Credit Hazard Forecasting

Piyush Gade
Department of *Computer Engineering*
San Jose State University

Divya sri Medarametla
Department of *Computer Engineering*
San Jose State University

Mounish Juvvadi
Department of *Computer Engineering*
San Jose State University

***Abstract*— Credit failures cause banks to suffer substantial financial losses, and in the end, it is the average borrower who pays the price for this mistake. Credit risk analysis is a tool that banks use to make sure they are lending money to trustworthy customers. Credit risk is the possibility that the borrower may not be able to repay the loan due to missing debt installments. The laborious process of having a person evaluate various credit criteria and conditions is reduced by applying machine learning to analyze credit risk. It also eliminates the human aspect, which could result in fraud and inaccurate calculations. Along with the other variables in the dataset, the individual's past banking behavior has an impact on the choice. This research will help us develop a model that can forecast whether or not a person will be given credit.**

INTRODUCTION

Consumer and business loans are the main source of funding for financial institutions like banks. Despite the fact that banks currently provide a variety of additional products, including equity funds, insurance, and so forth, loans remain their main source of revenue. The way banks generate revenue is through loans. In addition, banks suffer large losses if some of these borrowers fail to make their loan repayments. For their loan offerings, banks must therefore make a distinction between "good" and "bad" consumers. When it comes to consumer lending, a "good" customer is someone who has a solid credit history, a low risk of default, and is thus a good bet for the bank to extend the loan to. Contrarily, a "bad" client has high threat of default The bank's loans have a risk of default to a certain extent. Loan default costs the bank money and, in most situations, causes a significant loss. If these defaulters rise, the bank's financial stability may be in danger, particularly in a difficult economic environment. Therefore, reducing the amount of these defaulted loans is one of the bank's goals. Banks can lower the number of defaulters by tightening their lending standards, but doing so will also affect the total amount of loans made and their profit. On the other hand, if banks are lax, losses incurred in absorbing default loans may have an adverse effect on their profitability. The choice regarding customer assessment and risk profiling is highly important. Hence, Banks used to invest a lot of time and energy in studying their clients to distinguish between "good" and "bad" clients based on their financial and demographic histories. For them to be profitable, it is crucial to evaluate consumer credit default risk. Such judgments have historically been made by knowledgeable employees after gathering data on consumers and consulting them. Subjective evaluation is no longer practical due to the rise in loan volume and greater competition. Western banks created credit scoring as a way to gauge a loan applicant's level of financial risk. Based on the consumer's financial state, history, and recent transactions, a single score is generated. Risky consumers who are more prone to default are linked to lower scores. Sophisticated statistical and probability models have been developed over the decades and used extensively in many financial institutes. Due to tremendous progress in

computing power and algorithms, this process has largely been automated. Many banks can now make loan decisions within hours which used to take weeks before. Such fast turnaround time can benefit both the banks and the consumers.

Dataset Description

The dataset we are using has data for 32,581 borrowers and 12 features related to each of them. Below listed are the features.

Numerical Variables:
person_age, person_income, person_emp_length, loan_amnt, loan_int_rate, loan_percent_income, cb_preson_cred_hist_length.

Categorical Variables:
person_home_ownership, loan_intent, loan_grade, cb_person_default_on_file, loan_status.
Target variable: loan_status is the target variable used for prediction.

Source : https://www.kaggle.com/datasets/laotse/credit-risk-dataset

I. LITERATURE REVIEW AND RELATED WORK:

- Credit Risk Analysis Using Machine Learning Techniques

Banks and other financial institutions put a lot of work into finding borrowers who are creditworthy. The performance of various categorization algorithms used on a collection of actual data for consumer lending is examined in this work. The effectiveness of different classifier types (such as logistic regression, random forest, and neural networks) is assessed and contrasted. To choose the optimal features and get rid of outliers for classification, the data set is first pre-processed. Several criteria, including as classification accuracy, precision, recall, and specificity, are used to evaluate performance. It is discovered that logistic regression produces very good results. It is feasible to trade off precision and recall to screen those customers who are credit-worthy by adjusting the random forest's cut off ratio.

- Credit Risk Analysis and Prediction Modelling of Bank Loans Using R

Banks have access to enormous amounts of information on customer behavior, yet they are unable to determine whether an applicant is likely to fail or not. The goal of this study is to create a model and prototype it using data from the UCI repository. The model is a classification model based on decision trees that makes use of R Package functionalities. The dataset is pre-processed, reduced, and prepared to produce accurate predictions before the model is built. The test dataset is used to predict with the finished model, and the experimental findings demonstrate the effectiveness of the model's construction.

- Research on Credit Risk Evaluation of Commercial Banks Based on Artificial Neural Network Model

This article uses an artificial neural network model to primarily forecast the credit risk of corporate clients of commercial banks. First, this article chooses 14 financial indicators to build a credit risk evaluation index system based on the findings of prior studies; second, combined with cluster analysis and factor analysis to determine the actual credit rating of the sample data, thereby giving the sample data the category label; third, combining the aforementioned research, established and compared two conventional credit risk prediction models and three widely used Artificial Neural Network models.

- Big data and credit risk assessment: a bibliometric review, current streams, and directions for future research

This study used bibliometric analysis to follow the structural evolution of academic research on big data and credit risk assessment. All research with citations published between 2012 and 2021 are included in the bibliography, which was compiled using the Scopus database. According to the study's conclusions, big data and credit risk assessment are both extremely broad domains that have grown considerably over the past nine years. The majority of the documents' contributors were Chinese researchers and organizations. The current study comes to the conclusion that there are numerous ways to advance our understanding of credit risk assessment and big data.

- Credit Risk Measurement, Decision Analysis, Transformation and Upgrading for Financial Big Data

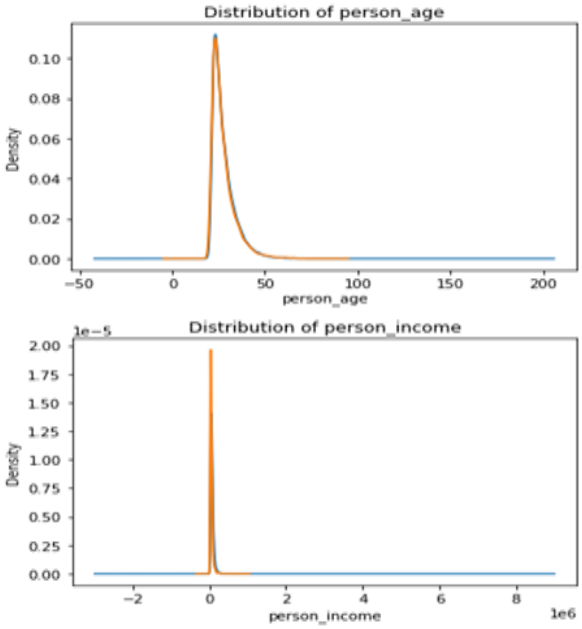
Credit risk measurement and decision analysis for financial big data lack well-developed theories, and there is no efficient system for their scientific evaluation. Reviewing them can help in gaining an awareness of the aforementioned subjects, as well as present difficulties, research challenges, and future research prospects. In addition, this article identifies four areas for future credit risk measurement and financial big data decision analysis study. For practitioners, researchers, financial institutions, and government agencies interested in complicated decision-making with big data, this study can also offer some guidance, recommendations, and insights.

Data Exploration and Visualization:

We performed univariate and bivariate analysis to Explore and visualize the data

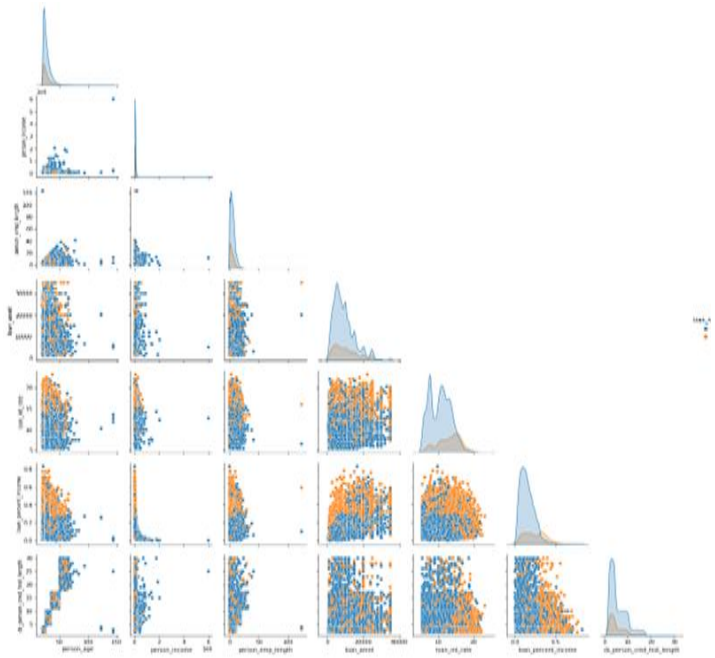
UNIVARIATE ANALYSIS ON NUMERICAL FEATURES:

We performed univariate analysis on all the numerical features in the dataset making data easier to interpret and to understand how data is distributed within a sample being studied.



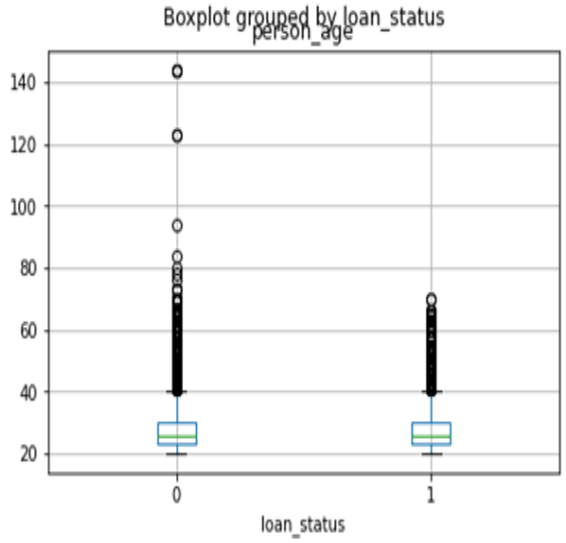
BIVARIATE ANALYSIS ON NUMERICAL FEATURES:

And then performed Bivariate analysis on all the numerical features in the dataset by using pairplot graph



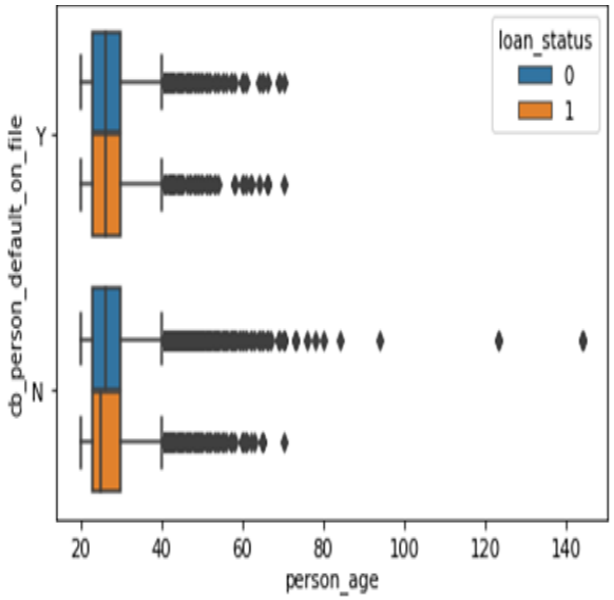
BOXPLOT ON NUMERICAL FEATURES:

We also performed a box plot visualization technique to quickly identify mean values, the dispersion of the data set, and signs of skewness.



BOXPLOT ON CATEGORICAL FEATURES:

We also explored the categorical features by comparing across different categorical variables or identifying outliers, if either of those exist in a dataset



DATA PREPROCESSING:

ELIMINATING NULL VALUES:

On Numerical data

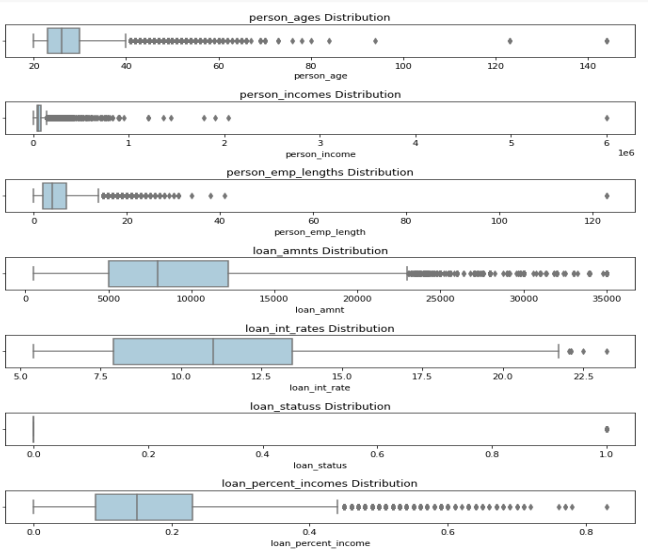
- Removed all the redundant data.
- Dropped the null values using the .isnull().sum() function.
- Identifying missing data columns,and filling them with Mean/Median

On Categorical data

- Filling missing values with Mode
- Checking for any random text with in features,and discarding them
- Checking the value counts of each categorical feature
- Encoding features using One-Hot-Encoder/Label-Encoder/Multi Label Binarizer

OUTLIER TREATMENT:

We Eliminated outliers by checking how much data points are deviated away from mean/median.



ENCODING CATEGORICAL FEATURES:

By label encoding the categorical features cb_person_default_on_file, loan_grade, loan_intent, person_home_ownership, we have built additional features. One hot encoded feature is produced using the new label encoded features. The initial categorization features are removed, and a total of 19 one-hot encoded features are built on top of the label encoded features. There are now 31 features in all.

TRAIN TEST SPLIT:

The independent variables are divided into train and test sets in a ratio of 5:1, and the target variable loan status is removed from the dataset.

STANDARDIZATION:

As a preprocessing, all the features are standardized. This method is used specifically with PCA in mind.

DEALING WITH CLASS IMBALANCE:

There is a significant class gap between the courses. After eliminating outliers and null valued rows, there are 21k class 0 points and 5.5k class 1 points. We made the decision to test both under- and over-sampling. We applied the Neighborhood cleanup rule for undersampling and SMOTE for oversampling.

DOWNSAMPLING WITH NEIGHBORHOOD CLEANING RULE:

The Neighborhood Cleaning Rule is an undersampling technique that eliminates redundant samples as well as examples that are noisy or unclear. The samples that are incorrectly classified by a KNN classifier are eliminated in one step. Less examples that are redundant are eliminated, and greater emphasis is made on purifying the examples that are saved. Here is the dataset before and after resampling.

```
Before UnderSampling, the shape of train_X: (21296, 30)
Before UnderSampling, the shape of train_y: (21296,)

After UnderSampling, the shape of train_X: (14791, 30)
After UnderSampling, the shape of train_y: (14791,)

Counts of label '0' - Before UnderSampling:16808, After UnderSampling: 10303
Counts of label '1' - Before UnderSampling:4488, After UnderSampling: 4488
```

OVERSAMPLING WITH SMOTE:

SMOTE selects examples in the feature space that are close to one another, draws a line between the examples, and then creates a new sample at a point along the line.For example, a random representative from the minority class is initially picked. Next, k nearest neighbors for that example are located. A synthetic example is made at a randomly chosen position in feature space between two instances and their randomly chosen neighbor. Here is the dataset before and after resampling.

```
Before OverSampling, the shape of train_X: (21296, 30)
Before OverSampling, the shape of train_y: (21296,)

After OverSampling, the shape of train_X: (33616, 30)
After OverSampling, the shape of train_y: (33616,)

Counts of label '0' - Before Oversampling:16808, After OverSampling: 16808
Counts of label '1' - Before Oversampling:4488, After OverSampling: 16808
```

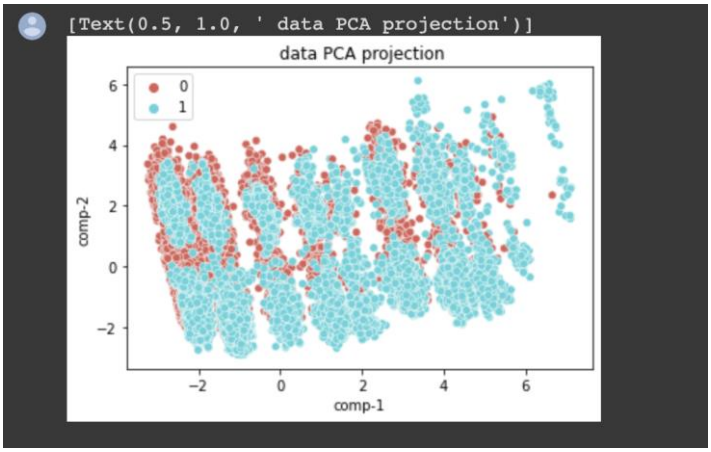
DIMENSIONALITY REDUCTION:

Principal Component Analysis:

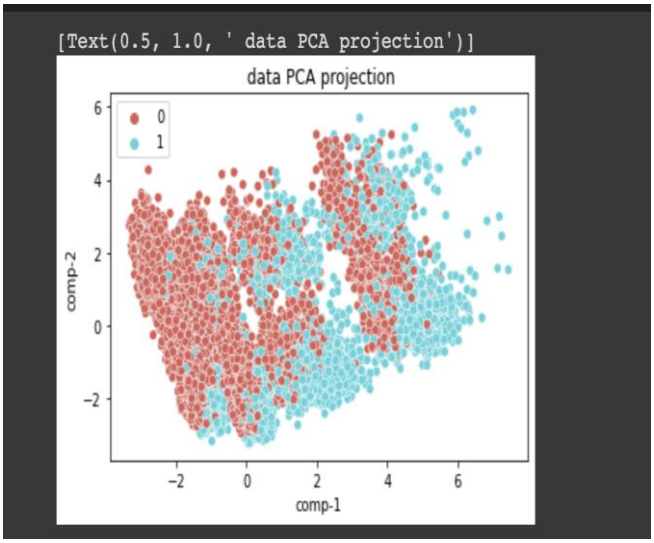
From 30 features, 19 features are found to preserve 99% of variance. PCA is experimented upon undersampled, oversampled and without resampling data.

Principal Component Analysis for data visualization:

Upon using 2 components for visualization of dataset, here is the results:
PCA on oversampled data:



PCA on undersampled data:



FEATURE SELECTION:

The Recursive Feature Elimination method with a Random Forest estimator has been applied. This is due to the fact that random forest tree-based techniques are inherently ordered by the extent to which they maintain node purity. The raw dataset initially contains 30 features, of which the best 20 are chosen for further modeling using the RFE function. These 30 features are fed into the RF model. Using all of the features in the training dataset as a starting point, RFE attempts to find a subset of features by successfully eliminating features one at a time until the desired number of features is left. This is accomplished by first fitting the base model's machine learning algorithm, ranking the features according to relevance, eliminating the least important features, and then re-fitting the model. Up until a

certain amount of features are still present, this process is repeated.

MODELS:

We have used the below four models to predict the creditworthiness of the customer.

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Dense Neural Networks

We have experimented each model from above with 7 dataset samples, namely Without resampling, without resampling and PCA, Undersampled data, undersampled data with PCA, oversampled data, oversampled data with PCA and selected features from feature extractor.

We performed GridSearchCV on each of the model variations to find out the best performance with each model. We used a stratified split for cv with 5 splits in order to maintain the class labels proportion in each split. We made the code modular and reusable by implementing functions for each of the reused code blocks.

We verified the best model for each class of algorithms with training score as well in order to check for the overfitting and chose the model that is not overfitting.
ROC curves for all the best models found out from gridsearchCV were plotted to understand the performance of the model on each class.

METRIC:

We believe that Recall score on class 1 is the metric to be optimized for, since all the class 1 i.e people who default on loan needs to be correctly classified as the misclassifying a true class 1 point can be very costly for the bank. Hence finally all the models are compared on class 1 recall score. We maintained a dictionary for best recall class 1 score for each model in order to get a final conclusion.

1. LOGISTIC REGRESSION -

As a baseline model, we initially used a linear model. Assuming that the data can be separated into classes in a linear fashion, the linear model of logistic regression seeks to fit a linear surface as the decision boundary. It naturally encourages the binary classification issue. Additionally, it automatically forecasts the likelihood of each class for a specific location.

We implemented logistic regression using the Sklearn library. We used c value as the hyperparameter in logarithmic space from 100 to 0.01. We used saga solver out of all the solvers provided by SkLearn because saga is the only solver which supports both L1 and L2 formulation of logistic regression. Hence all the c values are searched with both L1 and L2 penalties.

After running Logistic regression on all the 7 dataset samples, we found that Logistic Regression with neighborhood cleaning rule undersampling and PCA gave better Recall score for Class 1.

```
{'LR1': 0.5471180237877402,
 'LR2': 0.27813357731015553,
 'LR3': 0.7319304666056725,
 'LR4': 0.7392497712717292,
 'LR5': 0.7657822506861848,
 'LR6': 0.7685269899359561,
 'LR7': 0.5425434583714547}

Logistic Regression with neighborhood cleaning rule undersampling and PCA gave better Recall score for Class 1
```

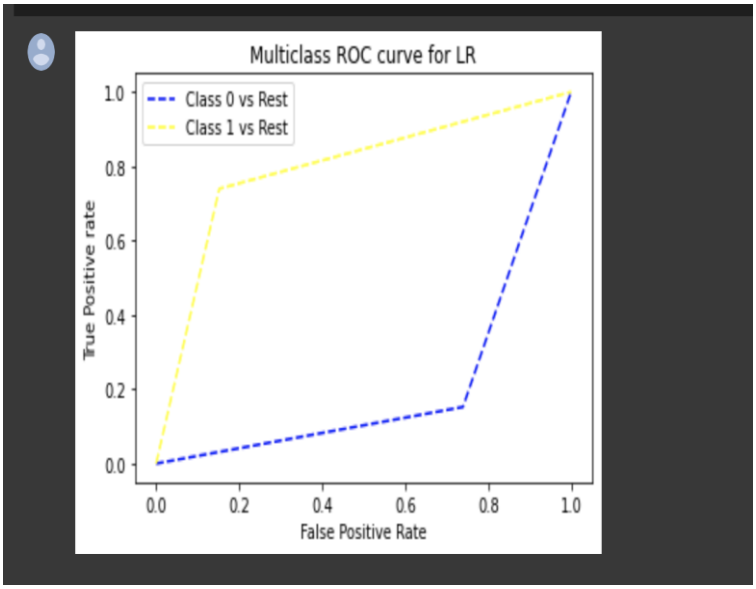
It gave a class 1 recall score of 0.769. It's Cross validation scores:

```
Best: 0.880468 using {'C': 100, 'penalty': 'l2', 'solver': 'saga'}
0.880468 (0.002876) with: {'C': 100, 'penalty': 'l2', 'solver': 'saga'}
0.880468 (0.002876) with: {'C': 100, 'penalty': 'l1', 'solver': 'saga'}
0.880468 (0.002876) with: {'C': 10, 'penalty': 'l2', 'solver': 'saga'}
0.880468 (0.002876) with: {'C': 10, 'penalty': 'l1', 'solver': 'saga'}
0.880401 (0.002835) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'saga'}
0.880265 (0.002770) with: {'C': 1.0, 'penalty': 'l1', 'solver': 'saga'}
0.880198 (0.002779) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'saga'}
0.880198 (0.002022) with: {'C': 0.1, 'penalty': 'l1', 'solver': 'saga'}
0.880198 (0.002194) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'saga'}
0.875533 (0.003092) with: {'C': 0.01, 'penalty': 'l1', 'solver': 'saga'}
```

It's training score :

```
training class 1 recall Score: 0.7375222816399287
```

ROC curve for the model:

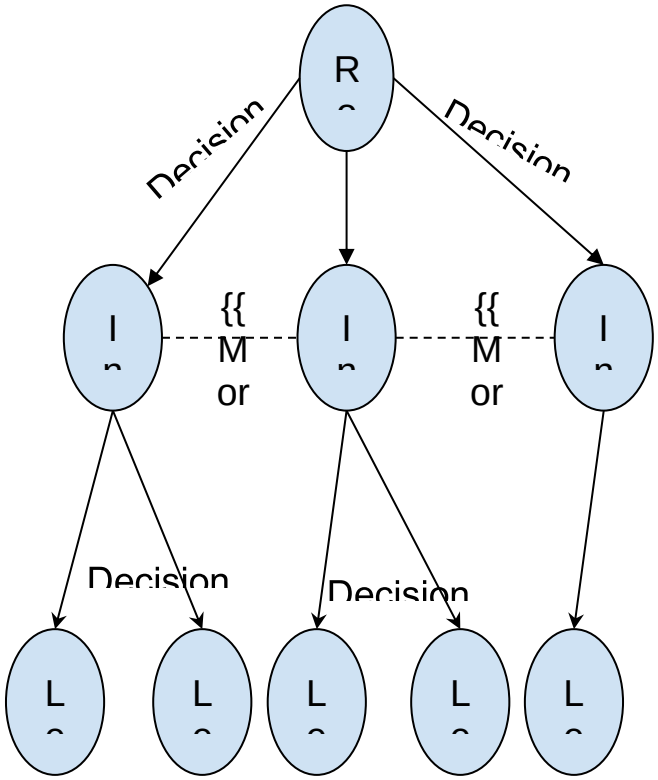


Now we tried tree based models and neural networks expecting a better class 1 recall score.

2. DECISION TREE -

A supervised machine learning algorithm called a decision tree employs a set of rules to arrive at decisions. The idea behind Decision Trees is to repeatedly partition the dataset until all the data points that belong to each class are isolated by using the dataset characteristics. With this procedure, the data is arranged in a tree form. You add a node to the tree each time you make a decision , where the root node is the first node of the tree. Based on this decision the dataset divides according to the importance of a characteristic and adds additional nodes. The last nodes produced are referred to as leaf nodes if you choose to end the process after a split.

Decision Trees employ loss functions that assess the split in light of the nodes' degree of purity. We have chosen entropy as our loss function in this project. The amount of impurity or uncertainty in a set of data is measured by entropy.
Structure of Decision Tree:



We have trained the Decision Tree (GridSearchCV) without resampling , without resampling and pca, with undersampling, with undersampling and pca, with upsampling,with upsampling and pca, with selected Features.

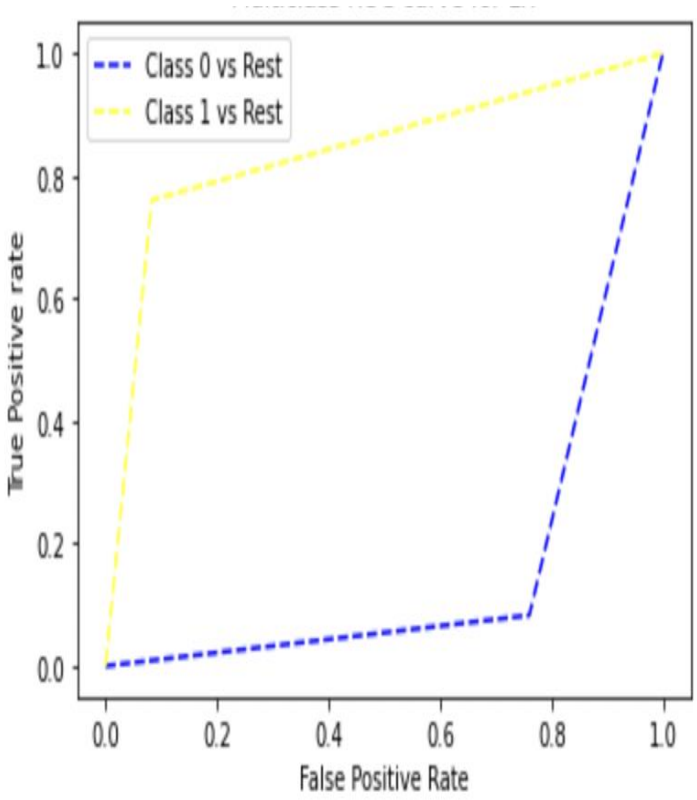
A method for locating the ideal parameter values in a grid from a collection of parameters is called GridSearchCV. In essence, it is a cross-validation method.
Here are the hyperparameters for the DT.

```
def DT(x_train, y_train):
    model = DecisionTreeClassifier(criterion="entropy")
    max_depth = [3, 4, 5, 6, 7, 8, 9]
    min_samples_split = [4, 10, 20]
    # define grid search
    grid = dict(min_samples_split=min_samples_split,max_depth=max_depth)
    grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=5)
    grid_result = grid_search.fit(x_train, y_train)
    return grid_result
```

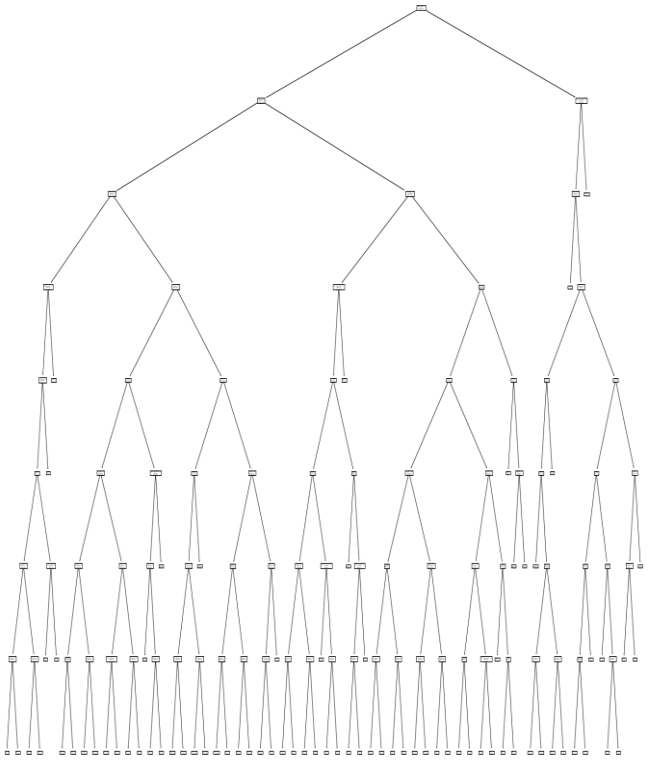
II. Test Results:
{'Decision Tree without resampling': 0.6999085086916743,
'Decision Tree without resampling and pca':
0.2799634034766697,
'Decision Tree with undersampling': 0.7602927721866423,
'Decision Tree with undersampling and pca':
0.7200365965233303,
'Decision Tree with upsampling': 0.7365050320219579,
'Decision Tree with upsampling and pca':
0.6999085086916743,
'Decision Tree with selected Features':
0.6999085086916743}

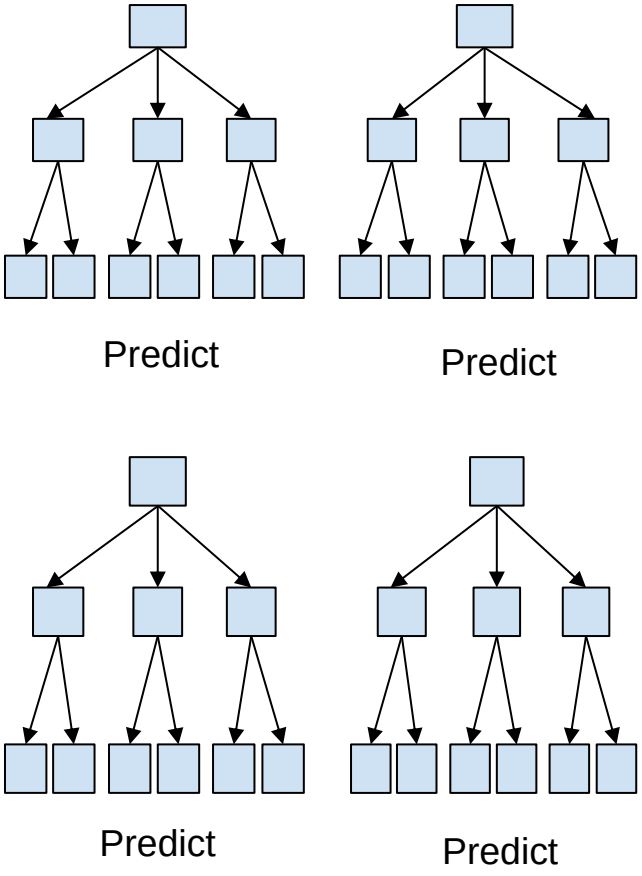
From these results we have observed that 'Decision Tree with undersampling' got the best results out of all with the best parameters being **{'max_depth': 9, 'min_samples_split': 20}**

ROC Curve for the Decision Tree with undersampling



Decision Tree Output –





Looking at the above random forest the number of predictions of "Yes" are more than "No", therefore the overall prediction will be "Yes".

3. RANDOM FOREST:

A huge number of distinct decision trees work together as an ensemble in a random forest. Every single tree in the random forest spits out a class prediction, and the classification that receives the most votes becomes the prediction made by our model.

Random forest Uses bagging technique(Bootstrap aggregation). Random sample is taken from the original dataset with replacement and a decision tree is built out of it and then the final output is based on majority voting after combining the results of all models.

Similar to the decision tree we have trained random forest (GridSearchCV) without resampling , without resampling and pca, with undersampling, with undersampling and pca, with upsampling, with upsampling and pca, with selected Features. Here are the hyperparameters:

```
def RF(x_train, y_train):
    model = RandomForestClassifier()
    n_estimators= [ 10, 50, 100]
    max_depth = [3, 4, 5, 6, 7]
    min_samples_split = [4, 10, 20]
    # define grid search
    grid = dict(n_estimators=n_estimators,min_samples_split=min_samples_split,max_depth=max_depth)
    grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=5)
    grid_result = grid_search.fit(x_train, y_train)
    return grid_result
```

TEST RESULTS:

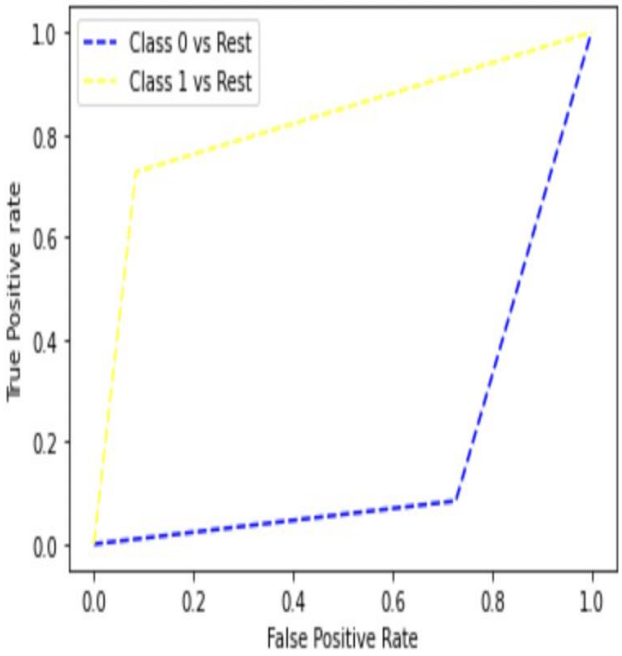
Below are the test results that are obtained.

{'Random Forest without resampling':
0.6559926806953339,
Random Forest without resampling and pca:
0.1354071363220494,
Random Forest with undersampling: 0.727355901189387,

Random Forest with undersampling and pca: 0.7035681610247027,
Random Forest with upsampling: 0.7319304666056725,
Random Forest with upsampling and pca: 0.727355901189387,
Random Forest with selected Features: 0.6843549862763038}

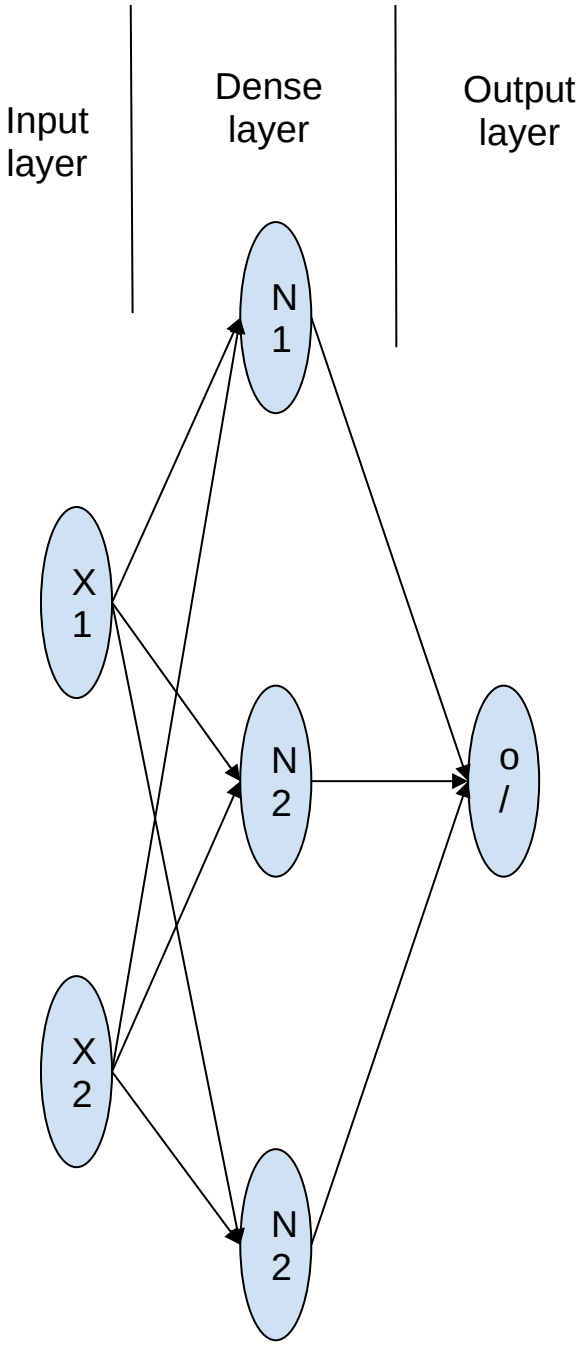
From the above results we have observed that the best score obtained is 0.72 which is the same for both Random Forest with undersampling and Random Forest with upsampling and pca with the best parameters being - **'max_depth': 7, 'min_samples_split': 10, 'n_estimators': 50**

ROC CURVE:



3. DENSE NEURAL NETWORKS:

The frequently utilized layer in neural networks is the keras dense layer. Because the dense layer is a neural network layer that is closely linked, each neuron in it receives input from every neuron in the layer below it. In the backdrop, the thick layer multiplies matrices and vectors. Backpropagation can be used to train and update the parameters that make up the matrix values. The dense layer produces a "n" dimensional vector. As a result, the primary function of a dense layer is to change the dimensions of the vector. Dense layers may also execute vector operations like rotation, scaling, and translation.



The above is a simple structure of a dense neural network consisting of two input X1 and X2 and three neurons N1,N2,N3 and one output layer o/p.

```
batch_size = 256
epochs = 30
np.random.seed(5)

model = Sequential()

model.add(Dense(input_shape= (X_train.shape[1],), units = 150, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(units = 75, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(units = 25, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))

model.add(Dense(units=1, kernel_initializer='uniform', activation='sigmoid'))
model.build()
model.compile(optimizer='Adam', loss='binary_crossentropy', metrics=[tf.keras.metrics.Recall(thresho
model.summary()

early_stops = EarlyStopping(patience=10, monitor='val_acc')
mc = ModelCheckpoint('best_model.h5', monitor='val_loss', verbose=0, save_best_only=True)
model.fit(X_train, y_train, callbacks=[early_stops, mc], batch_size= batch_size, epochs= epochs, verb
```

Similarly, we have created a dense neural network with three dense layers with 150, 75 and 25 neurons and one output layer with one neuron. It is preferable to have a decreasing learning rate during training in order to effectively approach the global minimum of the loss function.

We have used Rectified Linear Unit (ReLU) as activation function in the dense layer and sigmoid function in the output layer. ReLU will output the input directly if it is positive, otherwise, it will output zero and sigmoid will output the value between 0 to 1.

Additionally used Batch normalization at every dense layer to normalize the inputs to the layers by re-centering and re-scaling, which made training quicker and more stable. Also used dropout at 20% in order to overcome the overfitting. Initialized the model with uniform weights using kernel_ initializer and used adam as the optimizing function.

After creating the above model trained it for 30 epochs and below are the test results

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_20 (Dense)             (None, 150)               4650

batch_normalization_15 (Bat  (None, 150)               600
chNormalization)

dropout_15 (Dropout)         (None, 150)               0

dense_21 (Dense)             (None, 75)                11325

batch_normalization_16 (Bat  (None, 75)                300
chNormalization)

dropout_16 (Dropout)         (None, 75)                0

dense_22 (Dense)             (None, 25)                1900

batch_normalization_17 (Bat  (None, 25)                100
chNormalization)

dropout_17 (Dropout)         (None, 25)                0

dense_23 (Dense)             (None, 1)                 26

=================================================================
Total params: 18,901
Trainable params: 18,401
Non-trainable params: 500
_________________________________________________________________
Epoch 1/30
131/132 [============================>.] - ETA: 0s - loss: 0.4708 - recall_4: 0.9946WARNING:tensorflow:Early stopping conditioned on metric `val_acc` which i
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
```

TEST RESULTS:

DNN without resampling - 0.8929551692589204
DNN without resampling and pca - 0.6660567246111619
DNN with undersampling - 0.9258920402561757
DNN with downsampling and pca - 0.9268069533394327
DNN with upsampling - 0.9551692589204026

From the above results we have observed that DNN with upsampling got the best results.
Confusion matrix and classification report of the best model :

```
ROC_AUC of the test set: 0.9125549818217518

Confusion_metrix :
 [[2015 2216]
 [  49 1044]]
Precision :  0.3202453987730061
Recall :  0.9551692589204026
F1-score : 0.47966919365954513
```

RESULTS AND COMPARISON:

Among all the models we have trained and tested we have observed that the Dense neural network has got the best class 1 recall scores.

Model	Recall on Test set
Logistic Regression	73.9%
Decision Tree	76%
Random Forest	73.1%
Dense Neural Network	95.5%

CONCLUSION & FUTURE SCOPE:

For banks, predicting credit risk is a significant issue. For the banks, predicting loan defaulters might result in significant cost savings. Therefore, this is a problem in which the banks can invest. On the basis of the dataset used, we created binary classifiers and discovered that the top model can correctly classify loan defaulters 95 out of 100 times. As a result, this model can be used in real time.

Future work in this direction can be concentrated on selecting more data samples, utilizing additional data resampling methods to address class imbalance, utilizing additional imputing methods to address missing data, and experimenting with additional models, such as the more complex deep learning models XG Boost.

REFERENCES:

Using Decision Tree Classification Algorithm to Design and Construct the Credit Rating Model for Banking Customers.

RESEARCH ON CREDIT RISK EVALUATION OF COMMERCIAL BANKS BASED ON ARTIFICIAL NEURAL NETWORK MODEL

Big data and credit risk assessment: a bibliometric review, current streams, and directions for future research

Credit Risk Measurement, Decision Analysis, Transformation and Upgrading for Financial Big Data

Weighted Random Forests for Evaluating Financial Credit Risk.

https://towardsdatascience.com/credit-risk-modeling-with-machine-learning-8c8a2657b4c4

INDIVIDUAL CONTRIBUTION:

Divya sri Medarametla:

Dataset Description, Reading the data Univariate Analysis of Continuous Variables, Bivariate Analysis of numerical Features, Dealing with Null valued rows and outliers and PCA with upsample and no resample data

Piyush Gade:

Categorical variable encoding, Splitting into train and test, Column standardization
Feature selection, Oversampling and undersampling to deal with class imbalance, PCA on Oversampled data, Functions for metrics and roc and grid search results and Logistic regression with grid search cv on seven variations of dataset

Mounish Juvvadi:

Decision tree with grid search cv on seven variations of dataset, Random Forest with grid search cv on seven variations of dataset,
Dense neural networks on seven variations of dataset and final results comparision.