LOAN RISK PREDICTION

SUBMITTED BY

PREETHI B **(21MX118)**

SREE VARSHINI B **(21MX225)**

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY

OCTOBER 2022

DEPARTMENT OF COMPUTER APPLICATIONS

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE-641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE-641 004

20MX37 - Mini Project

LOAN RISK PREDICTION

Bonafide record of work done by

PREETHI B (21MX118)

SREE VARSHINI B (21MX225)

Dissertation submitted in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY

OCTOBER 2022

Faculty Guide

……

Dr.Vijayalakshmi Pai G A

CONTENTS

ACKNOWLEDGEMENT

We indulged to thank **Dr. K. PRAKASAN**, Principal, PSG College of Technology, for providing us with all the necessary facilities and motivation to carry out the project.

We extend our earnest gratitude to **Dr. A. CHITRA,** Professor and Head, Department of Computer Applications, PSG College of Technology, Coimbatore, for her support, encouragement, and, above all, her ardent motivation.

We are grateful to thank **Dr. R. MANAVALAN**, Associate Professor, Program Coordinator, Department of Computer Applications, for his full- fledged support and guidance.

We would like to express our gratitude to our guide, **Dr. VIJAYALAKSHMI PAI G A,** Professor (CAS) , Department of Computer Applications, PSG College of Technology, for his valuable suggestions and guidance in the completion of the project.

We also express our gratefulness to all the faculty members of the Department of Computer Applications, PSG College of Technology, and our friends for their encouragement and support

SYNOPSIS

Credit risk is the possibility of a loss resulting from a borrower's failure to repay a loan or meet contractual obligations. There is a chance that the borrower won't be able to repay the debt when lenders give mortgages, credit cards, or other sorts of loans. Similar to that, there is a chance that a consumer won't pay their invoices if a company provides credit to them. Credit risk also refers to the possibility that a bond issuer won't pay up when required or that an insurance provider won't be able to cover a claim. The borrower's general capacity to repay a loan in accordance with its original terms is used to determine credit risks.

However, since it's impossible to predict who exactly may pay late, correctly evaluating and managing credit risk can decrease the impact of a loss. It helps in predicting and/ or measuring the risk factor of any transaction.

CHAPTER 1

INTRODUCTION

1.1 Loan Risk

Loan risk refers to the possibility of a repayment on a debt due to the incomplete payments. The lender is at risk, and this typically includes things like lost principal and interest, disruption of cash flows, and higher collection expenses. Loans can be obtained for a number of purposes, including bank mortgages (or home loans), financing for a car purchase, credit card purchases, installment sales, retail loans, and more. Finances and credit loans run the danger of defaulting or falling behind.

Financial firms that lend money to people and businesses must perform credit risk analysis, including financial risk analysis, credit loan default risk analysis, and retail loan default analysis. Credit providers typically compile a lot of data on borrowers in order to identify the risk factors associated with credit customers. It is possible to assess or determine the risk levels associated with credits, finances, and loans, or default risk levels, using statistical predictive analytic approaches.

Being able to effectively manage the credit risk management process is crucial for banks' success as it is one of their fundamental business activities. Risks are usually related with banking activities and accepting risks is a regular practice in the banking industry, therefore banks are continually faced with them. By having formal credit risk management practices in place, banks must constantly balance their risks to ensure their sound financial standing. Implementing appropriate credit risk management also helps banks improve their earnings, customer management procedures, and raise accountability within the firm.

1.2 Prediction Models

Predictive modeling is a mathematical procedure that analyses patterns in a given collection of input data to forecast future events or outcomes. Although predictive modeling suggests a focus on projecting the future, it may also predict outcomes. If new data indicates changes in what is happening now, the influence on the expected future outcome must also be reassessed. Forecasting business outcomes through predictive analytics saves time, effort, and money.

Predictive analytics is another term for predictive modeling. In general, predictive modeling is chosen in academic contexts, whereas "predictive analytics" is favored in commercial uses of predictive modeling. The successful application of predictive analytics is strongly dependent on having unrestricted access to large amounts of accurate, clean, and relevant data. While predictive models using decision trees and k-means clustering can be quite complex, the most complex aspect is always the neural network; that is, the model by which computers are trained to anticipate outcomes. A neural network is used in machine learning to uncover correlations in extremely large data sets and to "learn" and recognise patterns within the data.The following are the top five predictive analytics models:

- Classification model is the most basic model and categorizes data for simple and quick query response. This approach groups data by common qualities. It works by grouping items or people with similar features or habits and planning solutions on a bigger scale for each group
- Clustering model groups data by common qualities. It works by grouping items or people with similar features or habits and planning solutions on a bigger scale for each group
- Forecast model is a popular model that works with everything that has a numerical value and is built on learning from historical data.
- Outliers model analyses anomalous or outlying data points.
- Time series model uses time to analyze a series of data items.

Machine learning or deep learning are used in predictive algorithms. Both are artificial intelligence subsets (AI) as shown in Figure 1.1. Structured data, such as spreadsheets or machine data, is used in machine learning (ML). Deep learning (DL) is concerned with unstructured data such as video, audio, text, social media posts, and images , everything with which humans communicate that is not a number or metric read.

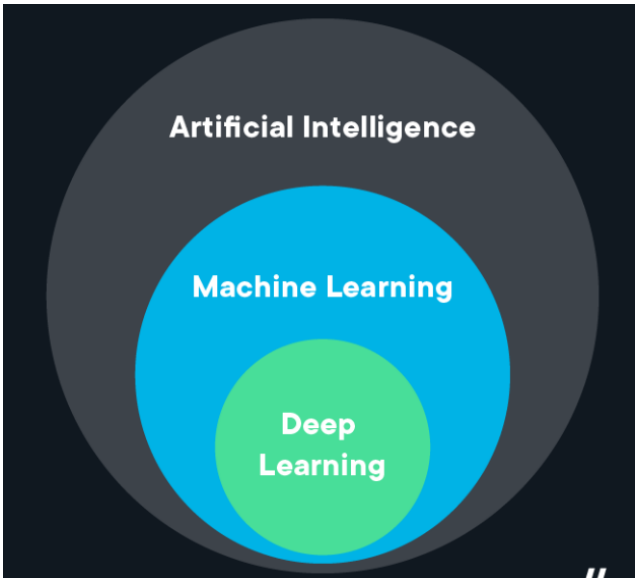

Figure 1.1 AI vs ML vs DL

The following are some of the more common prediction algorithms:

Random Forest is derived from a combination of decision trees, none of which are connected, and can categorize large volumes of data using both classification and regression.

Generalized Linear Model (GLM) for Two Values narrows the list of variables in order to obtain the "best match." It may discover tipping points and adjust data capture and other effects, such as categorical predictors, to decide the "best fit" conclusion, overcoming shortcomings in other models, such as conventional linear regression.

Gradient Boosted Model, like Random Forest, uses numerous coupled decision trees, but the trees are related. It constructs one tree at a time, allowing the subsequent tree to remedy errors in the preceding tree. It's frequently used in rankings, such as on search engine results pages.

K-Means, a popular and fast method that groups data points based on similarities, is frequently used for the clustering model. It can swiftly generate customized retail offers for individuals within a large group, such as a million or more clients who share a preference for lined red wool jackets.

Prophet: This technique is used for capacity planning in time-series or prediction models, such as inventory needs, sales quotas, and resource allocations. It is extremely adaptable and can easily incorporate heuristics and a wide range of helpful assumptions.

LVQ(Learning Vector Quantization): A special case of an artificial neural network based on prototype supervised learning classification algorithm and trained its network through a competitive learning algorithm.It is a precursor to self-organizing maps (SOM) and related to neural gas, and to the k-nearest neighbor algorithm (k-NN)

1.3 Problem Statement

Lenders consider the five Cs when assessing credit risk on a consumer loan: credit history, repayment capacity, capital, loan terms, and collateral. Some businesses have created departments that are completely responsible for assessing the credit risks of their current and future clients.

Information on the borrower's previous credit provided by other banks, previous home credit loans held by the applicant, previous home credit point of sale and cash loans held by the applicant, previous credit cards held by the applicant with home credit, as well as transactional history are samples of data that can be utilized to anticipate loan risk.

The project's purpose is to decide whether or not to offer the loan to the customer based on the following data items: loan amount, number of years, interest rate, installment, years of employment, annual income, and some other basic information. The dataset will be utilized to train the algorithm, and the outcome will be whether or not to offer a loan.

1.4 System Configuration

The following are the hardware and the software specifications used in this system.

- Hardware Configuration
    - Processor : Minimum Pentium Processor
    - RAM : 8 GB (Recommended)
    - Hard Disk Capacity : 1 GB
- Software Configuration
    - Programming Language: Python
    - IDE : Visual Studio Code | Sublime text
    - Operating System : Windows 10 or later

CHAPTER 2

DATA COLLECTION AND PREPROCESSING

Deep Learning models are data-hungry and require a lot of data to create the best model or a system with high fidelity. The quality of data is as important as its quantity. Preprocessing is necessary before using data. The process of transforming raw data into clean data is known as data preprocessing. Before executing the algorithm, the dataset is preprocessed to look for missing values, noisy data, and other inconsistencies. The format of the data must be suitable for machine learning models.

2.1 Dataset

Datasets are groups of data. In the case of tabular data, a data set relates to one or more database tables, where each row refers to a specific record in the corresponding data set and each column to a specific variable. A dataset is referred to as multivariate if it comprises two or more different data kinds. To put it another way, the multivariate dataset is made up of distinct measurements that were obtained as functions of three or more variables.

The dataset is collected from a popular P2P lending platform Lending Club. Lending Club is a financial services company headquartered in San Francisco, California. The dataset is obtained from the website Kaggle. The dataset consists of 151 columns and 2260701 records. The sample of the dataset is shown in figure 2.1

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	id	member_i	loan_amnt	funded_an	funded_an	term	int_rate	installmen	grade	sub_grade	emp_title	emp_lengt	home_ow	annual_inc	verificatio	issue_d	loan_statu	pymnt_pla	url	desc	purpose	title	zip_code	ad
2	68407277		3600	3600	3600	36 month	13.99	123.03	C	C4	leadman	10+ years	MORTGAG	55000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	debt_cons	Debt cons	190xx	PA	
3	68355089		24700	24700	24700	36 month	11.99	820.28	C	C1	Engineer	10+ years	MORTGAG	65000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	small_busi	Business	577xx	SD	
4	68341763		20000	20000	20000	60 month	10.78	432.66	B	B4	truck drive	10+ years	MORTGAG	63000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	home_improvement		605xx	IL	
5	66310712		35000	35000	35000	60 month	14.85	829.9	C	C5	Informatic	10+ years	MORTGAG	110000	Source Ver	Dec-15	Current	n	https://lendingclub.cc	debt_cons	Debt cons	076xx	NJ	
6	68476807		10400	10400	10400	60 month	22.45	289.91	F	F1	Contract S	3 years	MORTGAG	104433	Source Ver	Dec-15	Fully Paid	n	https://lendingclub.cc	major_pur	Major purc	174xx	PA	
7	68426831		11950	11950	11950	36 month	13.44	405.18	C	C3	Veterinary	4 years	RENT	34000	Source Ver	Dec-15	Fully Paid	n	https://lendingclub.cc	debt_cons	Debt cons	300xx	GA	
8	68476668		20000	20000	20000	36 month	9.17	637.58	B	B2	Vice Presic	10+ years	MORTGAG	180000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	debt_cons	Debt cons	550xx	MI	
9	67275481		20000	20000	20000	36 month	8.49	631.26	B	B1	road drive	10+ years	MORTGAG	85000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	major_pur	Major purc	293xx	SC	
10	68466926		10000	10000	10000	36 month	6.49	306.45	A	A2	SERVICE M	6 years	RENT	85000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	credit_car	Credit card	160xx	PA	
11	68616873		8000	8000	8000	36 month	11.48	263.74	B	B5	Vendor lia	10+ years	MORTGAG	42000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	credit_car	Credit card	029xx	RI	
12	68356421		22400	22400	22400	60 month	12.88	508.3	C	C2	Executive	6 years	MORTGAG	95000	Not Verifie	Dec-15	Current	n	https://lendingclub.cc	debt_cons	Debt cons	290xx	SC	
13	68426545		16000	16000	16000	60 month	12.88	363.07	C	C2	Senior Stru	1 year	MORTGAG	70000	Not Verifie	Dec-15	Current	n	https://lendingclub.cc	debt_cons	Debt cons	786xx	TX	
14	68338832		1400	1400	1400	36 month	12.88	47.1	C	C2	Logistics M	3 years	MORTGAG	64000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	other		275xx	NC	
15	66624733		18000	18000	18000	60 month	19.48	471.7	E	E2	Software I	7 years	RENT	150000	Not Verifie	Dec-15	Charged O	n	https://lendingclub.cc	debt_cons	Debt cons	916xx	CA	
16	68466961		28000	28000	28000	36 month	6.49	858.05	A	A2	Senior Mai	10+ years	MORTGAG	92000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	debt_cons	Debt cons	275xx	NC	
17	68354783		9600	9600	9600	36 month	7.49	298.58	A	A4	tech	8 years	MORTGAG	60000	Not Verifie	Dec-15	Fully Paid	n	https://lendingclub.cc	credit_car	Credit card	299xx	SC	

Figure 2.1 Sample Dataset

The Lending Club dataset is a multivariate dataset. The details of the dataset can be displayed using the method info(), it is shown in the figure 2.2

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2260701 entries, 0 to 2260700
Columns: 151 entries, id to settlement_term
dtypes: float64(113), object(38)
memory usage: 2.5+ GB
None
```

Figure 2.2 Dataset Information

2.2 Filling Null Values

Data cleaning is the process of identifying and correcting inaccurate/incorrect data in a dataset. Dealing with the dataset's missing values is one of the steps that are required. Using 'df.isnull().sum()', where df contains the dataset as a dataframe, we can see the null values in the dataset. Figure 2.3 displays the sum of the null values in the columns for the first 2000 rows in the dataset.

```
id                          0
member_id                2000
loan_amnt                   0
funded_amnt                 0
funded_amnt_inv             0
                          ...
settlement_status        1959
settlement_date          1959
settlement_amount        1959
settlement_percentage    1959
settlement_term          1959
Length: 151, dtype: int64
```

Figure 2.3 Sum of null values

The heatmap method in the Seaborn library can be used to display the dataset's null values. A heatmap is a type of graph where values are represented by colors and are used to show data. A package called Seaborn uses Matplotlib as its foundation to plot graphs. Figure 2.4 uses a heatmap to display the null values in the dataset without any preprocessing for the first 2000 records in the dataset. Regions with more null values are represented by lighter colors, whereas regions without null values are shown by darker colors.

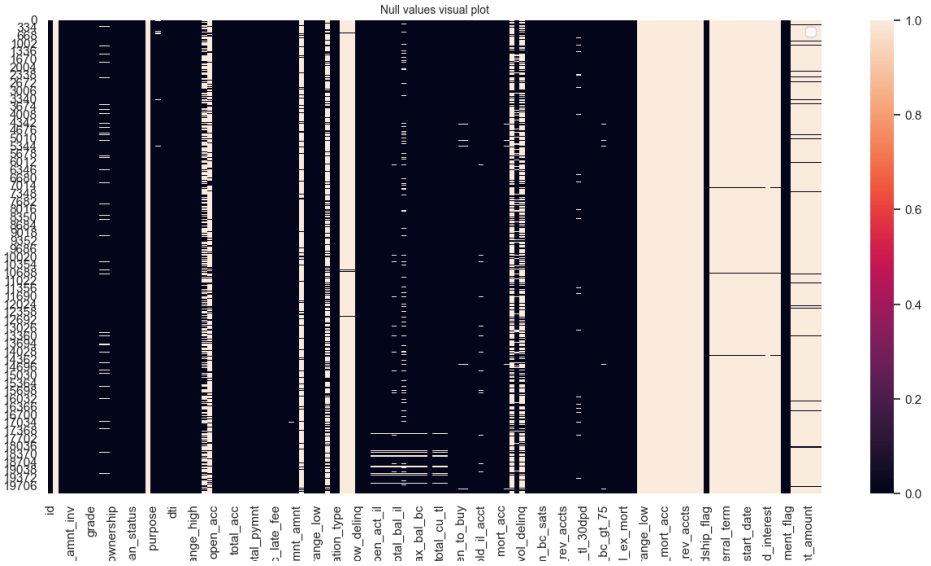

Figure 2.4 Heatmap before filling the null values

The null values in the dataset can be handled in a variety of ways, including dropping the rows and columns with null values, filling the missing value with imputation (using the mean, median, maximum, or minimum for numerical data, and a mode with the maximum occurrence, minimum occurrence, a new type for categorical data), imputation with a new column, and filling with regression.

The columns containing null values more than 10 percent of the total values are dropped. Filling more than 10 percent of the null values can affect the accuracy of the model. The rows with the null values are also dropped. The figure 2.5 shows the code snippet for dropping the rows and columns.

```
input_missing=pd.DataFrame(
    {'column':input_file.columns,
    'percent_missing':input_file.isnull().sum()*100/len(input_file)
})
input_file.drop(input_missing[input_missing['percent_missing']>10].index,axis=1,inplace=True)
input_file = input_file.dropna(axis = 0).reset_index(drop = True)
```

Figure 2.5 Dropping rows and columns

46 columns are dropped from the dataset because they have null values more than 10 percent. The details of the dataset after dropping the null values for are shown in the figure 2.6

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1654 entries, 0 to 1653
Columns: 105 entries, id to debt_settlement_flag
dtypes: float64(81), int64(1), object(23)
memory usage: 1.3+ MB
None
```

Figure 2.6 Dataset after dropping null values

2.3 Categorical to numerical

All input and output variables for machine learning models must be numeric. The levels of a categorical variable are high. The model's performance is reduced as a result. When working with categorical data, encoding is a necessary pre-processing step for machine learning algorithms. Ordinal encoding, one hot encoding, and dummy variable encoding are the various types of encoding.

Each distinct category value in ordinal encoding is given an integer value. Reversing it is simple. Integer values beginning with zero are frequently utilized. For each distinct integer value in the variable, a new binary variable is added and the integer-encoded variable is discarded in one hot encoding. In contrast to one hot encoding, which adds N columns for N unique values, dummy variable encoding adds N-1 columns for N unique values.

We use ordinal encoding in the verification status, home ownership, employment length, and loan status fields of the lending club dataset. There are three distinct values for the verification status and house ownership fields. The loan status field contains 10 unique values, and the employment length field has 11 distinct values. The loan status is encoded into three values: 1, 0, and -1, with records having a value of -1 being deleted. Due to the lack of information regarding these customers' status, these data are not necessary for estimating the loan risk. Figure 2.7 illustrates the ordinal encoding for the field loan status and the dataset's restriction to entries with values of 0 and 1.

```
replace_status = {'Fully Paid':1,
                  'Current': 1,
                  'Charged Off': 0,
                  'Does not meet the credit policy. Status:Charged Off':0,
                  'Does not meet the credit policy. Status:Charged Off':0,
                  'Does not meet the credit policy. Status:Fully Paid':1,
                  'Late (31-120 days)':-1,
                  'Late (16-30 days)':-1,
                  'In Grace Period':-1,
                  'Default':0
                 }
input_file['loan_status'] = input_file['loan_status'].replace(replace_status)
input_file = input_file[ (input_file['loan_status']== 1) | (input_file['loan_status']== 0)]
```

Figure 2.7 Ordinal Encoding of the loan status field

Figure 2.8 shows the ordinal encoding for the fields verification status, home ownership, and employment length.

```
input_file['emp_length_numeric']=input_file['emp_length'].map({
    '10+ years':1,'< 1 year':0.05,
    '2 years':0.2,'3 years':0.3,
    '1 year':0.1,'5 years':0.5,
    '6 years':0.6,'4 years':0.4,
    '8 years':0.8,'9 years':0.9,'7 years':0.7})
input_file['verification_status_numeric']=input_file['verification_status'].map({
    'Not Verified':0,
    'Source Verified':1,
    'Verified':0.5})
input_file['home_ownership']=input_file['home_ownership'].map({
    'OWN':1,
    'RENT':0,
    'MORTGAGE':0.5})
```

Figure 2.8 Ordinal Encoding of home ownership, employment length, verification status

The subgrade field represents the grade the bank assigned to the customer based on account maintenance. The lending club website is used to collect the risk percentage for each subgrade field, which is then used to transform the categorical values to numerical values. There are 5 different levels (1,2,3,4,5) in each of the 7 grades (A,B,C,D,E,F,G). As a result, the subgrade field has 35 different values. Figure 2.9 illustrates the code snippet for converting the subgrade's categorical value to a numerical value.

```
risk_rate_values={'A':[3.41,3.97,4.51,5.14,5.76],
                  'B':[8.28,8.97,9.66,10.35,11.03],
                  'C':[12.25,13.19,14.07,14.9,15.69],
                  'D':[17.57,19.5,22,24.6,25.94],
                  'E':[23.85,23.87,23.9,23.92,23.95],
                  'F':[24.3,24.64,25.12,25.6,25.7],
                  'G':[25.74,25.79,25.84,25.89,25.94]}

ir_row = input_file['sub_grade'].apply(lambda x : x[0:1])
ir_col = input_file['sub_grade'].apply(lambda x : x[1:2])
ir_col = [eval(k) for k in ir_col]
temp = []
for i in range(len(input_file)):
        temp.insert(i,risk_rate_values[ir_row[i]][ir_col[i]-1])
input_file = input_file.assign(risk_rate=temp)
```

Figure 2.9 Converting the subgrade field

The example of data in the subgrade field in the categorical data before processing and the numerical data after processing is shown in the figure 2.10

1	C1	1	12.25
2	C5	2	15.69
3	F1	3	24.30
4	C3	4	14.07
	..		...
16877	C5	16877	15.69
16878	C1	16878	12.25
16879	C4	16879	14.90
16880	B4	16880	10.35
16881	B4	16881	10.35

Figure 2.10 Subgrade field in categorical and numerical form

Data in the field term is transformed from categorical to numerical. The term is expressed in months which is converted to years . First, the field is cleared of the prefix months. The field is then transformed into an integer field. Years are afterwards transformed from months. The code snippet for this process is shown in figure 2.11

```
input_file['term_years'] = input_file['term'].apply(lambda x : x[0:3])
input_file['term_years'] = input_file.term_years.astype('Int64')
input_file['term_years'] = input_file['term_years']/12
input_file['term_years'] = input_file.term_years.astype('Int64')
```

Figure 2.11 Conversion of the field term

A small example of the categorical data in months before processing and numerical data in years after processing is shown in the figure 2.12

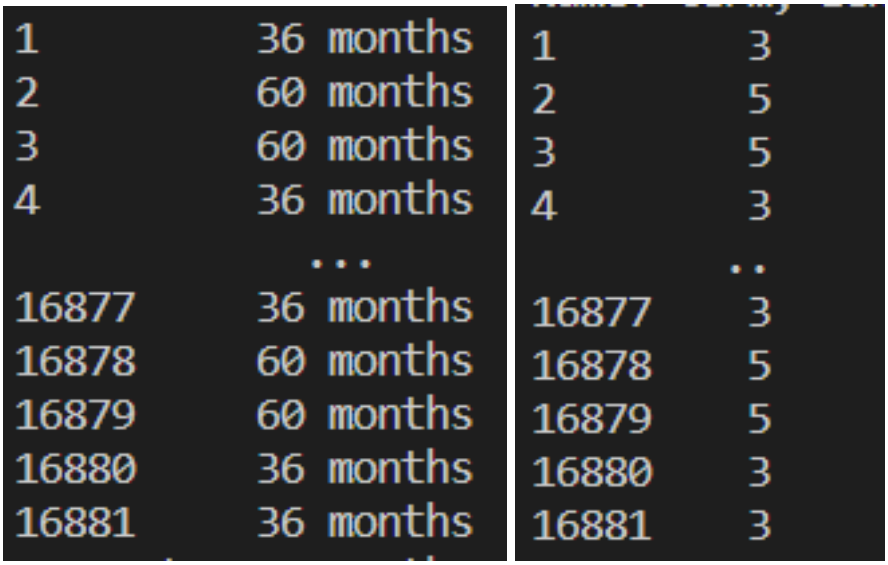

Figure 2.12 Term field in categorical and numerical form

2.4 Feature Selection

When developing a machine learning model in practice, it is usually never the case that all of the variables in the dataset are usable for building a model. Repetitive inputs decrease a classifier's capacity to generalize and may also lower the classifier's overall accuracy. When creating a predictive model, the process of feature selection involves reducing the number of

input variables. In some circumstances, reducing the number of input variables might enhance the efficiency of the model and also minimize the computing cost of modeling. Finding the optimal set of features for building practical models of the phenomena being studied is the aim of feature selection in machine learning.

After removing the columns having null values more than 10%, we are left with 105 features in the dataset. 21 of these columns are chosen to train the models. The dataset is trained using only the fields that are necessary. Fields like address, zip code, member id, etc. do not aid in determining a person's credit risk. The features which are selected to train the model is shown in figure 2.13

```
columns=['loan_amnt','term_years','int_rate','installment','risk_rate','emp_length_numeric',
         'annual_inc','dti','delinq_2yrs','fico_range_low','inq_last_6mths','open_acc','pub_rec',
         'revol_bal','revol_util','total_acc','tot_cur_bal','home_ownership','verification_status_numeric',
         'total_pymnt','loan_status']
input_file=input_file[columns]
```

Figure 2.13 Feature Selection

The description of the features selected for training the model in the figure 2.13 is explained in the table 2.1

Table 2.1 Description of the features

FEATURE	DESCRIPTION
loan_amnt	The listed amount of the loan applied by the borrower
term_years	The number of payments of the loan in years
int_rate	Interest rate of the loan
installment	The monthly payment owed by the borrower
risk_rate	Lending Club assigned risk rate
emp_length_numeric	Employment length of the borrower
annual_inc	Annual income of the borrower during registration

dti	A ratio calculated using the borrower's total monthly debt payments on the debt obligations, excluding mortgage and the requested loan, divided by the borrower's self - reported annual income
deling_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
fico_range_low	The lower boundary range the borrower's FICO at loan origination belongs to.
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
open_acc	Number of open credit line in the borrowers credit file
pub_rec	Number of derogatory public records
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit
total_acc	The total number of credit lines currently in the borrowers credit file
tot_cur_bal	Total current balance of all accounts
home_ownership	The home ownership status provided by the borrower during the registration or obtained from the credit report
verification_status_numeric	Indicates if the co-borrowers' joint income was verified, not verified, or if the income source was verified
total_pymnt	Payments received to date for total amount funded
loan_status	Current status of the loan

2.5 Normalization

The fundamental purpose of data normalization is to reduce or eliminate duplicate data. Data duplication is a serious problem. This is due to the fact that maintaining identical data in several locations while storing it in a dataset is becoming increasingly difficult.

It is simpler to apply algorithms to a set of normalized data. Algorithms applied to

normalized data produce more precise and efficient outputs. On these data, more specialized data analysis techniques can be used. The purpose of normalization is to convert features so that they are all on the same scale.

2.5.1 Techniques in normalization

The four common techniques used in normalization are scaling to a range, feature clipping, log scaling and z score. The technique used for normalization differs based on the data in the input feature. Same technique of normalization cannot be applied for all data.

Scaling is the method of changing floating-point feature values from their original range into a predetermined range, typically between 0 and 1. When the upper and lower bounds are known and the data is evenly distributed across a range, it is a wise decision. The formula for scaling is given in figure 2.14

$$x' = (x - x_{min})/(x_{max} - x_{min})$$

Figure 2.14 Formula for scaling

For data values with extreme outliers, feature clipping is used, which sets all feature values above (or below) a predetermined value. Either before or after the normalizing procedure, feature clipping may be used. The formula for feature clipping is

Set min/max values to avoid outliers

When the majority of your values have few points and a small number of them have many, log scaling can be effective. This data distribution is known as the power law distribution. In order to enhance the performance of a linear model, log scaling modifies the distribution. To compress a wide range of data into a narrow range, log scaling computes the log of given values. The formula for log scaling is given below

x' = log(x)

The number of standard deviations from the mean is represented by a scaling variant called a z-score. The feature distributions are ensured to have mean = 0 and standard = 1 . When there are a few outliers, it is helpful but not as extreme as feature clipping. The formula for z-score of a point x is given in the figure 2.15.

$$x' = (x - \mu)/\sigma$$

Figure 2.15 Formula for z-score

2.5.2 Normalization in the dataset

Scaling is the normalizing method applied to the dataset. The values in the dataset have been scaled to lie between 0 and 1. The features' maximum and minimum values are chosen first. The values are then modified utilizing the scaling formula shown in figure 2.14.

The scaling approach is used to normalize the values in the 16 input features. As a result, the algorithm performs better when it is implemented. Figure 2.16 displays the code snippet for normalizing the chosen input feature.

```
col=['loan_amnt','int_rate','installment','risk_rate','annual_inc','fico_range_low',
     'dti','open_acc','pub_rec','revol_bal','total_acc','tot_cur_bal','total_pymnt',
     'max_bal_bc','total_rec_int','avg_cur_bal']
for i in col:
        temp=input_file[i]
        max=temp.max()
        min=temp.min()
        input_file[i] = (input_file[i]-min)/(max-min)
```

Figure 2.16 Normalizing the data

The algorithm accepts either an integer or float data type as input. Since predictive modeling requires numerical data, all the dataset's chosen attributes are converted into integer and float values. Figure 2.17 displays the selected input features and their details after data processing. There are 2 int data type features and 19 float data type features in the dataset. After removing the null value rows from the 10,000 records, all 21 columns include just non-null values and have 8426 records. The number of records removed from the dataset varies depending on the number of records taken as input.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8426 entries, 0 to 8490
Data columns (total 21 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   loan_amnt                     8426 non-null   float64
 1   term_years                    8426 non-null   Int64
 2   int_rate                      8426 non-null   float64
 3   installment                   8426 non-null   float64
 4   risk_rate                     8426 non-null   float64
 5   emp_length_numeric            8426 non-null   float64
 6   annual_inc                    8426 non-null   float64
 7   dti                           8426 non-null   float64
 8   delinq_2yrs                   8426 non-null   float64
 9   fico_range_low                8426 non-null   float64
 10  inq_last_6mths                8426 non-null   float64
 11  open_acc                      8426 non-null   float64
 12  pub_rec                       8426 non-null   float64
 13  revol_bal                     8426 non-null   float64
 14  revol_util                    8426 non-null   float64
 15  total_acc                     8426 non-null   float64
 16  tot_cur_bal                   8426 non-null   float64
 17  home_ownership                8426 non-null   float64
 18  verification_status_numeric   8426 non-null   float64
 19  total_pymnt                   8426 non-null   float64
 20  loan_status                   8426 non-null   int64
dtypes: Int64(1), float64(19), int64(1)
memory usage: 1.4 MB
None
```

Figure 2.17 Details of the input features after data preprocessing

CHAPTER 3

LVQ AS A PREDICTION MODEL

3.1 Introduction to LVQ

Learning Vector Quantization (LVQ) is a type of artificial neural network inspired by biological neural system models It is built on a prototype supervised learning classification algorithm , its network is trained using a competitive learning method similar to Self-Organizing Map (SOM) and supervised counterpart of vector quantization systems. It supports both binary (two-class) and multi-class classification problems. Multiclass or multinomial classification is the problem of classifying instances into one of three or more classes. The input layer and the output layer make up the two layers of LVQ as shown in Figure 3.1.

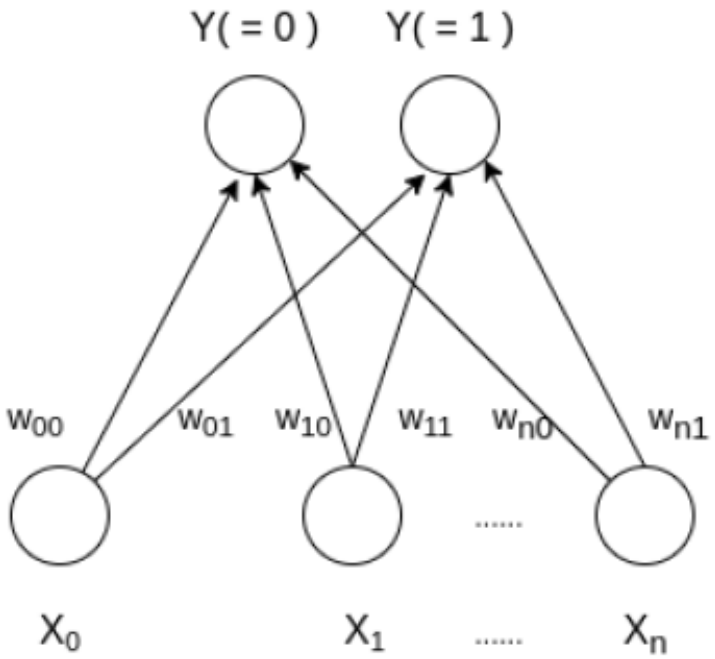

Figure 3.1 Layers of LVQ

A prototype is typically used to assess a new design in order to improve precision. Prototyping provides requirements for an actual, working system rather than to a theoretical one. The task of classification is determining which of a set of categories an observation belongs to.

3.1.1 Supervised learning

Supervised learning (SL) is a machine learning paradigm for problems with labeled examples, which means that each data point has characteristics and an associated label. Based on example input-output pairings, supervised learning algorithms develop a function that maps feature vectors (inputs) to labels (outputs). It derives a function from labeled training data, which consists of a set of training samples. The training data provided to the machines acts as a supervisor, teaching the machines to accurately forecast the outcome. It employs the same principle that a student would learn under the supervision of a teacher.

Supervised learning is the process of supplying correct input and output data to a machine learning model. A supervised learning algorithm's goal is to find a mapping function that maps the input variable (x) to the output variable (y). The algorithm generalizes from the training data to unseen scenarios. This statistical quality of an algorithm is quantified by the generalization error, which is a measure of how well an algorithm predicts outcome values for previously unseen data. Figure 3.2 shows the classification of supervised learning.

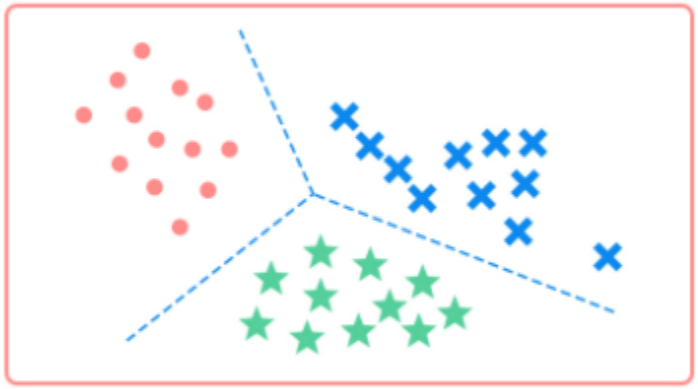

Figure 3.2 Classification in Supervised Learning

3.1.2 Vector Quantization

Vector quantization (VQ) is a classical quantization technique from signal processing that allows the modeling of probability density functions by the distribution of prototype vectors. Originally, it was utilized for data compression. It divides a big set of points (vectors) into groups

with roughly the same number of points nearest to them. As in k-means and other clustering algorithms, each group is represented by its centroid point. Vector quantization's density matching property is particularly useful for determining the density of huge and high-dimensional data. Because data points are represented by the index of their nearest centroid, often occurring data have low error and rare data have high error.

Vector quantization is based on the competitive learning paradigm, so it is closely related to the self-organizing map model and to sparse coding models used in deep learning algorithms such as autoencoder. Competitive learning in artificial neural networks is a type of unsupervised learning in which nodes fight for the right to respond to a portion of the input data. Competitive learning is a type of Hebbian learning, operated by improving the specialization of each node in the network. It works effectively for locating clusters in data.

The Hebbian learning rule is one of the neural network's first and simplest learning rules used to classify patterns. It is a one layer neural network, which means it has only one input and one output layer. The input layer can have a large number of units, say n. There is only one unit in the output layer. For each training sample, the Hebbian rule updates the weights between neurons in the neural network.

3.2 Technical Aspects of LVQ

An input data of size (m, n) where m is the number of training examples and n is the number of features in each example and a label vector of size (m, 1). First, it initializes the weights of size (n, c) from the first c number of training samples with different labels and should be discarded from all training samples. Here, c is the number of classes. Then iterate over the remaining input data, for each training example, it updates the winning vector (weight vector with the shortest distance using Euclidean distance formula from the training example). The weight updation rule is given by:

if correctly_classified:

$$w_{ij}(new) = w_{ij}(old) + alpha(t) * (x_i^k - w_{ij}(old))$$

else:

$$w_{ij}(new) = w_{ij}(old) - alpha(t) * (x_i^k - w_{ij}(old))$$

where alpha is a learning rate at time t, j denotes the winning vector, i denotes the ith feature of training example and k denotes the kth training example from the input data. After training the

LVQ network, trained weights are used for classifying new examples. A new example is labeled with the class of the winning vector.

General steps involved are :

- Weight initialization
- For 1 to N number of epochs
- Select a training example
- Compute the winning vector
- Update the winning vector
- Repeat steps 3, 4, 5 for all training examples.
- Classify test sample

3.2.1 LVQ Model Representation and Prediction

The representation for LVQ is a collection of codebook vectors. A codebook vector is a list of numbers that have the same input and output attributes as your training data. For example, if your problem is a binary classification with classes 0 and 1, and the inputs width, length height, then a codebook vector would be composed of all four attributes: width, length, height and class. The model representation is a fixed pool of codebook vectors, learned from the training data. They look like training instances, but the values of each attribute have been adapted based on the learning procedure. In the language of neural networks, each codebook vector may be called a neuron, each attribute on a codebook vector is called a weight and the collection of codebook vectors is called a network.

Predictions are made using the LVQ codebook vectors in the same way as K-Nearest Neighbors.Predictions are made for a new instance (x) by searching through all codebook vectors for the K most similar instances and summarizing the output variable for those K instances. For classification this is the mode (or most common) class value. Typically predictions are made with K=1, and the codebook vector that matches is called the Best Matching Unit (BMU). To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) for each attribute j.

$$\text{EuclideanDistance}(x_i, x_j) = \text{sqrt}(\sum (x_i - x_j)^2)$$

3.2.2 Learning an LVQ Model From Data

The LVQ algorithm learns the codebook vectors from the training data. The number of codebook vectors to use has to be chosen such as 20 or 40. The best number of codebook vectors to use can be found by testing different configurations on a training dataset. The learning algorithm starts with a pool of random codebook vectors. These could be randomly selected instances from the training data, or randomly generated vectors with the same scale as the training data. Codebook vectors have the same number of input attributes as the training data. They also have an output class variable. The instances in the training dataset are processed one at a time. For a given training instance, the most similar codebook vector is selected from the pool. If the codebook vector has the same output as the training instance, the codebook vector is moved closer to the training instance. If it does not match, it is moved further away. The amount that the vector is moved is controlled by an algorithm parameter called the learning_rate. If the input variable (x) of a codebook vector is moved closer to the training input value (t) by the amount in the learning_rate if the classes match as follows

$$x = x + learning\_rate * (t - x)$$

else the opposite case of moving the input variables of a codebook variable away from a training instance is calculated as

$$x = x - learning\_rate * (t - x)$$

This would be repeated for each input variable. Because one codebook vector is selected for modification for each training instance the algorithm is referred to as a winner-take-all algorithm or a type of competitive learning. This process is repeated for each instance in the training dataset. One iteration of the training dataset is called an epoch. The process is completed for a number of epochs chosen (max_epoch) such as 200, initial learning rate (such as alpha=0.3). The learning rate is decreased with the epoch, starting at the large value specified at epoch 1 which makes the most change to the codebook vectors, and finishing with a small value near zero on the last epoch, making very minor changes to the codebook vectors. The learning rate for each epoch is calculated as

$$learning\_rate = alpha * (1 - (epoch/max\_epoch))$$

where learning_rate is the learning rate for the current epoch (0 to max_epoch-1), alpha is the learning rate specified by the algorithm at the start of the training run, and max_epoch is the total number of epochs to run. The algorithm is also specified at the start of the run. The intuition for

the learning process is that the pool of codebook vectors is a compression of the training dataset to the points that best characterize the separation of the classes

3.3 LVQ in Loan Risk

LVQ is one of the best classification-based prediction algorithms. The Lending Club dataset used for loan risk prediction is preprocessed so that it contains only numerical data. The output for this prediction model is class. Therefore, the output can be easily classified using the classification algorithm. The scalability and complexity of the algorithm also play an important role. LVQ is scalable and not too complex.

Many different difficulties can be solved with the use of classification. It enables one to make more intelligent decisions. The dataset consists of densely interconnected features. Hence, LVQ is used to develop the loan risk prediction model. The detailed design and implementation of the algorithm are explained in the following chapters.

CHAPTER 4

LVQ FOR LOAN RISK PREDICTION

The Learning Vector Quantization (LVQ) algorithm is a sort of Artificial Neural Network (ANN) that is similar to k-Nearest Neighbors. Finding the best match from a library of patterns is how predictions are made. The distinction is that instead of employing training patterns directly, the library of patterns is learned from training data. The design, implementation, tuning of parameters for accuracy and the output visualization of the loan risk prediction using the LVQ algorithm is discussed in this chapter.

4.1 Design and Implementation

In LVQ, we use a pattern library known as the codebook vectors. A codebook is the name given to each pattern in the library of patterns. The initialization of the codebook vectors uses values chosen at random from the training dataset. Then, using a learning algorithm, they are modified over a number of epochs to best summarize the training data. The code snippet for choosing the codebooks at random is shown in figure 4.1.

randrange method in the random library is used to select the random records. It returns randomly selected records from the specified range. Here, the randrange method runs for the number of codebooks given by the user.

The learning algorithm shows one training record at a time, locates the best matching unit among the codebook vectors, and moves it in either direction depending on whether they belong to the same class or a different class from the training record. Once ready, the codebook vectors are utilized using the k-Nearest Neighbors algorithm, where k=1, to create predictions.

```
def random_codebook(train):
    n_records = len(train)
    n_features = len(train[0])
    codebook = [train[randrange(n_records)][i] for i in range(n_features)]
    return codebook
```

Figure 4.1 Selecting the codebook vector

4.1.1 Euclidean Distance

The distance between two rows in a dataset must first be calculated. Rows of data are mostly made up of numbers, and drawing a straight line is an easy way to calculate the distance between two rows or vectors of numbers. This works well in 2D and 3D and scales well to higher dimensions. Using the Euclidean distance measure, we can compute the straight line distance between two vectors. It is calculated as the square root of the sum of the two vectors' squared differences.

The smaller the value of the Euclidean distance, the more similar the two records will be. A value of 0 indicates that no difference exists between two records. Figure 4.2 shows a code snippet for the Python function euclidean distance(). The function assumes that the last column in each row is an output value that is ignored when calculating the distance.

```
def euclidean_distance(row1, row2):
    distance = decimal.Decimal(0)
    for i in range(len(row1)-1):
        distance += (decimal.Decimal(row1[i]) - decimal.Decimal(row2[i]))**decimal.Decimal(2)
    return sqrt(distance)
```

Figure 4.2 Function to calculate euclidean distance

The Decimal method in the decimal library is used to calculate the Euclidean distance. It is used to perform some decimal floating point tasks. This module provides accurate- rounded floating point arithmetic

4.1.2 Best Matching Unit

The codebook vector that is most similar to a new piece of data is known as the Best Matching Unit, or BMU. To find the BMU for a new piece of data within a dataset, compute the distance between each codebook and the new data. It is accomplished using the euclidean_distance method. After the distances have been calculated, sort all of the codebooks by their nearness to the new data. The first or most similar codebook vector is then returned. It is accomplished by storing the distance between each record in the dataset as a tuple, sorting the list of tuples by distance (in descending order), and then retrieving the BMU.

Figure 4.3 shows the code snippet of the implementation of the get_best_matching_unit method. The euclidean_distance() function developed in the previous step is used to calculate the distance between each codebook and the new test_row

```
def get_best_matching_unit(codebooks, test_row):
    distances = list()
    for codebook in codebooks:
        dist = euclidean_distance(codebook, test_row)
        distances.append((codebook, dist))
    distances.sort(key=lambda tup: tup[1])
    return distances[0][0]
```

Figure 4.3 To find the best matching unit

A custom key is used to sort the list of codebook and distance tuples, ensuring that the second item in the tuple (tup[1]) is used in the sorting operation. The 1-nearest neighbor algorithm has been used. That is, for each new pattern, predict the most similar codebook vector in the set and return the class value associated with it.

4.1.3 Training codebook vectors

For each training pattern, the best matching unit is found and only this best matching unit is updated. The error is calculated as the difference between the training pattern and the BMU. The class values (which are assumed to be the last in the list) are compared. If they match, the error is added to the BMU to bring it closer to the training pattern; if they don't, the error is subtracted to push it further away.

A learning rate determines how much the BMU is adjusted. This is a weighted average of the changes made to all BMUs. A learning rate of 0.3, for example, indicates that BMUs are only moved by 30% of the error or difference between training patterns and BMUs.

The steps that follow are repeated iteratively.

- Epochs: At the highest level, the process is repeated for a set number of epochs or training data exposures.
- Training Dataset: Each training pattern is used one at a time within an epoch to update the set of codebook vectors.

- Pattern Features: Each feature of a best matching codebook vector is updated for a given training pattern to move it closer or further away.

Figure 4.4 shows the code snippet for the above explained process.The function takes three additional arguments in addition to the training dataset: the number of codebook vectors to create and train, the initial learning rate, and the number of epochs to train the codebook vectors over.

```
def train_codebooks(train, n_codebooks, lrate, epochs):
    codebooks = [random_codebook(train) for i in range(n_codebooks)]
    for epoch in range(epochs):
        rate = lrate * (1.0-(epoch/float(epochs)))
        sum_error = decimal.Decimal(0)
        for row in train:
            bmu = get_best_matching_unit(codebooks, row)
            for i in range(len(row)-1):
                error = row[i] - bmu[i]
                sum_error += decimal.Decimal(error)**decimal.Decimal(2)
                if bmu[-1] == row[-1]:
                    bmu[i] += rate * error
                else:
                    bmu[i] -= rate * error
        print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, rate, error))
    return codebooks
```

Figure 4.4 Training the codebook

4.1.4 Train-test split

When machine learning algorithms are used to make predictions on data that was not used to train the model, the train-test split procedure is used to estimate their performance. It is a quick and simple procedure with results that make it possible to compare the performance of machine learning algorithms for the predictive modeling problem. When we have a large dataset, a costly model to train, or we need a good estimate of model performance quickly, the train-test split procedure is appropriate.

The procedure involves dividing a dataset into two subsets. The first subset, known as the training dataset, is used to fit the model. The second subset is not used to train the model; instead, the model is fed the dataset's input element, and predictions are made and compared to expected values. The second dataset is known as the test dataset.

The train and test data are split using the train_test_split() method in the sklearn library. There are three common ways of splitting the input dataset. They are represented in the ratios of 80:20, 67:33 and 50:50. For the loan risk prediction model, the training data is 67 percent, and the test data is 33 percent.

4.1.5 Accuracy

The most well-known machine learning model validation method used in classification problems is accuracy. One reason for its popularity is its ease of use. It is simple to comprehend and implement. For simple cases, accuracy is a good metric to use to evaluate model performance.The number of correct predictions divided by the total number of predictions yields the accuracy. The formula for calculating accuracy is given figure 4.5

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

Figure 4.5 Formula for accuracy

The accuracy of binary classification is calculated by using the True Positive, False Positive, True Negative, and False Negative values. These values are described in table 4.1. Figure 4.6 shows the formula for calculating the accuracy for binary classification.

Table 4.1 Values in accuracy formula

VALUE	DESCRIPTION
True Positive	The model correctly predicts the positive class
True Negative	The model correctly predicts the negative class
False Positive	The model incorrectly predicts the positive class
False Negative	The model incorrectly predicts the negative class

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

Figure 4.6 Formula for accuracy (Binary Classification)

Since the loan risk prediction model is a binary classification model, we are finding the true positive, true negative, false positive, and false negative values and calculating the accuracy using the formula in figure 4.6. The code snippet of the implementation of the accuray_metric function is shown in figure 4.7.

```
def accuracy_metric(actual, predicted):
    correct=0
    tp=0
    tn=0
    fp=0
    fn=0
    for i in range(len(actual)):
        if actual[i]==1 and predicted[i]==1:
            tp+=1
        if actual[i]==0 and predicted[i]==0:
            tn+=1
        if actual[i]==1 and predicted[i]==0:
            fn+=1
        if actual[i]==0 and predicted[i]==1:
            fp+=1
        if actual[i] == predicted[i]:
            correct += 1
    acc=(tn+tp)/(tn+tp+fn+fp)
    return acc*100.0
```

Figure 4.7 Calculating the accuracy

4.2 Tuning Parameters

The challenge of selecting a set of ideal hyperparameters for a learning algorithm is known as hyperparameter optimization, or tuning in machine learning. A parameter whose value is used to influence the learning process is known as a hyperparameter. To generalize various data patterns, the same machine learning model may need different constraints, weights, or learning rates. Hyperparameters are those variables that need to be adjusted in order for the model to work best when solving a machine learning challenge. An ideal model is produced

through hyperparameter optimization, which identifies a tuple of hyperparameters that minimizes a predetermined loss function on given independent data.

In LVQ, we can change the values of the number of epochs, number of codebooks, and learning rate. The codebook vector contains records based on the number of codebooks given as input. This step is for the initial weight initialization. Epochs are similar to iterations. The learning rate represents the difference between the training patterns and the best matching unit. Table 4.2 shows the values of the above parameters and their corresponding accuracy.

Table 4.2 Parameters and their corresponding accuracy

S.No.	Number of records	Learning rate	Number of epochs	Number of codebooks	Accuracy
1	2000	0.3	20	20	79.97
2	2000	0.3	10	10	79.97
3	5000	0.3	20	20	84.25
4	10000	0.3	10	40	84.57
5	10000	0.3	10	50	84.57
6	10000	0.3	20	40	84.60
7	10000	0.3	10	60	84.53
8	10000	0.3	10	70	84.60
9	10000	0.3	50	70	84.50
10	10000	0.3	30	80	84.64
11	10000	0.3	40	80	84.53
12	10000	0.3	20	100	84.64
13	10000	0.3	100	100	84.57
14	10000	0.3	20	200	84.25
15	15000	0.2	10	10	83.10
16	15000	0.2	10	30	83.10

17	15000	0.3	10	10	83.10
18	15000	0.3	10	100	83.03
19	15000	0.3	40	100	83.00
20	15000	0.3	25	100	83.10

4.3 Output Visualization

We must understand how the model makes decisions. In the case of neural networks, the scope of the problem becomes especially obvious. Because they perform several non-linear transformations during training, real-world neural network models are extremely complex internally. Building trust in models that can be essential in preparing for relief or security activities would be made easier with the aid of visualization of such complex models.

Figure 4.8 shows the change in the learning rate and the error rate for every epoch. Here, the number of epochs is 10, the number of codebooks is 10, the number of records taken as input is 10000, and the initial learning rate is 0.3. Finally, the accuracy is calculated for the test data set of 33 percent.

```
>epoch=0, lrate=0.300, error=-0.158
>epoch=1, lrate=0.270, error=-0.167
>epoch=2, lrate=0.240, error=-0.164
>epoch=3, lrate=0.210, error=-0.173
>epoch=4, lrate=0.180, error=-0.177
>epoch=5, lrate=0.150, error=-0.212
>epoch=6, lrate=0.120, error=-0.228
>epoch=7, lrate=0.090, error=-0.243
>epoch=8, lrate=0.060, error=-0.258
>epoch=9, lrate=0.030, error=-0.273
Accuracy : 84.6458108594031
```

Figure 4.8 Command line output

The actual and predicted values for the inputs are 10000 (number of records), 0.3 (learning rate), 10 (number of epochs), and 10 (number of codebooks) are plotted using the matplotlib library in Python. Figure 4.9 shows the variation in the actual and predicted output values.

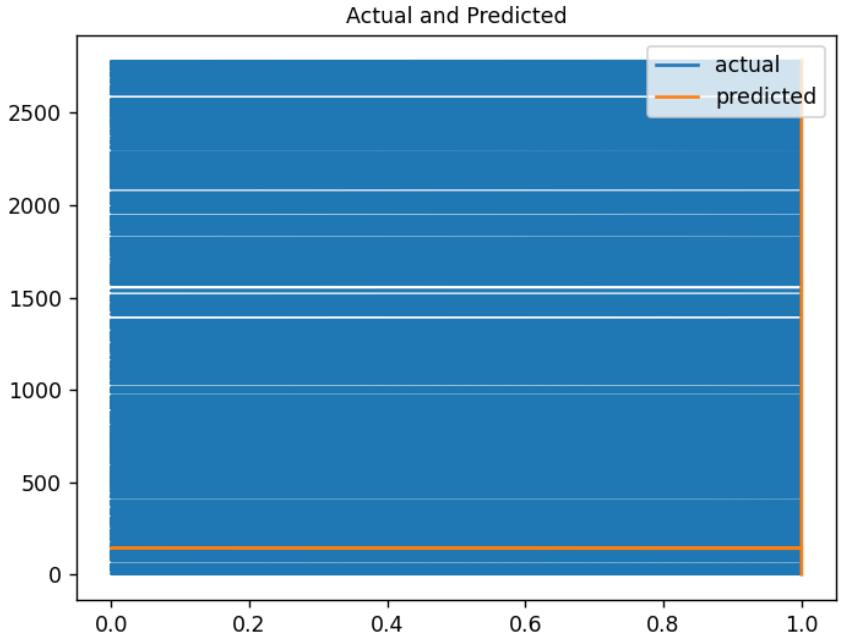

Figure 4.9 Actual and predicted values

The decrease in the values of the learning rate from 0.3 to 0.10 and the error rate from -0.16 to -0.30 is shown in figures 4.10 and 4.11, respectively.

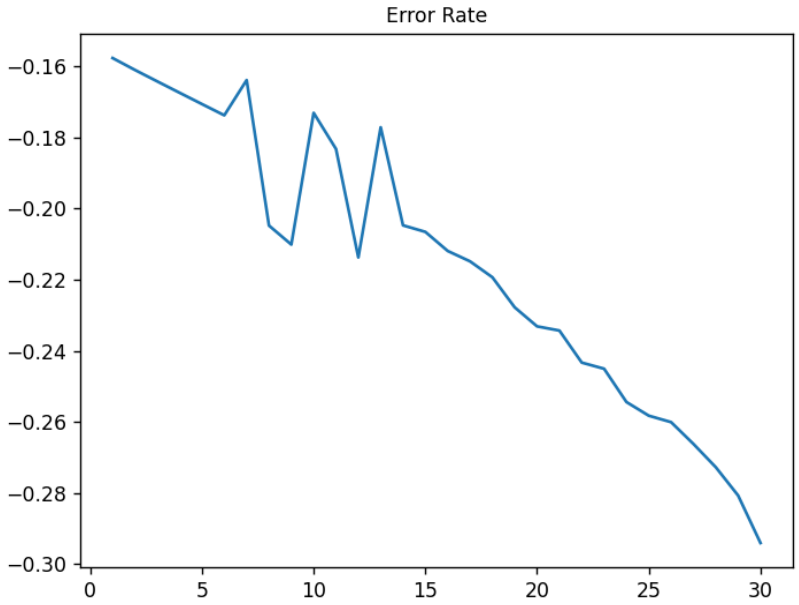

Figure 4.10 Decrease in error rate

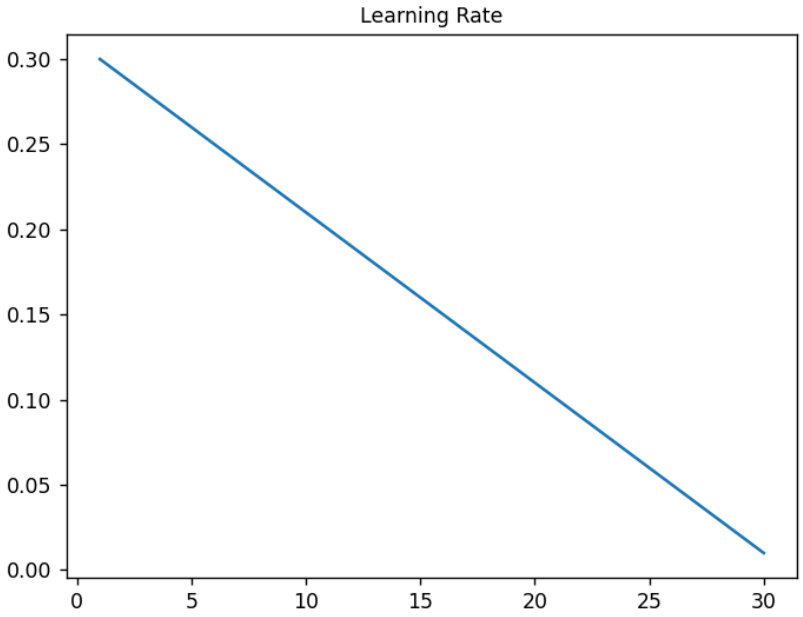

Figure 4.11 Decrease in learning rate

CONCLUSION

A supervised learning classification algorithm, LVQ, is used for this prediction. This helps the banks to prevent loss due to the loans. If properly predicted, loans can bring a huge profit to the banks. Loan risk prediction model is calculated using the selected features from the users credit history. The densely interconnected features are used for the prediction. Thus a neural network model is more useful.

Loan risk cannot be predicted for the users if they do not provide inputs for the features used for predicting the model since the prediction is based on these features. The risk cannot be calculated accurately for the customers who have recently opened their account in the bank. The bank does not have enough data for these customers. Most of the data provided during the creation of the account is not for the prediction.

In future, the accuracy of the model can be improved by learning the extensive versions of the LVQ or by using the algorithms to increase the performance of the model. Higher the accuracy, the model is more reliable.

BIBLIOGRAPHY

[1] https://www.techtarget.com/searchenterpriseai/definition/predictive-modeling

[2] https://machinelearningmastery.com/learning-vector-quantization-for-machine-learning/

[3] https://www.geeksforgeeks.org/learning-vector-quantization/

[4] https://en.wikipedia.org/wiki/Supervised_learning

[5] https://en.wikipedia.org/wiki/Competitive_learning

[6] https://www.irjet.net/archives/V8/i3/IRJET-V8I3446.pdf

[7] https://developers.google.com/machine-learning/data-prep/transform/normalization