	0.0
3	1389973	337855	-4	236572.110	225000	2250.0	2250.0
4	1891521	126868	-1	453919.455	450000	0.0	11547.0

installments_payments.csv

This dataset gives information about payments, installments supposed to be paid, and their details. There is one row for every made payment and one row for every missed payment.

Table 5. Installments_payments.csv

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALLMENT_VERSION	NUM_INSTALLMENT_NUMBER	DAYS_INSTALLMENT	DAYS_ENTRY_PAYMENT	AMT_INSTALLMENT
0	1054186	161674	1.0	6	-1180.0	-1187.0	6948.360
1	1330831	151639	0.0	34	-2156.0	-2156.0	1716.525
2	2085231	193053	2.0	1	-63.0	-63.0	25425.000
3	2452527	199697	1.0	3	-2418.0	-2426.0	24350.130
4	2714724	167756	1.0	2	-1383.0	-1366.0	2165.040

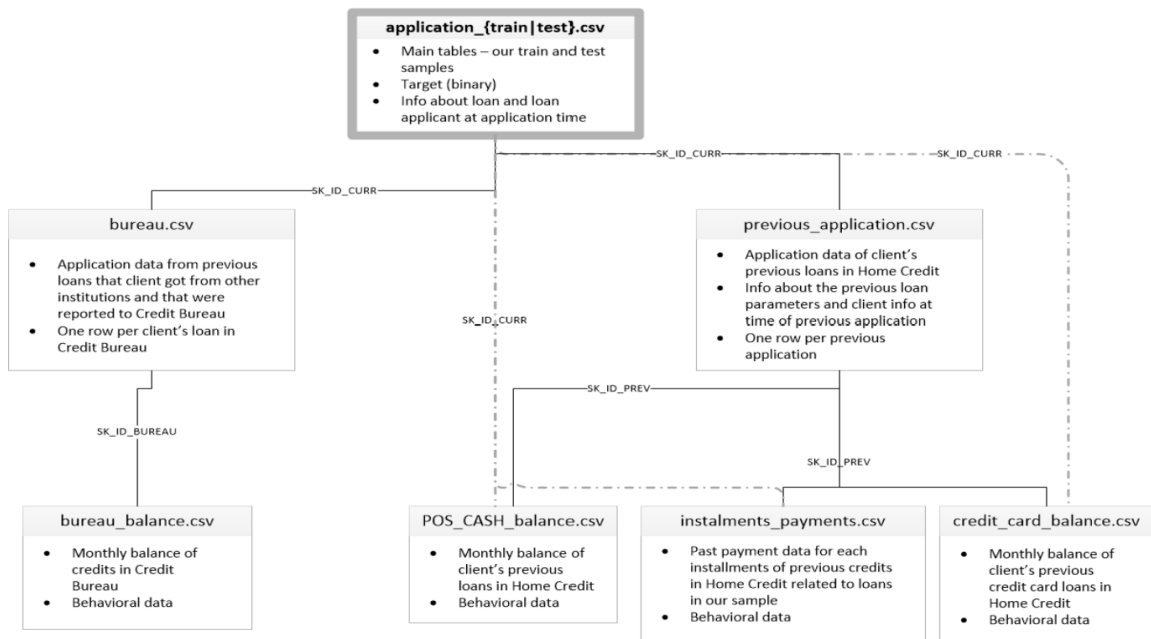
previous_application.csv

This dataset contains information about previous application details of an application. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.

Table 6. previous_application.csv

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT
0	2030495	271877	Consumer loans	1730.430	17145.0	17145.0	0.0
1	2802425	108129	Cash loans	25188.615	607500.0	679671.0	NaN
2	2523466	122040	Cash loans	15060.735	112500.0	136444.5	NaN
3	2819243	176158	Cash loans	47041.335	450000.0	470790.0	NaN
4	1784265	202054	Cash loans	31924.395	337500.0	404055.0	NaN

Figure 1: Data Description Diagram



Task To Be Tackled:

- We need to predict the customer repayment status such as if the user is a defaulter or not by Multi-Layer Perceptron using Pytorch using the datasets provided by Kaggle.

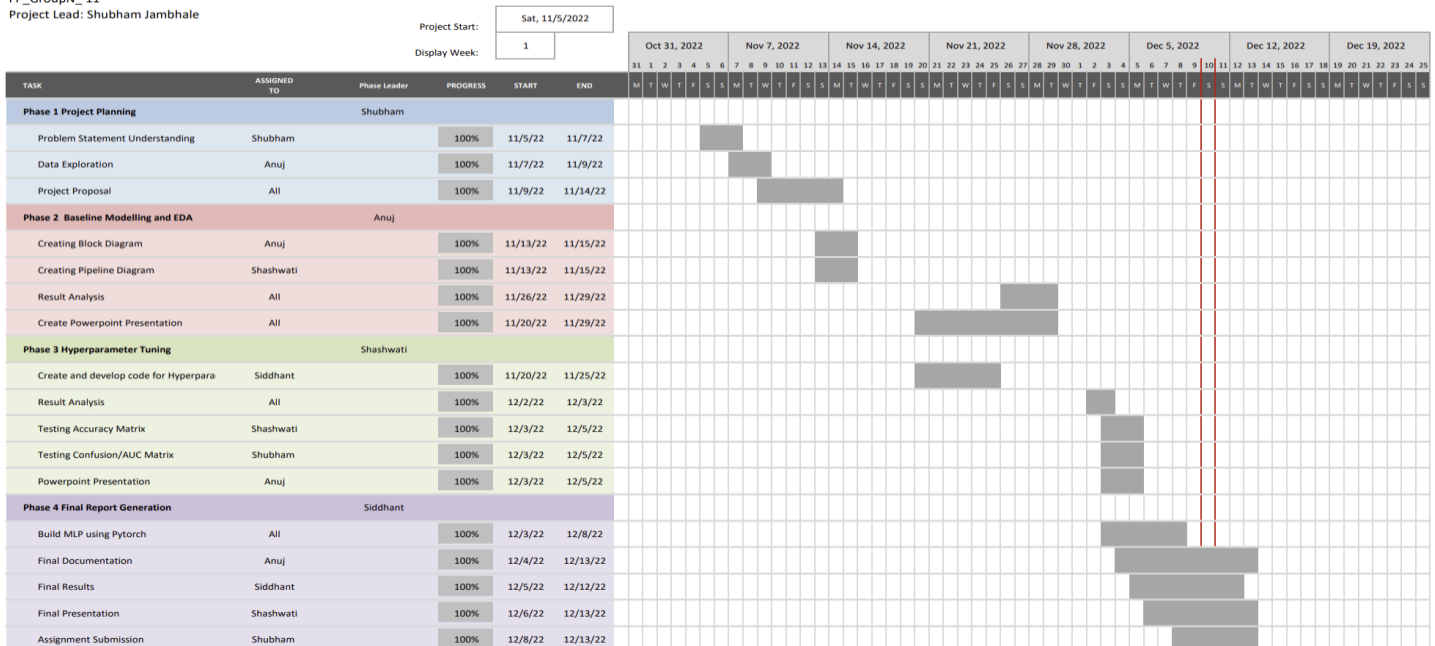
Gantt Chart

Figure 2. Gantt Chart

CSCI-P 556: Applied Machine Learning

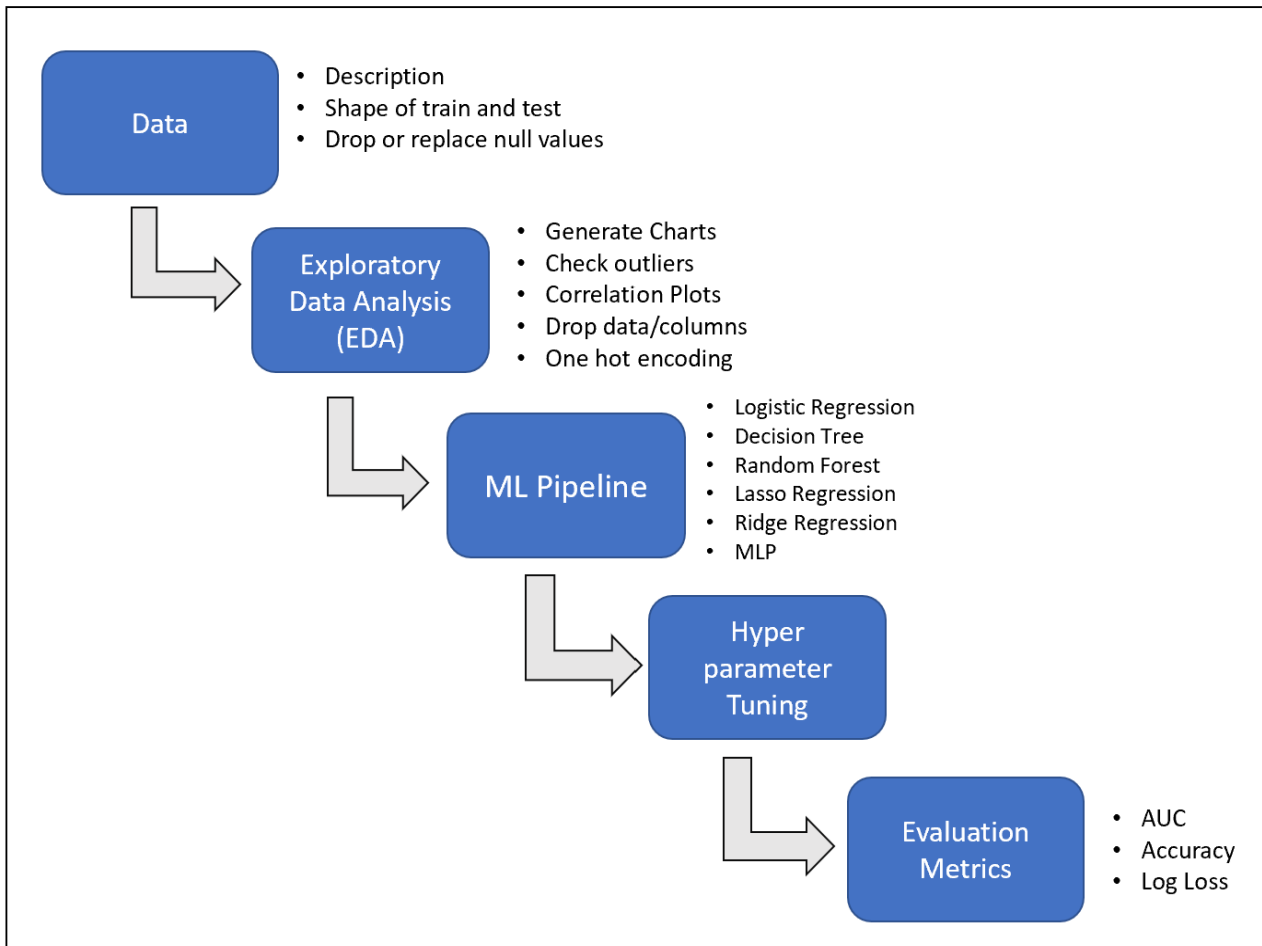
FP_GroupN_11

Project Lead: Shubham Jambhale



Machine Learning Workflow:

Figure 3: Workflow



Machine Learning Algorithm and Metrics

The outcome of this project is to predict whether the customer will repay the loan or not. That's why this is a classification task where the outcome is 0 or 1. To classify this problem we will be building the following machine-learning models:

1. **Logistics Regression:**

- In our case, the number of features is relatively small i.e. <1000, and no. of examples is large. Hence logistic regression can be a good fit here for the classification.

2. **Decision Tree:**

- Decision trees are better for categorical data and our target data is also categorical in nature that's why decision trees are a good fit.

3. **Random Forest:**

- Random Forest works well with a mixture of numerical and categorical features.
- As we have a good amount of mixture of both types of features random forest can be a good fit.

4. Lasso Regression:

- The bias-variance trade-off is the basis for Lasso's superiority over least squares. The lasso solution can result in a decrease in variance at the cost of a slight increase in bias when the variance of the least squares estimates is very large. Consequently, this can produce predictions that are more accurate.

5. Ridge Regression:

- Any data that exhibits multicollinearity can be analyzed using the model-tuning technique known as ridge regression. This technique carries out L2 regularization. Predicted values differ much from real values when the problem of multicollinearity arises, least-squares are unbiased, and variances are significant.

6. Multi-Layer Perceptron:

- The perceptron can be used to define a linear decision boundary in a binary classification model. The separating hyperplane that reduces the separation between incorrectly classified points and the decision boundary is found. Input and output layers, as well as one or more hidden layers with a densely packed number of neurons, make up a multilayer perceptron.

Loss Function:

➤ BCE Loss:

- The binary cross entropy loss between the input and target (predicted and actual) probability is calculated using the BCELoss() function.

$$\text{BCELoss}() = -w (y \log(x) + (1 - y) \log(1 - x))$$

Metrics:

1. Confusion Metrics:

- A confusion matrix, also called an error matrix, is used in the field of machine learning and more specifically in the challenge of classification. Confusion matrices show counts between expected and observed values. The result "TN" stands for True Negative and displays the number of negatively classed cases that were correctly identified. Similar to this, "TP" stands for True Positive and denotes the quantity of correctly identified positive cases. The term "FP" denotes the number of real negative cases that were mistakenly categorized as positive, while "FN" denotes the number of real positive examples that were mistakenly classed as negative. Accuracy is one of the most often used metrics in classification.

Figure 4: Confusion Matrix

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

2. AUC:

- AUC stands for "Area under the ROC Curve." It measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1). It is a widely used accuracy method for binary classification problems

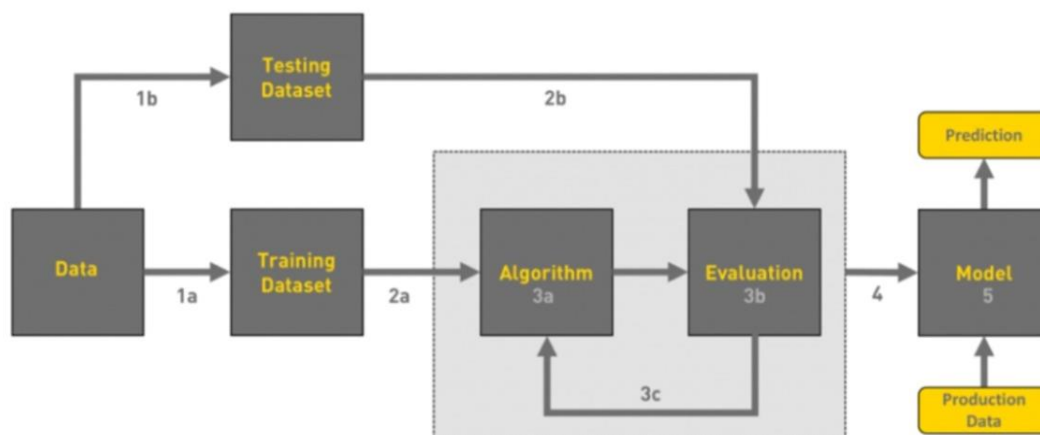
3. Accuracy:

- The accuracy score is used to gauge the model's effectiveness by calculating the ratio of total true positives to total true negatives across all made predictions. Accuracy is generally used to calculate binary classification models.

$$\text{Accuracy}() = \frac{\text{True}_{\text{positives}} + \text{True}_{\text{Negatives}}}{\text{True}_{\text{positives}} + \text{True}_{\text{Negatives}} + \text{False}_{\text{positives}} + \text{False}_{\text{Negatives}}}$$

Block Diagram:

Figure 5: Block Diagram of Project



Overview of the Workflow of ML

Referenced from: <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>

Feature Extraction

➤ **Step 1:**

- Following columns give us the number of enquiries done.
- AMT_REQ_CREDIT_BUREAU_HOUR
- AMT_REQ_CREDIT_BUREAU_DAY
- AMT_REQ_CREDIT_BUREAU_WEEK
- AMT_REQ_CREDIT_BUREAU_MON
- AMT_REQ_CREDIT_BUREAU_QRT
- AMT_REQ_CREDIT_BUREAU_YEAR

In all these columns for some entries values are not mentioned, so we can assume that there are no enquiries for those inputs. We can replace null values in such inputs by 0

➤ **Step 2:**

- Following columns give us the number of immediate connections who have a loan in Home Credit.
- OBS_30_CNT_SOCIAL_CIRCLE
- DEF_30_CNT_SOCIAL_CIRCLE
- OBS_60_CNT_SOCIAL_CIRCLE
- DEF_60_CNT_SOCIAL_CIRCLE

In all these columns for some entries, values are not mentioned, so we can assume that there are no immediate connections for those inputs. We can replace null values in such inputs with 0.

➤ **Step 3:**

- AMT_GOODS_PRICE values are depending on NAME_FAMILY_STATUS categories. So we replaced null/N/A values with medians with respect to NAME_FAMILY_STATUS.

➤ **Step 4:**

- Used zero to fill in the empty or missing values for CNT FAM MEMBERS

➤ **Step 5:**

- We need to replace EXT SOURCE 3 null or N/A values. For this, we discovered some of the highly associated variables. We found out that there is a high correlation between EXT SOURCE 3 and DAYS BIRTH. Given that DAYS BIRTH is numerical, we used Linear Regression to fill in the null or NA values.

➤ **Step 6:**

- In order to replace the EXT SOURCE 2 null or N/A values, we discovered the some of the highly associated variables. We found out that there is high correlation between EXT SOURCE 2 and REGION RATING CLIENT. Due to the categorical nature of REGION RATING CLIENT, we fill null or NA values with the median depending on categories.

Including new features in the training data.

We tested and trained on a few chosen columns mentioned below,

- Salary-to-Credit Ratio
- Total External Source

Additional Feature Engineering:

- Along with the above-mentioned feature engineering we performed the following additional feature engineering
 - **Step 1:**
 - In the baseline pipeline, we eliminated features having 50 percent or more null values.
 - We analyzed that we can drop features having 30 % or more null values as there is no significant impact on those features.
 - **Step 2:**
 - Dropped rows from DAYS_LAST_PHONE_CHANGE, AMT_ANNUITY column because there are very few rows with null values.

Including new features in the training data.

We tested and trained on a few chosen columns mentioned below

- Salary-to-Credit Ratio
- Total External Source
- AMT Credit to Annuity Ratio.
- Annuity to Salary Ratio.

Besides the Salary-to-credit-Ratio and Total external Source added in the baseline, in additional feature engineering, we added 2 more columns, the Amt credit to annuity ratio and annuity-to-salary ratio.

We analyzed that ratio of AMT_CREDIT to AMT_ANNUITY and the ratio of AMT_ANNUITY to AMT_INCOME_TOTAL can provide us the better result in the aspect of prediction.\

Impact of newly added Feature:

- We analyzed that newly added features impacted the decision tree model by a significant amount.
- Test accuracy of the decision tree model is increased by 6 % due to the newly added features as compared to its baseline model.
- There is no significant impact on logistic regression and the random forest model.

Why we choose the technique and strategy:

- We removed some of the features which have the greatest number of null values, as they will contribute very little to the predictions.
- For other features, It is best to handle filling in null values for categorical and continuous variables individually. It won't be possible to fill all category variables with the most or least frequent values and all continuous variables with the median value.
- A reliable metric to assess a person's reliability and repayment capacity would be ratios between income, credit requested, and credit to be paid per year. So, we thought about putting above.

Previous Experiments:

➤ Phase 2: Baseline

- Below are the experiments we conducted in Phase 2 for baseline models.

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Accuracy	Loss	Train Time(s)	Test Time(s)	Validation Time(s)	Experiment description
0	Baseline with 81 inputs	92.0	91.9	91.7	0.506443	91.917467	0.246888	12.5042	0.0495	0.0410	Selective Features - Baseline LogisticRegression
1	Baseline Decision Tree with 81 inputs	86.4	86.1	86.3	0.570952	86.148643	2.362000	29.5238	0.0733	0.0594	Selective Features - Baseline Decision Tree
2	Baseline Random Forest with 81 inputs	92.2	92.2	92.0	0.522292	92.187373	0.270439	245.7127	1.2979	1.0884	Selective Features - Baseline Random Forest

➤ Phase 3: Hyper-Parameter Tuning

- Below are the experiments we conducted in Phase 3 for hyper parameter tuning models.

Logistic Regression:

	ExpID	Cross fold train accuracy	Test Accuracy	Valid Accuracy	AUC	Loss	Accuracy	Train Time(s)	Test Time(s)	Experiment description	Best Hyper Parameters
0	Hypertuned Logisitc rgeression with 74 inputs	91.91%	91.98%	91.96%	0.5	0.287519	91.976152	120.5867	0.0364	Hypertuned LogisticRegression with 4 cv, logis...	{'logistic__C': 10, 'logistic__tol': 0.001}
1	Hypertuned Logisitc rgeression with 74 inputs	91.91%	91.98%	91.96%	0.5	0.287519	91.976152	206.4487	0.0150	Hypertuned LogisticRegression with 5 cv, logis...	{'logistic__C': 10, 'logistic__tol': 0.0001}
2	Hypertuned Logisitc rgeression with 74 inputs	91.91%	91.98%	91.96%	0.5	0.287519	91.976152	207.5919	0.0150	Hypertuned LogisticRegression with 10 cv, logi...	{'logistic__C': 10, 'logistic__tol': 0.001}
3	Hypertuned Logisitc rgeression with 74 inputs	91.91%	91.98%	91.96%	0.5	0.287519	91.976152	142.8426	0.0140	Hypertuned LogisticRegression with 5 cv, logis...	{'logistic__C': 10, 'logistic__tol': 0.001}

Decision Tree:

	ExpID	Cross fold train accuracy	Test Accuracy	Valid Accuracy	AUC	Loss	Accuracy	Train Time(s)	Test Time(s)	Experiment description	Best Hyper Parameters
0	Hypertuned Decision Tree with 74 inputs	92.19%	92.13%	92.03%	0.535858	0.248343	92.130081	71.7969	0.0530	Hypertuned Decision Tree with 2 cv, Max Depth ...	{'decisionTree__criterion': 'gini', 'decisionT...
1	Hypertuned Decision Tree with 74 inputs	91.91%	91.98%	91.96%	0.500000	0.257044	91.976152	56.4094	0.0515	Hypertuned Decision Tree with 3 cv, Max Depth ...	{'decisionTree__criterion': 'gini', 'decisionT...
2	Hypertuned Decision Tree with 74 inputs	91.91%	91.98%	91.96%	0.500000	0.257044	91.976152	109.6553	0.0590	Hypertuned Decision Tree with 5 cv, Max Depth ...	{'decisionTree__criterion': 'gini', 'decisionT...
3	Hypertuned Decision Tree with 74 inputs	91.91%	91.98%	91.96%	0.500000	0.257044	91.976152	55.6498	0.0525	Hypertuned Decision Tree with 5 cv, Max Depth ...	{'decisionTree__criterion': 'gini', 'decisionT...

Random Forest:

	ExpID	Cross fold train accuracy	Test Accuracy	Valid Accuracy	AUC	Loss	Accuracy	Train Time(s)	Test Time(s)	Experiment description	Best Hyper Parameters
0	Hypertuned random Forest with 74 inputs	91.91%	91.98%	91.96%	0.5	0.269295	91.976152	147.6134	0.125	Hypertuned Random Forest with 3 cv, Max Depth ...	{'randomForest__max_depth': 1, 'randomForest__...
1	Hypertuned random Forest with 74 inputs	91.91%	91.98%	91.96%	0.5	0.268306	91.976152	262.9420	0.191	Hypertuned Random Forest with 3 cv, Max Depth ...	{'randomForest__max_depth': 1, 'randomForest__...
2	Hypertuned random Forest with 74 inputs	91.91%	91.98%	91.96%	0.5	0.269295	91.976152	68.3533	0.126	Hypertuned Random Forest with 5 cv, Max Depth ...	{'randomForest__max_depth': 1, 'randomForest__...

Lasso and Ridge Regression:

	ExpID	Cross fold train accuracy	Test Accuracy	Valid Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description	Best Hyper Parameters
0	Hypertuned Lasso Regression with 74 inputs	-6.90%	-6.87%	-6.89%	0.755738	37.7049	0.0260	Hypertuned Lasso Regression with 5 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...
1	Hypertuned Lasso Regression with 74 inputs	-6.90%	-6.87%	-6.89%	0.755738	31.7134	0.0262	Hypertuned Lasso Regression with 4 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...
2	Hypertuned Lasso Regression with 74 inputs	-6.90%	-6.87%	-6.89%	0.755738	24.2078	0.0270	Hypertuned Lasso Regression with 3 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...
3	Hypertuned Ridge Regression with 74 inputs	-6.89%	-6.87%	-6.89%	0.756875	8.0231	0.0250	Hypertuned Ridge Regression with 5 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...
4	Hypertuned Ridge Regression with 74 inputs	-6.89%	-6.87%	-6.89%	0.756874	8.1149	0.0278	Hypertuned Ridge Regression with 4 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...
5	Hypertuned Ridge Regression with 74 inputs	-6.89%	-6.87%	-6.89%	0.756862	7.0266	0.0270	Hypertuned Ridge Regression with 3 cv and alph...	{'randomForest__max_depth': 1, 'randomForest__...

Neural Network:

- A neural network component called a perceptron performs specific calculations in order to find features or business information in the input data. It is a mapping function that takes as an input, multiplies it by the learned weight coefficient, and outputs the prediction.
- The perceptron can be used as a binary classification model. It finds the separating hyperplane that minimizes the distance between misclassified points and the decision boundary.
- Perceptron uses Stochastic Gradient Descent as the optimization function.
- There are multiple types of perceptron as below:
 - Single Layer
 - Multi-Layer
- There are different activation functions in perceptron:
 - Sigmoid
 - ReLU
 - Tanh,
 - Identity
- In this phase we have used sigmoid and ReLU as activation function.

Experiments:

We have conducted multiple experiments on the Pytorch MLP model. Below is the latest experiment logs among all the experiments we have done.

We have performed various combinations including permutations of epoch and activation functions, below are the most recent one.

	ExpID	Train Time	Test Time	Accuracy	AUC	Comments
0	Multi Layer Perceptron	363.6610	0.0234	0.915039	0.601393	150 - ReLU + Sigmoid
1	Multi Layer Perceptron	366.6223	0.0326	0.932617	0.500000	150 - ReLU
2	Multi Layer Perceptron	160.7423	0.0156	0.920000	0.527774	100 - Sigmoid

Note: We have performed multiple experiments and we received fluctuating values AUC. Below are the screenshots for different experiments conducted along with time.

```
Running epoch - > 133 out of total - > 150 epochs
Running epoch - > 134 out of total - > 150 epochs
Running epoch - > 135 out of total - > 150 epochs
Running epoch - > 136 out of total - > 150 epochs
Running epoch - > 137 out of total - > 150 epochs
Running epoch - > 138 out of total - > 150 epochs
Running epoch - > 139 out of total - > 150 epochs
Running epoch - > 140 out of total - > 150 epochs
Running epoch - > 141 out of total - > 150 epochs
Running epoch - > 142 out of total - > 150 epochs
Running epoch - > 143 out of total - > 150 epochs
Running epoch - > 144 out of total - > 150 epochs
Running epoch - > 145 out of total - > 150 epochs
Running epoch - > 146 out of total - > 150 epochs
Running epoch - > 147 out of total - > 150 epochs
Running epoch - > 148 out of total - > 150 epochs
Running epoch - > 149 out of total - > 150 epochs
Accuracy: 0.925
AUC: 0.596
```

In [64]: 1 expLog

Out[64]:

	ExpID	Train Time	Test Time	Accuracy	AUC	Comments
0	Multi Layer Perceptron	195.7175	0.0156	0.876953	0.520811	
1	Multi Layer Perceptron	222.3465	0.0326	0.895508	0.560529	
2	Multi Layer Perceptron	574.4512	0.0312	0.912109	0.555222	
3	Multi Layer Perceptron	349.5075	0.0282	0.913086	0.500000	ReLU
4	Multi Layer Perceptron	235.7002	0.0156	0.922000	0.500000	Sigmoid
5	Multi Layer Perceptron	351.6967	0.2815	0.924805	0.595612	ReLU + Sigmoid


```

Running epoch - > 82 out of total - > 100 epochs
Running epoch - > 83 out of total - > 100 epochs
Running epoch - > 84 out of total - > 100 epochs
Running epoch - > 85 out of total - > 100 epochs
Running epoch - > 86 out of total - > 100 epochs
Running epoch - > 87 out of total - > 100 epochs
Running epoch - > 88 out of total - > 100 epochs
Running epoch - > 89 out of total - > 100 epochs
Running epoch - > 90 out of total - > 100 epochs
Running epoch - > 91 out of total - > 100 epochs
Running epoch - > 92 out of total - > 100 epochs
Running epoch - > 93 out of total - > 100 epochs
Running epoch - > 94 out of total - > 100 epochs
Running epoch - > 95 out of total - > 100 epochs
Running epoch - > 96 out of total - > 100 epochs
Running epoch - > 97 out of total - > 100 epochs
Running epoch - > 98 out of total - > 100 epochs
Running epoch - > 99 out of total - > 100 epochs
i -> 593
Accuracy: 0.912
AUC: 0.563

```

```

1 %load_ext tensorboard
2 %tensorboard --logdir runs

```

```

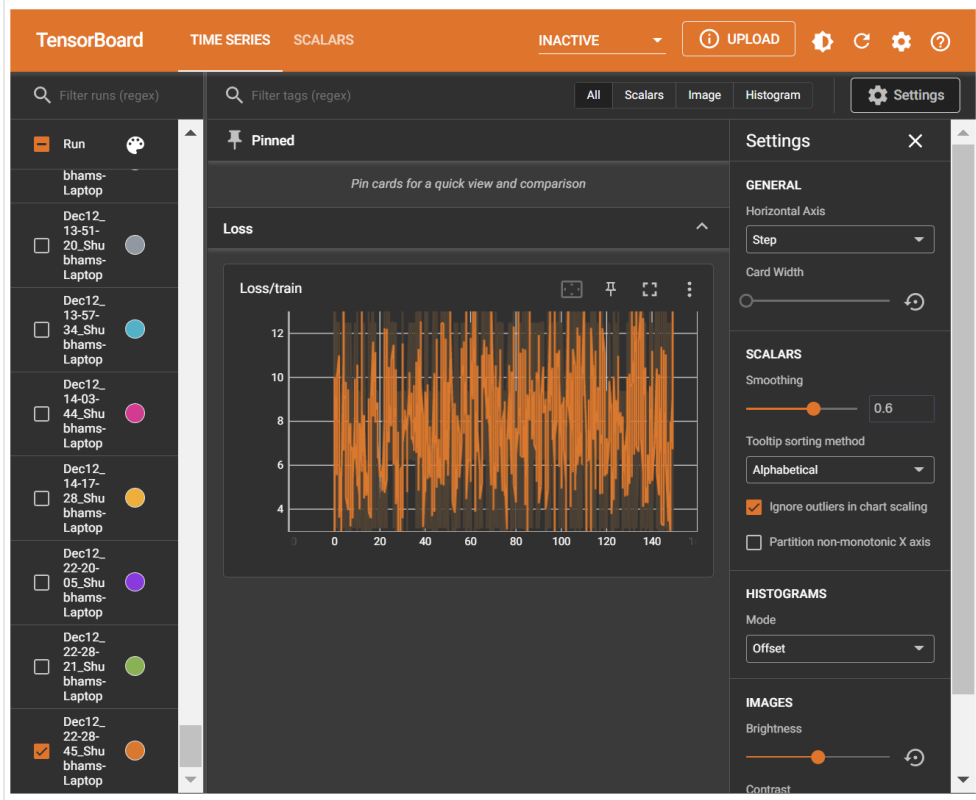
1 expLog

```

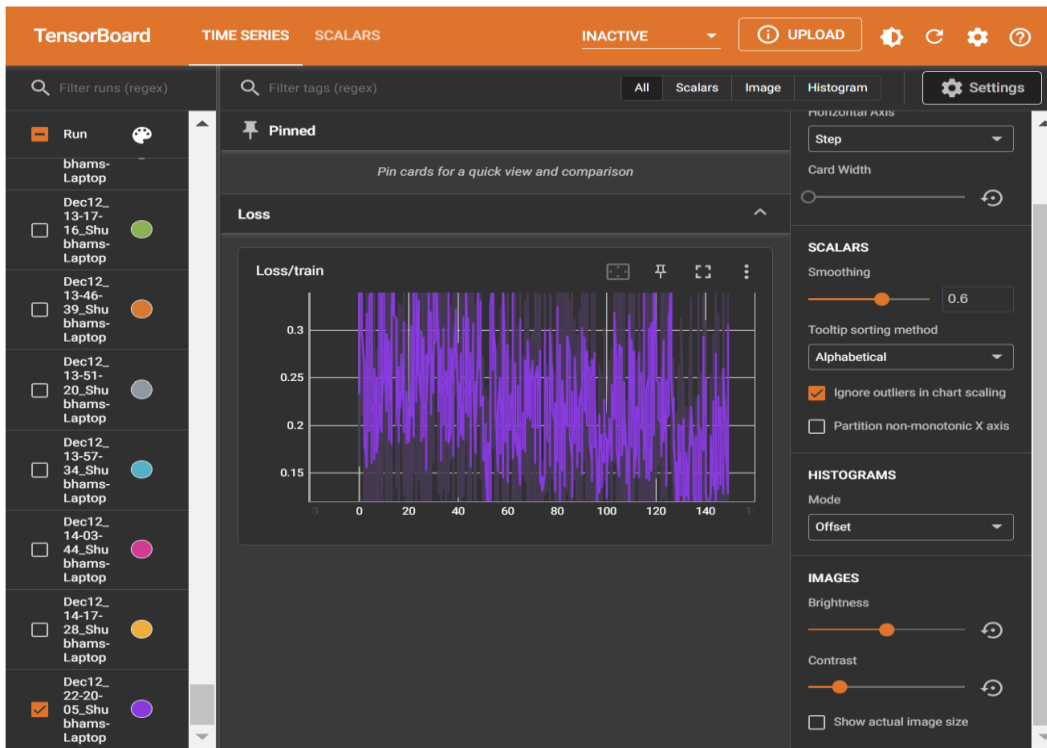
	ExpID	Train Time	Test Time	Accuracy	AUC	Comments
0	Multi Layer Perceptron	525.3236	0.0250	0.895508	0.512441	ReLU + Sigmoid
1	Multi Layer Perceptron	356.5367	0.0156	0.936523	0.500000	ReLU
2	Multi Layer Perceptron	232.7926	0.0312	0.921000	0.500000	Sigmoid

TensorBoard for Our Experiments:

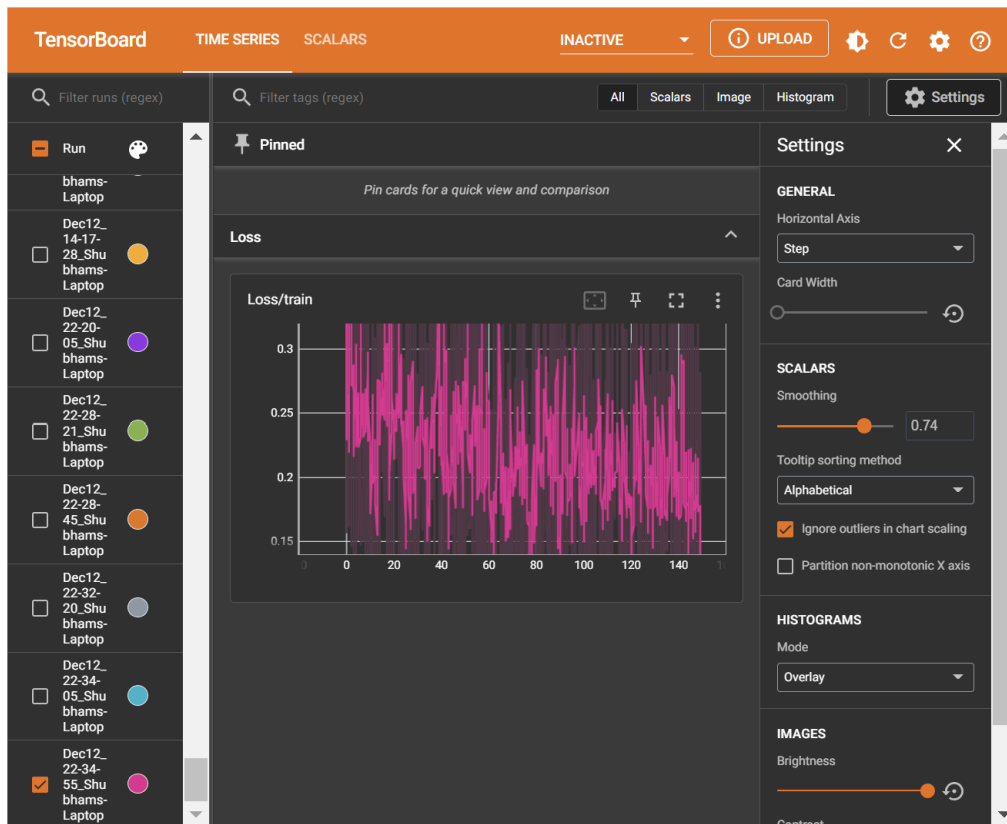
Experiment 1:



Experiment 2:



Experiment 3:



We have also performed MLP using in-built classifier.

1. Single Layer Perceptron.

```
1 from sklearn.neural_network import MLPClassifier
2 mlp_pipe = Pipeline([
3     ("data_prep", data_prep_pipe),
4     ('MLP', MLPClassifier(random_state=42))])
5
6 params = {'MLP__hidden_layer_sizes':[1],
7           'MLP__alpha':[0.1,0.5],
8           'MLP__activation':['identity','tanh','relu']}
9 sgp_gridsearch = GridSearchCV(mlp_pipe, param_grid = params, cv = 3, scoring='accuracy', n_jobs = -1)
```

In our Neural network we have used 1 linear hidden layer.

In the forward propagation we have experimented with different activation functions like relu, tanh, identity.

2. Multi-Layer Perceptron:

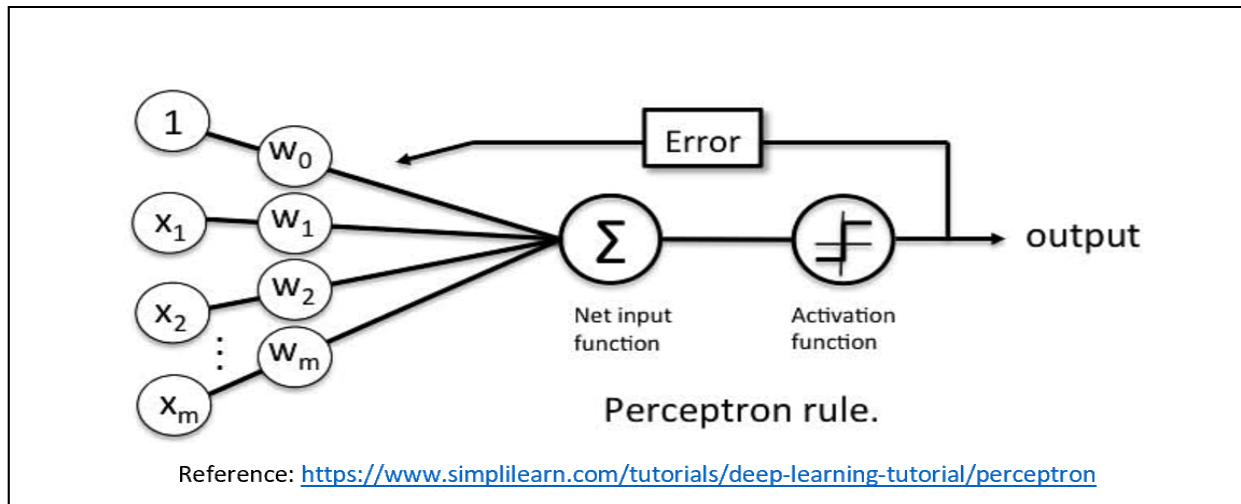
```
1 from sklearn.neural_network import MLPClassifier
2 mlp_pipe = Pipeline([
3     ("data_prep", data_prep_pipe),
4     ('MLP', MLPClassifier(random_state=42))])
5
6 params = {'MLP__hidden_layer_sizes':[5,10],
7           'MLP__alpha':[0.1,0.5],
8           'MLP__activation':['identity','tanh','relu']}
9 mlp_gridsearch = GridSearchCV(mlp_pipe, param_grid = params, cv = 3, scoring='accuracy', n_jobs = -1)
```

In our Neural network we have used 5 and 10 linear hidden layers.

In the forward propagation we have experimented with different activation functions like relu, tanh, identity.

Below are experiments conducted with in-built classifier.

	ExpID	Time	Accuracy	Valid Acc	AUC	Comment
0	MLP (Single hiddne layer)	15.8018	0.924275	0.909667	0.778271	MLP (Single hiddne layer)
1	MLP (multiple hidden layer)	15.8018	0.925275	0.909667	0.787373	MLP (Multiple hiddne layer)



Gap Analysis:

- In previous phases we performed different Experiments on different models and have the following results:
 - For our Kaggle submission we have Lasso as our best model with ROC AUC of (0.733)
 - As compared to the Lasso Regression, we observed a Gap in AUC of other models such as Logistic Regression, Decision Tree, and Random Forest.
 - We also observed our best model to be decision tree in terms of TEST accuracy.
- In Phase 4, we performed the Multi-Layer Perceptron Model and this gave us following results:
 - Testing AUC of (0.6) which is on lower side as compared to the previous phases.
 - More experimentation may lead to more effective deep learning models.

Leakage:

There is a slight leakage in our dataset which is explained below:

- The dataset is quite unbalanced, and as a result, our ML models are performing with excellent accuracy. Around 90 to 95 % haven't repaid their debt, while the remaining entries have, creating a significant imbalance in the TARGET feature, which essentially indicates whether an applicant has repaid loan or not.
- When undertaking feature engineering, we notice a significant difference between borrowers and borrowers who have paid back their loans. Because of this imbalance dataset, our model can accurately anticipate most entries on training dataset.
- Given that we are receiving higher AUC during MLP but lower AUC on Kaggle, it appears that there might be overfitting of the training data.

Result and Discussion:

In Phase 1, we finalized the machine learning models and different project tracking tools allocated tasks to team members built the Phase leader plan the credit assignment plan and Gantt chart. Along with this we also performed some basic data exploration and explored the dataset.

In phase 2 we performed the baseline models of all the selected models and found out the training and testing accuracies.

In Phase 3 we performed feature engineering and hyper parameter tuning and reran the hyperparameter tuned models. We added 2 new models the Lasso Regression and Ridge regression which gave us the improved ROC AUC of 75.57 % than the baseline models.

Some important finding in phase 3 were:

- The hyper-tuned decision tree model's train (92.19) and test (92.13) accuracy have greatly risen in comparison to the model's baseline, indicating that it is functioning well on the input dataset.
- The overall accuracy of the decision tree grew significantly and reached 92%. Hyper tuned Decision Tree is the best-fit algorithm since it surpasses others models in Phase 3.
- We saw an improvement of 6% in test accuracy and 6% in total accuracy. Decision tree outperforms the other models since its log loss (0.24) is on the lower side and has greatly lowered when compared to its baseline model.

In this Phase, we implemented deep learning, and our model was a multi-layer perceptron. We then generated a visualization of the loss function using Tensor Board to visualize our training model. We received a AUC score of 0.6 in our results. Our accuracy for the multi-Layer perceptron was out to be 92.4% which is quite efficient and overall good for the given data.

We have performed various experiments and below is the brief description:

We can conclude that our best models are Decision Tree with accuracy on higher side (92.03), ROC score of 53.58% and Lasso Regression with accuracy on a lower side but ROC score on a higher side giving us ROC score of 75.57%.

We also built a Deep learning model Multi-Layer Perceptron using PyTorch. The Test Accuracy was on a higher side but we were getting ROC score on the Lower side (60.13%) as compared to the best AUC found in phase 3.




Following are more details below:

1. Logistic Regression: In Phase 2 we performed, baseline logistic regression with limited feature engineering and it gave us the Train Accuracy of 92 % and test accuracy of 91.9 %. In phase 3 we performed feature engineering and after running it on multiple experiments it gave us the train Accuracy of 91.91% and test accuracy of 91.98%.
2. Random Forest: In Phase 2 we performed, baseline Random forest with limited feature engineering and it gave us the Train Accuracy of 92.2 % and test accuracy of 92.2%. In phase 3 we performed feature engineering and after running it on multiple experiments it gave us the train Accuracy of 91.91% and test accuracy of 91.98%.

3. **Decision Tree:** In Phase 2 we performed, baseline Decision tree with limited feature engineering and it gave us the Train Accuracy of 86.4 % and test accuracy of 86.1 %. In phase 3 we performed feature engineering and after running it on multiple experiments it gave us the train Accuracy of 92.19% and test accuracy of 92.13%.
4. **Lasso Regression:** In Phase 3 after hyper-parameter tuning and addition feature engineering it gave us training and testing accuracy on a lower side but ROC on a higher side of 75.57%.
5. **Ridge Regression:** In Phase 3 after hyper-parameter tuning and addition feature engineering it gave us training and testing accuracy on a lower side but ROC on a higher side of 75.68%.
6. **Deep Learning Model:** In Phase 4 we built a Multi-layer perceptron model and experimented with different types of activation function like RuLU, Sigmoid etc. along with different epochs. Our best testing accuracy was 93.65 % along with ROC of 60.13%

Below is our best submission from each phase 2 ,3, 4

Figure 24: Kaggle Submission

Submissions			
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.			
<div> <div></div> Submissions evaluated for final score </div>			
<div> <div>All</div> <div>Successful</div> <div>Selected</div> <div>Errors</div> </div>		Recent ▾	
Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
<div>  Submission4.csv Complete (after deadline) · now </div>	0.50593	0.50461	<input type="checkbox"/>
<div>  Submission3.csv Complete (after deadline) · 1s ago </div>	0.71673	0.73086	<input type="checkbox"/>
<div>  Submission2.csv Complete (after deadline) · 1m ago </div>	0.50195	0.50271	<input type="checkbox"/>

Conclusion

The major objective of this project is to develop a machine learning model that can forecast a loan applicant's ability to repay a loan. Without any statistical analysis, many deserving applicants with no credit history or default history are accepted. The HCDR dataset is used in our work to train the machine learning model. A user's average, minimum, and maximum balances as well as reported Bureau scores, salary, and other factors are used to generate a credit history, which serves as a gauge of their reliability.

We improved our classification model through feature engineering, feature selection, and hyperparameter tuning to more precisely forecast whether the loan applicant will be able to repay the loan or not. As part of this project, we developed machine learning pipelines, undertaking exploratory data analysis on the datasets provided by Kaggle, and evaluating the models using several evaluation measures before deploying one.

During Phase 2 we concluded that the best model for this dataset will be logistic regression having the highest accuracy of 91.9 %. In Phase 3, we expanded our feature engineering and added some more relevant features like AMT Credit to Annuity Ratio and Annuity to Salary Ratio. On top of that we discarded features with more than 30 % null values as compared to phase 2. Because of hyperparameter tuning our accuracy and AUC both increased significantly. Decision tree turned out the best model in phase 3 along with the best parameters as gini criterion, max_depth = 4 and min samples leaf = 4. We achieved accuracy of 92.13 % by decision tree. Along with that, Lasso also turned out to be best model in terms of AUC 0.71 on kaggle.

As part of phase 4, we implemented Multi-Layer Perceptron with activation function ReLU and Sigmoid along with multiple hidden layer settings, we got the result that ReLU and Sigmoid together can give us the effective result. We have determined that the Multi-Layer Perceptron's accuracy for the provided dataset using PyTorch was 93.65%, which is extremely effective and generally good. The test AUC for MLP has been fluctuating and has come down to 0.6 after performing multiple experiments.

With all the experiments taken into consideration Decision Tree and Lasso Regression turned out to be the best performing models. Additional experimentation on MLP model could have yielded better results.

Bibliography:

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition

- by Aurélien Géron

Lab-End_to_end_Machine_Learning_Project

- by James Shahnan