ABW508/ABW508D ANALYTICS LAB

Supervised By

Assoc. Prof. Dr Teh Sin Yin

Personal Loans Acceptance Predictive Modelling Using Machine Learning Techniques

By

Kareem Mustafa Muhamed Eldemerdash

Master of Business Analytics

P-EM0112/22

Year of Submission

2023

ACKNOWLEDGEMENT

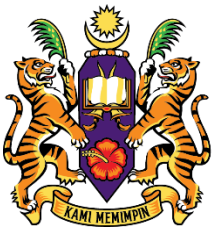
I would like to express my heartfelt gratitude to my supervisor, Assoc. Prof. Dr Teh Sin Yin, for her invaluable guidance and support throughout my research project. Her expertise and constant encouragement helped me to develop my skills and stay focused on my work.

I am deeply grateful to my family and friends for their love, support, and understanding throughout my studies. Without their constant encouragement, I could not have completed this work.

Finally, I would like to acknowledge the research participants who generously gave their time and contributed to this study. Their willingness to share their experiences and insights is appreciated.

Thank you all for your contributions to my research journey.

Table of Contents



Abstract

In this paper, the author proposes a machine learning approach to predict the acceptance of personal loans by clients of a bank. The motivation for this study is the challenge faced by banks in identifying the best potential customers for personal loans, and in determining the characteristics that make a customer more likely to accept a loan. The author obtained data on past personal loan applications and use machine learning algorithms to analyze the data. The goal is to develop a decision support system that can accurately classify clients into two groups: those who are likely to accept a personal loan, and those who are not.

The author also aims to identify the optimal combination of characteristics that a potential customer should possess to be classified as likely to accept a personal loan. To achieve this, they employ a feature selection method to determine the most key features in the dataset. The results of the study show that the machine learning model developed in this work can accurately predict whether a client is likely to accept a personal loan with a high degree of accuracy.

The author discusses the implications of these findings for the banking industry and suggest areas for future research. They conclude that the use of machine learning techniques can significantly improve the accuracy of personal loan acceptance prediction and can potentially lead to more efficient loan approval processes for banks.

List of Tables

List of Figures

1. Introduction

1.0. Introduction

Most industries especially the medical, industrial, and financial ones have been impacted by machine learning. In addition to handling big data that human saves a significant amount of time and money.

This chapter aims to explain the study's background, Personal loans' importance and vital economic roles, Problem Statement, Research Objectives, Research Questions, Significance of the Study, and Organization of the Remaining Chapters.

1.1. Study Background of Machine Learning in Banking.

A bank is a type of financial entity that lends money, its two main activities are taking deposits from the public and offering loans. The bank can use capital markets to conduct lending activities directly or indirectly. (Marois, 2022)

Because most transactions are now conducted online, this affected the banking industry significantly. The bank is a governmental entity subject to governmental regulations. The Bank offers six diverse kinds of bank loans: Loans for education, PPF loans, business loans, personal loans, gold loans, and loans for vehicles. (Duggirala, 2020)

There are over nine hundred public banks worldwide, and they are a significant force. Their assets values are close to $49 trillion, amid the fintech revolution banks were one of the most targeted segments to apply fintech technology to enhance their operations. Digital banking is considered one of the most important fintech products easing the customer's lives and lowering the bank's costs (Marois, 2022)

Machine learning became a crucial tool in risk scoring and clustering clients, also it helps in marketing where it helps in choosing the efficient client to target. on assumptions about the data, particularly assumptions about the distribution because they are data-driven and computationally based. Although they are seen to be more dependable and effective at handling complicated non-linear interactions, they are also thought to be challenging to comprehend. (Galindo & Tamayo, 2000)

1.2. Personal Loans

Personal loans make it possible to purchase products and services. A loan allows us to purchase something on credit. (Leo et al., 2019)

A personal loan can be defined as money that a person borrows from a bank or other financial organization for personal use. Personal loans are a type of consumer financing provided by lenders such as banks and financial institutions to assist consumers in facing a short-term shortage in personal finances. It is for purchasing big items, for family use, and/or other household big expenses. Such loans are either secured by assets such as collateral or by a related individual who acts as the guarantor, and loans are unsecured. The emergence of mass markets for consumer goods and the attainment of lofty standards of living by Western consumers are often attributed to the widespread availability of personal loans. Much is known about the marketing of personal loan services in Western countries. Yet, undoubtedly owing to its introduction into the retail-banking scene, not too many studies are available on this topic in the context of developing countries. (Ismail et al., 2013)

1.3. Problem Statement

The banking sector primarily focuses on providing loans to customers which gains more money for the bank. The identification of a potential customer who should accept taking a loan from the bank is a critical issue because of the risk of Non-payment back. Thus, the credit team decided to choose people who have deposits in the bank because they have no risk of paying back. Sales teams face these issues:

Failure in choosing the best customer who will accept the loan and Failure in identifying the characteristics of the ideal potential customer.

1.4. Research Objectives

1) To develop a decision support system using a machine learning model that classifies the clients of the bank into two groups who are more likely to take loans or not.

2) To determine the optimal combination of characteristics that are likely to accept personal loans.

1.5. Research Questions

RQ 1: what is the optimal machine learning algorithm that can classify personal loan clients?

RQ 2: What are the highest features that indicate the clients may accept the loans?

1.6. Significance of the Study

The study's practical importance is applying the findings to all banking institutions that are interested in personal loans which will save more time and money and increase bank revenues. and lead to the development of the credit strategy of the bank

Enhancing machine learning in the personal loan process makes it more objective in a data-driven organization and eliminates human error.

1.7. Organization of the Remaining Chapters:

Chapter 1: This chapter offers an introduction to the topic in this study as well as the study's purpose, objectives, research questions, and scope. Chapter 2: This chapter expands on the literature reviews undertaken for this investigation. Chapter 3: This chapter describes the study's methodological structure, from data retrieval to model evaluation. Chapter 4: The outcomes of the experiment on the selected algorithms are reported in this chapter. Chapter 5: This chapter examines the study's empirical findings, ramifications, and limitations and concludes the study by making recommendations for further work.

2. Literature Review

2.0. Introduction

This chapter is divided into two parts the first one is the literature review conducted for this study. Machine learning algorithms that used in similar or nearest cases for classification purposes using supervised machine learning. The second part is a review of different algorithms that were performed.

2.1. Personal Loans Prediction

The banking system contains two sides to the accounting entry creditors who lend bank money to invest or give loans then banks pay interest as a rent of money and the other side of the entry debtors may be companies or persons who need money for several purposes investing or consumption then pays the money and higher interest to the bank and the difference between two interests is the profit of the bank.

Personal loans that the loans for consumption purposes, not investment, and this study tries to identify the potential clients for personal loans from our creditors two increase profits and Promote interest between the bank and its creditors.

This paper (Duggirala, 2020) discussed the same issue and defined this problem as a binary classification model and implemented and compared the accuracy in this research between seven models: Linear Classifiers: Logistic Regression, K-Nearest Neighbor, Support Vector Machine (SVM), Decision Trees, Random Forest, Naive Bayes, Gradient Boosting and used confusion matrix to evaluate the model effectiveness.

They gained different results because of different algorithms that were applied to the dataset. The logistic Regression test accuracy score is 94%. However, the false negative is fifty-eight. KNN test accuracy rating is 92%. However, there are seventy-seven false negatives in the confusion matrix. The support Vector Machine test accuracy Algorithm is 94% and false negative 143. The decision Tree Algorithm test accuracy is 98.2% and 16 are false negatives.

The Accuracy of Random Forest is 98.8% with a false Negative of fifteen. The Accuracy of Naive Bayes' is 88% and 56 false negatives. Gradient Boosting has a 98.6% accuracy rate and the fewest false negatives twelve, compared to all other models.

Random forest gained the highest accuracy among 98.8% of all models and Gradient Boosting had the least false negative twelve.

In this article, (Agarwal et al., 2022) the same topic was examined. A binary classification model was used to identify the issue, and accuracy was tested using five different models: Logistic Regression, K-Nearest Neighbor, Support Vector Machine (SVM), Decision Trees, and Random Forest.

They scored different accuracies because of different preprocessing Logistic Regression scored Test data accuracy 0.9533 and Training data accuracy 0.9566. K-Nearest Neighbor Test data accuracy 0.9080 and Training data accuracy 1.0000. Support Vector Machine (SVM) Test data accuracy 0.9740 and Training data accuracy 0.9786. Decision tree Test data accuracy is 0.9833 and Training data accuracy is 0.9897. Random forest test data accuracy is 0.9880 and training data accuracy is 1.0000.

Logistic Regression scored 0.8558 in precision and 0.6181 in the recall. K-Nearest Neighbor scored 0.5385 in precision and 0.2917 in recall. The decision tree scored 0.9161 in precision and 0.9097 in recall. Random forest scored 0.9772 in precision and 0.8958 in the recall. Support Vector Machine (SVM) scored 0.9487 in precision and 0.7708 in recall.

Random forest is the best algorithm according to this article it scored test data accuracy 0.9880, training data accuracy 1.0000, 0.9772 in precision, and 0.8958 in the recall.

In this article (Furkan Akça & Sevli, 2022) they collected some previous papers that discussed the same problem and found that Logistic Regression accuracies are from77% to 81.1%, SVM accuracies are from 77% to 85%, Random Forest about 80%, Decision Tree is 95%, Then used four types of SVM Linear SVC, Poly SVC, Sigmoid SVC, Rbf SVC.

In terms of accuracy, SVM is one of the top statistical learning or machine learning algorithms. A support vector machine technique with four different kernel types was applied in this investigation. The analysis findings indicate that a polynomial kernel (97%) that cost a lot of time produced the greatest outcomes, while a sigmoid kernel (83%) produced the lowest results.

So, we have four models we can use in this study Gradient Boosting, Random Forest, SVM, and decision tree.

2.2. Supervised Learning for Binary Classifications

Personal loan prediction can be regarded as a binary classification problem because the aim is to classify data points into one of two classes: either Yes or No to predict if the client will take a personal loan or not.

According to the previous papers, we can focus on just four of the supervised learning algorithms: support vector machine, Decision tree, random forest, and gradient boosting. They were chosen due to their simplicity and popularity in binary classification. Also, in this paper, a deep learning model will be conducted using a Neutral Network algorithm in addition to algorithms. These algorithms will be discussed in the following sections.

2.2.1. Support Vector Machine

The Support Vector Machine oversees determining the decision boundary to divide several classes and increase the margin which is the separations (perpendicular to the line) between the nearest dots and the line. Finding a hyperplane that splits a dataset into two groups most effectively is the foundation of SVMs. Data points that fall (exactly) on the margins' edges are known as support vectors. To calculate the margin, just these points are required.(Ke et al., 2023)

There are two types of data separable and non-separable data. linear separable data, SVM tries to find the Hyperplane. Unfortunately, in real datasets, most of the data are non-linear separable so it is hard to find the hyperplane that's why SVM creates new two concepts: Soft Margin and Kernel Tricks.(Esteki & Naghsh-Nilchi, 2022)

The soft margin is a method used to balance the tradeoff between having a model that is more accurate on the training data and one that generalizes better to unseen data. This is done by allowing some misclassification of the training data, rather than trying to fit the model perfectly to the training data. The soft margin is often used in support vector machines (SVMs), which are a type of model used for classification tasks.

The kernel trick is a mathematical technique used to map data from a low-dimensional space into a higher-dimensional space, to perform a computation more efficiently or to improve the performance of a machine-learning algorithm. It is often used in conjunction with SVMs. In an SVM, the kernel trick is used to transform the data into a higher-dimensional space, where it is

easier to find a hyperplane that can separate the data points into different classes. The kernel function used to perform the transformation can be chosen based on the characteristics of the data.

Sklearn.svm.SVC(), includes linear, poly, rbf, sigmoid, precomputed, or a callable as our kernel/transformation.

SVM has advantages like in the higher dimension, it is quite effective. Also, useful when there are more features than training samples. When classes can be separated, the best algorithm. Outliers have less effect. However, all these advantages it has disadvantages, especially when working for a large amount of time requires time and does not work effectively with overlapping classes.(Gupta et al., 2019)

SVM can be applied in several fields Document classification or document categorization, computer vision, and handwriting recognition.

2.2.2. Decision Tree

Decision Trees are a supervised machine learning in which the training data is continually segmented based on a certain parameter, with you describing the input and the associated output. Decision nodes and leaves are the two components that are used to explain the tree. The choices or results are represented by the leaves. The data is divided at the decision nodes. (Fan et al., 2022)

There are two main types of Decision Trees: one is used for classification tasks with categorical classes, and the other is used for regression tasks with continuous data. Classification trees are also known as "Yes/No" types, while regression trees are used to predict continuous values. One of the greatest algorithms for learning based on different learning techniques is decision trees. They improve predictive models' accuracy, usability, and stability. Since the tools can handle difficult data-fitting problems like classification and regression, they are also useful for fitting non-linear relationships. It has statistical algorithms for building decision trees are available, including CART (Classification and Regression Trees), C4.5, CHAID (Chi-Squared Automatic Interaction Detection), and QUEST (Quick, Unbiased, Efficient, Statistical Tree.(Ismaeil et al., 2022)

The strength points of decision trees are that it is easy to use and understand, can handle categorical and numerical data, resistant to outliers, so data preprocessing effort is low, new features can be easily added and can be used to create larger classifiers using ensemble methods.

And weak points of decision trees are that it is easy to overfit, Structures such as decision trees are unstable if a slight change in the data occurs, Decision trees are sometimes more complex than other algorithms., and biased learning trees can be created when certain classes dominate.(Rathore et al., 2023)

The decision tree can be applied in Assessing prospective growth opportunities, using demographic data to find prospective clients, and Serving as a support tool in several fields.

2.2.3. Random Forest

Random forest is also known as decision forests. It is an ensemble learning method (bagging). The main issue with decision trees is that they are greedy algorithms. They choose the best variable to be divided by that greedy algorithm which reduces errors. As a result, the decision trees might still have structural similarities and, consequently, have a high correlation in their predictions, even when using bagging.(Venkateswarlu & Anmala, 2023)

Random Forest is used in banking to detect customers who are more likely to repay their debt on time. it is also used to predict who will use a bank's services more frequently. They even use it to detect fraud. Also, Stock traders use Random Forest to predict a stock's future behavior. It is used by retail companies to recommend products and predict customer satisfaction as well. Random Forest is sometimes used in computational biology and genetics research. (Al-Manaseer et al., 2023; Venkateswarlu & Anmala, 2023)

It has the same hyperparameters as a decision tree or a bagging classifier, but it searches for the best feature among a random subset of features. form most current machine learning systems.

Random Forest is popular, and for good reasons. It offers a variety of advantages, from accuracy and efficiency to relative ease of use. For data scientists using Python, scikit-learn provides a convenient library for implementing Random Forests. Using the bagging method and random feature selection when executing this algorithm completely resolves the problem of overfitting. Shortly, Random Forest is easy to use, accurate, efficient, and recommended for beginners more than neural nets. (Zeng et al., 2022)

Although there are not drawbacks to Random Forest, every tool has some limitations. Because random forest employs decision trees, they may consume a significant amount of memory for larger projects. This issue can occasionally affect the entire forest because this method is based on decision trees, which frequently experience overfitting Because Random Forest uses random feature subsets and creates smaller trees using those subsets, this issue is typically avoided by default.

2.2.4. Gradient Boosting

A machine learning method called gradient boosting is utilized, among other things, in regression and classification problems. It provides a prediction model in the form of an ensemble of weak prediction models, most often decision trees. it is one of the most powerful techniques for building predictive models. The concept of boosting originated from the dilemma of whether a poor learner could be improved. When a hypothesis or learner performs at least slightly better than chance, it is said to be weak. The aim behind hypothesis boosting was to filter observations, keeping only those that the weak learner could manage, and concentrating on developing new weak learners to handle the remaining challenging observations.(Abbasniya et al., 2022)

Adaptive Boosting, or AdaBoost for short, was the first boosting realization that experienced significant application success. Decision trees with a single split, sometimes known as decision stumps due to their briefness, are the weak learners in AdaBoost. AdaBoost works by weighing the observations, giving harder-to-classify examples more weight and easier-to-classify instances less. Weak learners are gradually introduced, and they focus their learning on the more challenging patterns. The prediction of the weaker learners is voted on by a majority, with each prediction's correctness being given more weight. AdaBoost was the name of the AdaBoost algorithm, which was the most effective version for binary classification issues.(Vadivukkarasi & Santhi, 2021)

Gradient Boosting is known as gradient tree boosting. The statistical framework framed boosting as a numerical optimization problem where the goal is to add weak learners using a gradient descent-like method to minimize the loss of the model. The stage-wise additive model was used to characterize this class of methods. This is because the model only adds one new weak

learner at a time, freezing and maintaining the weak learners that already exist.(Shahsavar et al., 2022; Uttam & Mangal, 2020)

Gradient boosting involves three elements. First, Loss Function which used according to the type of the problem. It should be differentiable, but there are standard loss functions you can use or define on your own. Second, Weak Learner the decision trees specifically regression trees. Third, Additive Model The model's existing trees are not changed, and new trees are added one at a time. The loss when adding trees is reduced using a gradient descent procedure.

Gradient boosting has advantages in accuracy, and it is fast in the train, especially on large datasets, handling categorical values, and some of them handle missing values but it may be expensive and take a lot of time, especially on CPUs, and hard to interpret the champion model. The field of learning to rank can benefit from gradient boosting. In their machine-learned ranking engines, commercial web search engines employ different iterations of gradient boosting. In High Energy Physics, gradient boosting is also used for data analysis.(Jai Vignesh et al., 2023)

2.2.5. Neural Network

Neural Networks named artificial neural networks (ANNs) are a part of machine learning and the backbone of deep learning algorithms. Their name and design were derived from the human brain and how it works and sends signals to one another. A node layer in an ANN consists of an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, is interconnected and has a threshold and weight that go with it. A node is activated and begins sending data to the network's next layer if its output is greater than the specified threshold value for that node. No data is otherwise transmitted to the network's next layer.(Muksin et al., 2023)

It can be confusing because the terms "deep learning" and "neural networks" are frequently used interchangeably in speech. It is important to remember that the "deep" in deep learning simply denotes the number of layers in a neural network. A neural network with more than three layers, including the inputs and outputs, is referred to as a "deep learning algorithm." Simply put, a basic neural network is one that only has two or three layers. The connected layers of neural networks, which serve as a model of the human brain, have a wide range of uses, including the ability to predict the weather and identify faces.(Xing et al., 2023)

In applications that give users an automated robotic experience, neural networks serve as the foundation. In understanding the working conditions and generating desired outputs, the current systems need to be modified in many ways. In order to study the conditions under which human testing is restricted, more sophisticated mechanisms are required for a variety of applications and problems, including space exploration to provide workable results that can aid in the advancement of research in these scenarios, it must evolve as a substitute.(Lee et al., 2022; Walker et al., 2023)

3. Methodology

3.0 Introduction

This chapter discusses all steps to conduct the study. The steps were performed using Jupyter Notebook (6.5.4) on a PC with AMD Ryzen 5 4500U with Radeon Graphics 2.38 GHz processor and 8 GB RAM Using Windows 11 operating system. The project was conducted using Python programming. The end-to-end flow consists of six main steps performed to determine the champion model to predict who will accept personal loans. The full implementation of the steps discussed can be found in Appendix A. This chapter discusses step by step the entire process from data retrieval, data pre-processing, exploratory data analysis (EDA), and machine learning algorithms.

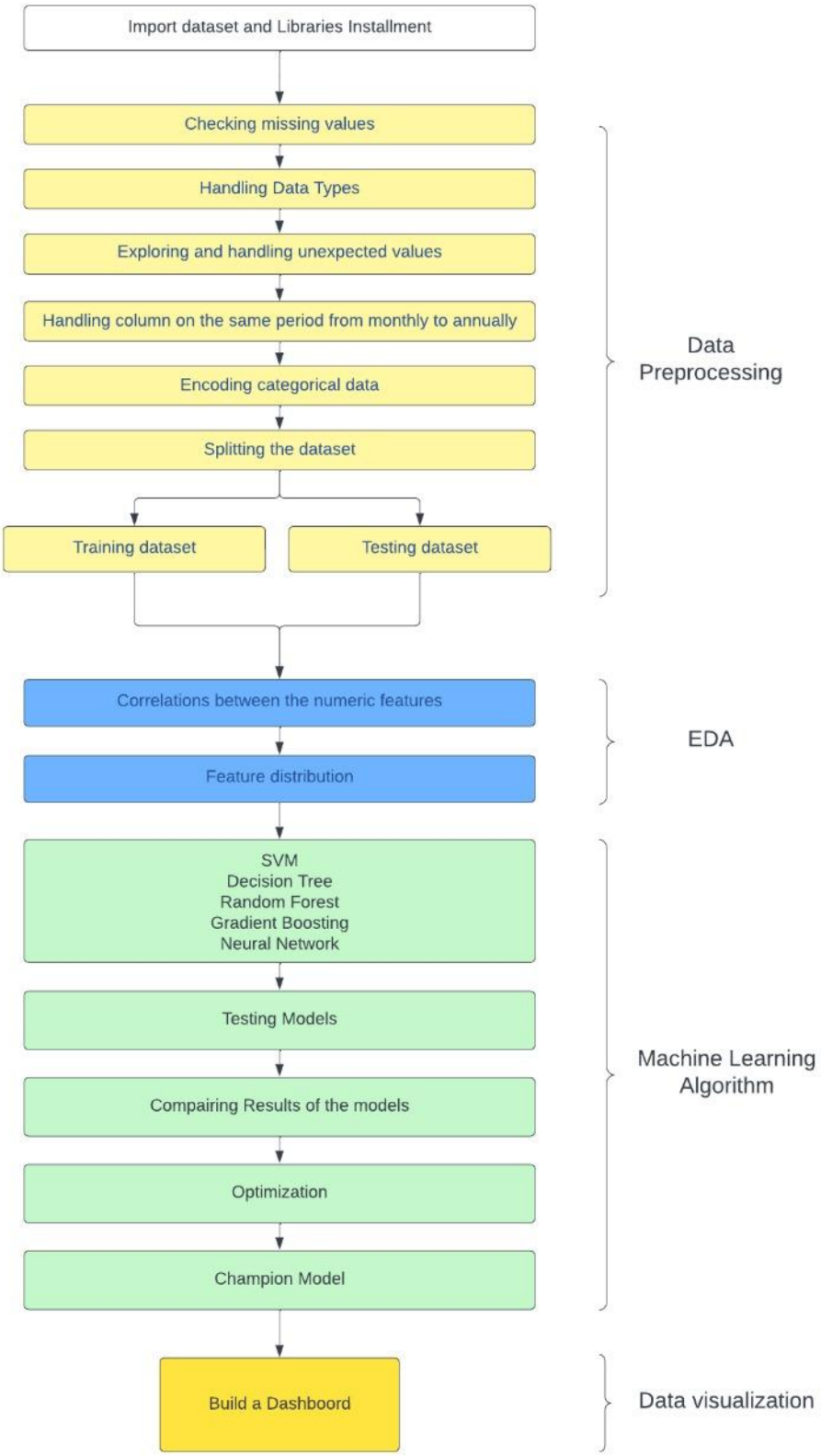


Figure 3-1: Flowchart of prediction of personal loans acceptance

3.1 Data Retrieval

Personal loan data used in this study is available on the Kaggle website. Thera Bank is a US bank that has a growing customer base. The bulk of these clients are liability clients (depositors), and their deposits range in amount. Asset clients make up a very tiny portion of the bank's customer base, and the bank is interested in swiftly growing this group to attract more loan business and, as a result, increase its interest-earning potential. The management is especially interested in finding ways to turn its liability consumers into personal loan customers (while retaining them as depositors).

The data available is in a comma-separated values (CSV) file format with labeled axes. Double backward slashes were used while specifying the file path. The dataset consists of fourteen features and five thousand observations. The features explanation is shown in table 3-1:

Table 3-1 Features dictionary and description

Feature Name	Feature type	Description
ID	independent	unique identifier.
Personal Loan	Dependent (Target)	Customers accept the personal load offered (1=Yes, 0=No).
Age	independent	The customer's age.
Experience	independent	The number of years of professional experience.
Income	independent	The annual income of the customer.
Zip code	independent	Home address zip code.
Family	independent	Family size of the customer.
CC Avg	independent	Average spending on credit cards per month.
Education	independent	Education level (1) undergraduate, (2) graduate, (3) advanced/professional.
Mortgage	independent	Value of house mortgage.
Securities	independent	Customers have a securities account with the bank (1=Yes, 0=No).
CD Account	independent	Customers have a certificate of deposit with the bank (1=Yes, 0=No).

Online	independent	Customers use Internet banking facilities (1=Yes, 0=No).
Credit Card	independent	Customers use a credit card issued by Universal Bank (1=Yes, 0=No).

3.2 Data Pre-processing

Preparing the raw data to be usable for a machine learning model. This step includes deleting unnecessary columns, checking missing values, handling data types, exploring and managing unexpected values, handling columns on the same period from monthly to annually, encoding categorical fields, and splitting the dataset.

In this dataset, these columns for ID and Zip code have no impact on the target. So, they were removed from the dataset using Pandas function **DataFrame.drop()**

3.2.1. Checking Missing Values

For identifying missing values in data, Pandas has the **isna()** and **isnull()** functions. The number of missing values in each column was returned using the pandas' method **DataFrame.isnull().sum()**. Then, all columns return zeros.

3.2.2. Handling Data Types

The dataset has values with int type in fields which should be categorical like

'Personal Loan', 'Securities Account', 'Family', 'CD Account', 'Online', 'Credit Card', 'Education'.

The types should be changed because the **int** type may affect the results.

3.2.3. Exploring and Handling Unexpected Values

During exploring dataset statistical features, the minimum value of Experience was -3 which is irrelevant because there is no negative experience. This issue was handled by replacing every negative value in this column with the mean using **DataFrame.mean()**

3.2.4. Handling Column on the Same Period from Monthly to Annually

CCAvg is the average spending on credit cards per month & the Income is per year so should convert one to another. So, a new column (ann_CC) was created instead of CCAvg which is just average monthly spending multiplied by twelve.

3.2.5. Encoding Categorical Data

pandas.get_dummies() had been used for encoding Family and Education and adding the new data frame to the using **pandas.concat([old_data,dummy_data])** then, the old columns are dropped to avoid duplicates using **DataFrame.drop()**.

Encoding helps the models in fitting because models can not fit on categorical therefore categorical data has to be encoded.

3.2.6. Splitting the Dataset

Any model must be tested so, in this section the data was split into two parts train and test using sklearn.model_selection.**train_test_split()**about (4200 rows) 80% of the dataset was for training and (eight hundred rows) about 20% for testing the models.

3.3 Exploratory Data Analysis (EDA)

In statistics, exploratory data analysis (EDA) is a way of examining data sets to highlight their key features, frequently utilizing statistical graphics and other techniques for data visualization in this section. Matplotlib and Seaborn had been used and the rest of EDA had been made in a dashboard.

3.3.1. Correlations between Numeric Features

seaborn.**heatmap**() shows that 'Age' and 'Experience' are correlated with each other perfectly. Therefore, one of them must be dropped and 'Experience' has been dropped. 'Income' and ‘ann_CC' correlated with each other.

3.3.2. Features Distribution

seaborn.**distplot()** was used to show the distribution of the continuous feature.

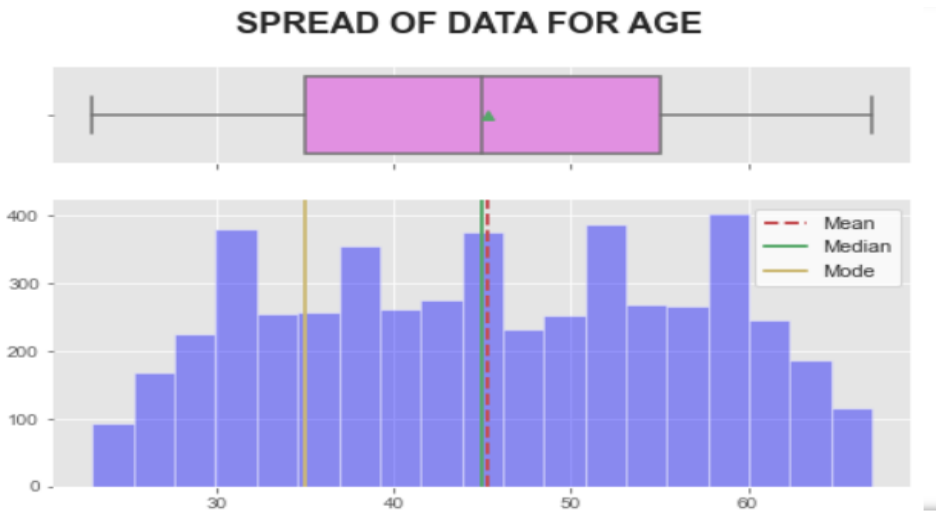


Figure 3-2: Age Distribution

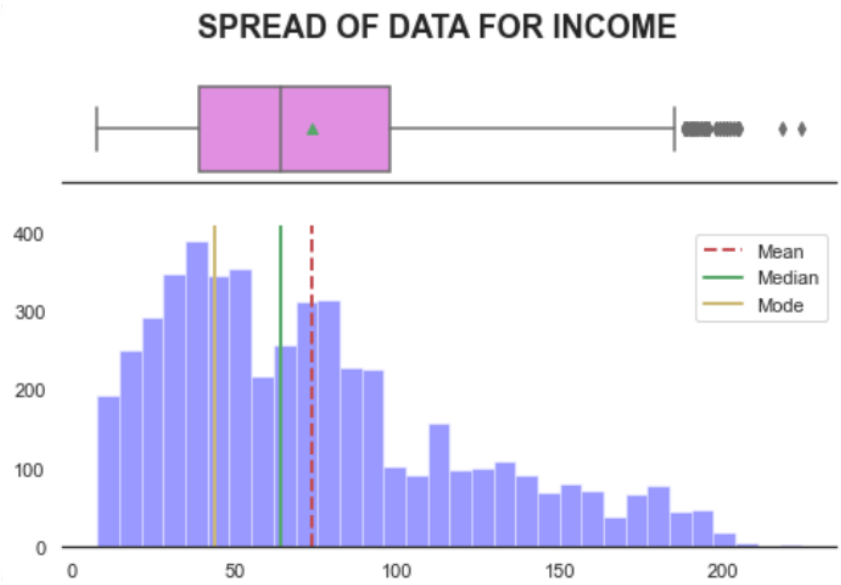


Figure 3-3: Income Distribution

Age has no outliers, and the median equals the mean. Annual spending and annual income are right skewed with some outliers. The mortgage is about zeros and outliers.

seaborn.**countplot()** was used to show the categorical feature regarding personal loans. Figure 3.4 shows people who accept loans for about 10% of all clients.

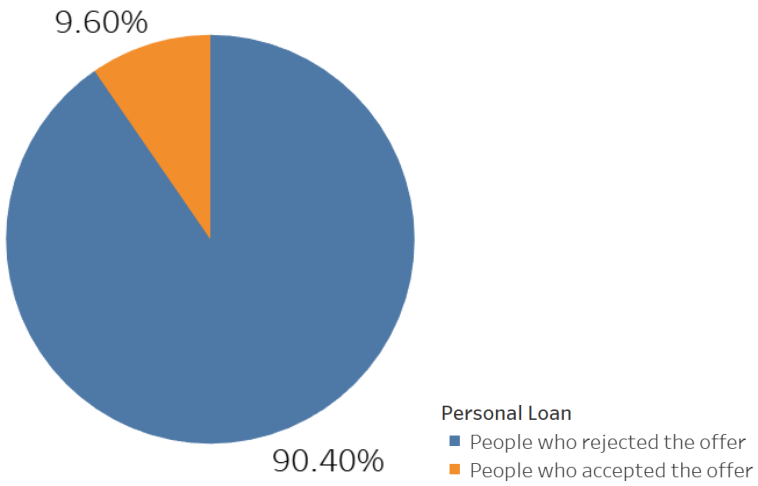


Figure 3-4: People's total responses to the offer

Family numbers have a slight effect on the acceptance of personal loans. Education also has its own effect on the higher education level accepting more than the lower for personal loans. Securities account, CD account, Credit Card, and Online have their effects on personal loans.

3.4 Modeling and Evaluation

For choosing the right parameters for every algorithm sklearn.model_selection. **GridSearchCV()** had been used for scoring recall.

3.5.1. SVM

sklearn.svm.**SVC()** was applied to this model. GridsearchCV chooses a linear kernel that makes a higher recall with 0.649, and by customization for the other parameters C = 10 and gamma = 'scale' the recall is up to 0.65897 and Training Accuracy: 0.9615& Testing Accuracy:0.965.

Table 3-2 is showing the confusion matrix of SVM algorithm

903	7
28	62

3.5.2. Decision Tree

Decision tree was Applied by Using sklearn.tree.**DecisionTreeClassifier()**. GridsearchCV chooses criterion='gini', max depth=20, splitter='best', and after this combination the training Accuracy: 1.0, Testing Accuracy: 0.979, and Recall Accuracy: 0.9.

Table 3-3 is showing the confusion matrix of Decision tree algorithm

898	12
9	81

3.5.3. Random Forest

Random Forest was Applied by Using sklearn.ensemble.**RandomForestClassifier()**. GridsearchCV chooses criterion = 'gini', max_depth = 20, n_estimators = 100, this combination gets Training Accuracy: 1.0, Testing Accuracy: 0.988, and recall Accuracy: 0.9111

Table 3-4 is showing the confusion matrix of Random Forest algorithm

906	4
8	82

3.5.4. Gradient Boosting

Gradient Boosting was Applied by Using sklearn.ensemble.**GradientBoostingClassifier()**. GridsearchCV chooses criterion = 'friedman_mse', loss = 'deviance', 'n_estimators': 1000 this combination gets Training Accuracy:1.0, Testing Accuracy: 0.993, and recall Accuracy: 0.9222

Table 3-5 is showing the confusion matrix of Gradient Boosting algorithm

910	0
7	83

3.5.5. Neural Network

Neural Network was Applied by Using sklearn.neural_network.**MLPClassifier().**GridsearchCV chooses activation = 'logistic', batch_size =10, hidden_layer_sizes = 100, learning_rate = 'invscaling', solver = 'adam' this combination gets Training Accuracy: 0.989

Testing Accuracy: 0.978, recall Accuracy: 0.8778

Table 3-6 is showing the confusion matrix of Neural Network algorithm

899	11
11	79

4. Results and Discussion

4.1 Introduction

Chapter 4 discusses the results of preprocessing and EDA in the previous chapter. Then, comparing the models and determining the champion model.

4.2 Preprocessing Results

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Age                 5000 non-null   int64
 1   Experience          5000 non-null   int64
 2   Income              5000 non-null   int64
 3   Family              5000 non-null   category
 4   CCAvg               5000 non-null   float64
 5   Education           5000 non-null   category
 6   Mortgage            5000 non-null   int64
 7   Personal_Loan       5000 non-null   category
 8   Securities_Account  5000 non-null   category
 9   CD_Account          5000 non-null   category
 10  Online              5000 non-null   category
 11  CreditCard          5000 non-null   category
dtypes: category(7), float64(1), int64(4)
memory usage: 230.6 KB
```

Figure 4-1: Data Information

	Age	Experience	Income	CCAvg	Mortgage
count	5000.00000	5000.00000	5000.00000	5000.00000	5000.00000
mean	45.33840	20.10460	73.77420	1.93794	56.49880
std	11.46317	11.46795	46.03373	1.74766	101.71380
min	23.00000	-3.00000	8.00000	0.00000	0.00000
25%	35.00000	10.00000	39.00000	0.70000	0.00000
50%	45.00000	20.00000	64.00000	1.50000	0.00000
75%	55.00000	30.00000	98.00000	2.50000	101.00000
max	67.00000	43.00000	224.00000	10.00000	635.00000

Figure 4-2: Continues Feature Descriptive data

The dataset shape is twelve columns and five thousand observations (after doing preprocessing) as showed in figure 4-1, and all column types are integers where it should be categorical. There are no null values in any columns. Also, in the data description Experience minimum value showed in figure 4-2 was -3 which is irrational.

4.3 EDA Results

Figure 4-3: Features Correlations

Figure 4-3 shows there is a perfect positive relationship between Experience and Age. There is a positive relationship (r = 0.98) between income and average annual spending.

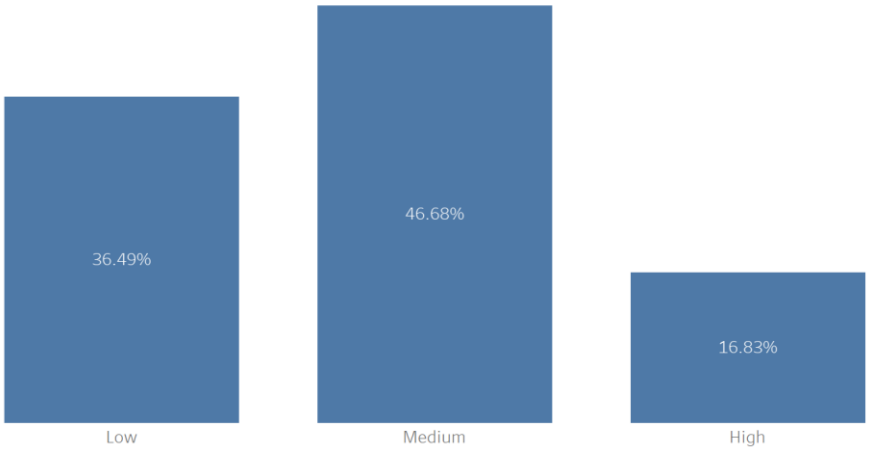


Figure 4-4:Income Levels

Income can be divided into three groups low, medium, and high, and about 47% of bank users' income is in the medium level, 36% in the low level, and 17% in the high level as shown in figure 4-4.

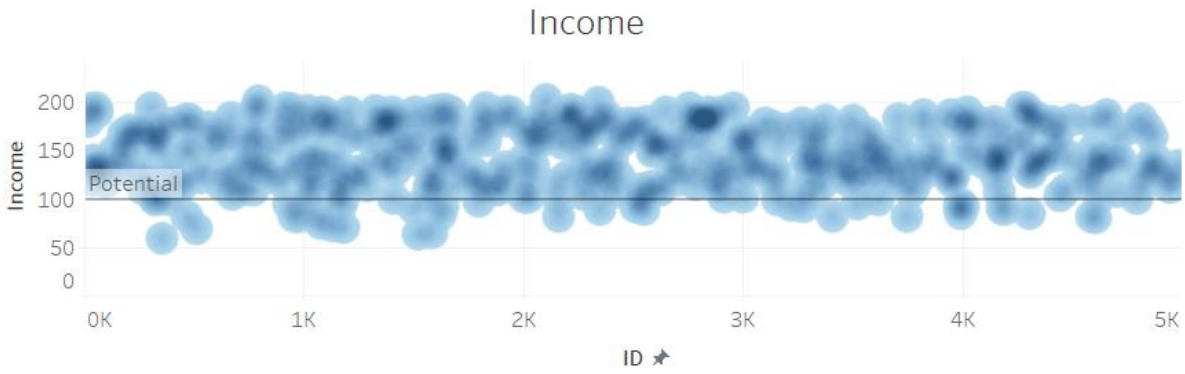


Figure 4-5: Income of clients who Accepted

Figure 4-5 shows the income of clients who agreed to take loans. After an annual income of 100K, the client is more likely to accept a personal loan.

Table 4-1 shows about 30% of all users are singles, 60% of bank users use online services, and 42% of the bank clients are undergraduates.

Table 4-1:Clients Family Number

Family Number	percentage
1	29.44%
2	25.92%
3	20.20%
4	24.44%

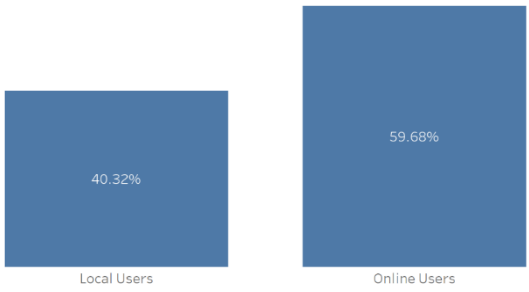


Figure 4-6: Local Users VS Online Users

Figure 4-6 shows online users of the bank take about 60% of all clients and the rest of users are Local users.

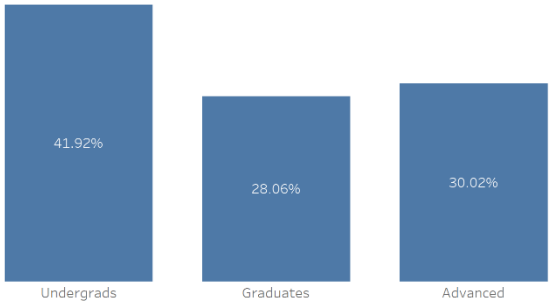


Figure 4-7: Education Distribution

Education Distribution is shown in figure 4-7. Undergrads are about 42% from all users and Advanced about 30% and 28% of all clients are Graduates.

4.4 Comparing Results

Comparing different algorithms' needs evaluation criteria. Therefore, four evaluation equations have been used in this paper. Equation 4-1 showing accuracy, equation 4-2 showing Precision, equation 4-3 is showing Recall, and equation 4-4 is showing F1.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (4\text{-}1)$$

$$Precision = \frac{TP}{TP+FP} \qquad (4\text{-}2)$$

$$Recall = \frac{TP}{TP+FN} \qquad (4\text{-}3)$$

$$F1 = 2 \,.\, \frac{Precision.Recall}{Precision+Recall} \qquad (4\text{-}4)$$

Where:

• TP is the number of true positives. It tells observations that are part of the positive class that predicted correctly.

• TN is the number of true negatives. It tells observations that are part of the negative class that predicted correctly.

• FP is the number of false positives which is also called a Type I error. It tells observations predicted to be part of the positive class that is part of the

negative class.

• FN is the number of false negatives which is also called a Type II error. It tells observations predicted to be part of the negative class that are part of the positive class.

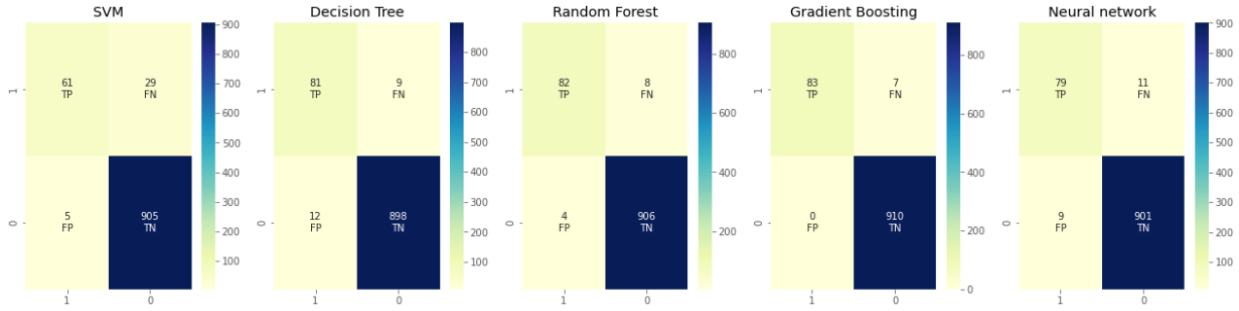


Figure 4-8: Comparison between all algorithm

Figure 4-8 shows the confusion matrix for every model and the best is Gradient Boosting which has gotten the highest true positive eighty-three, the highest true negative 910, and just seven false negatives with no false positives. This model comes in first place regarding the precision, recall and f1 score with 0.97 precision, 0.95 recall and f1 score 0.96.

Table 4-2 shows the descending scores of every model

Table 4-2:Comparison report between Models

Rank	Model	Accuracy	Precision	Recall	F1-Score
1	Gradient Boosting	0.99	1.00	0.96	0.98
2	Random Forest	0.99	0.97	0.95	0.96
3	Neural Network	0.98	0.94	0.93	0.94
4	Decision Tree	0.98	0.93	0.94	0.94
5	SVM	0.97	0.95	0.84	0.88

4.5 Optimization (SMOTE oversample)

The target column in the dataset was unbalanced, meaning that only a small percentage of the observations were labeled as positive (accepted loans) while the majority were labeled as negative (rejected loans). To address this issue, the imblearn library's SMOTE (Synthetic Minority Over-sampling Technique) method was used to oversample the minority class (accepted loans) in

order to balance the target column. After oversampling, the five models were retested to evaluate their performance on the balanced dataset..

4.6 Champion Model

The new results after resampling data were good for Decision Tree and SVM and slightly negative for Neural Network and no effect for Gradient Boosting and Random Forest, where the recall of decision tree increased to 0.95, recall of SVM increased to 0.90 and recall of neural network decreased to 0.92.

Table 4-3: Recall Comparison between Models before and after Optimization

Rank	Model	Recall Before	Recall After
1	Gradient Boosting	0.96	0.96
2	Random Forest	0.95	0.95
3	Neural Network	0.93	0.92
4	Decision Tree	0.94	0.95
5	SVM	0.84	0.90

Hence, Gradient Boosting is the Champion model for predicting acceptance of personal loans.

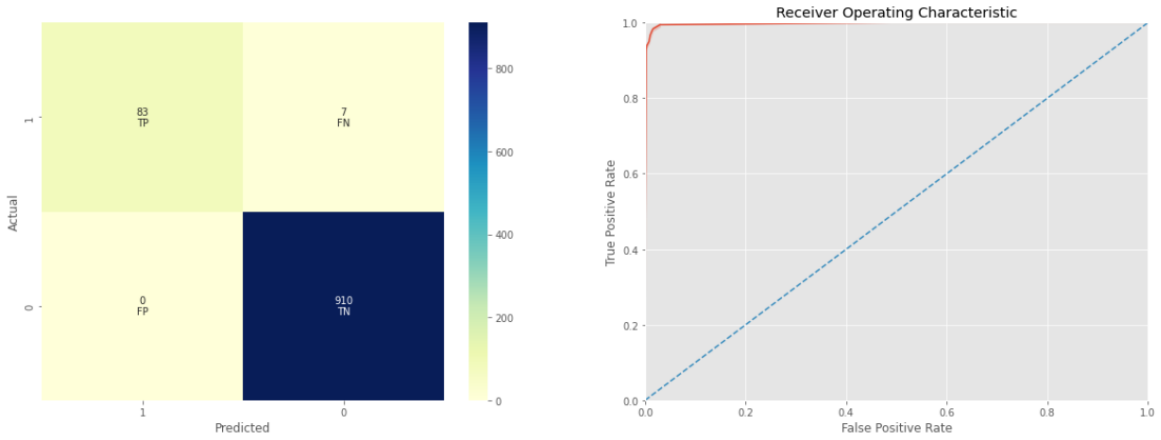


Figure 4-9: Confusion Metrics and AUC for the champion mode

4.7 Dashboard

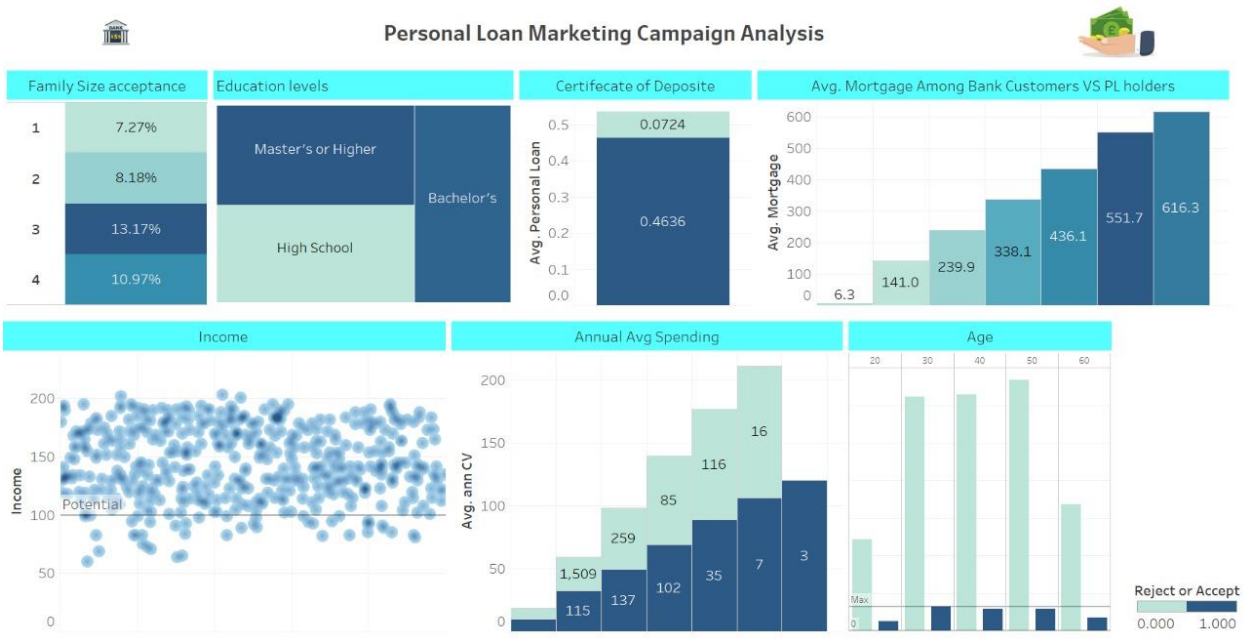


Figure 4-10: Personal Loans Marketing Campaign Analysis Dashboard

Figure 4-10 shows relationships between features and the response to the offers to take loans. Here, Profiling the perfect client can be conducted by selecting a client whose family contains three members and this income exceed a hundred thousand per year with high average

mortgages. It will be better if the client has master's or higher degree and taking certificate of deposits

Chapter 5

Conclusion

5.0. Introduction

Chapter 5 provides an overview of the study, summarizing the empirical results, potential implications, and study limitations. The empirical results are based on the classifier performance in chapter 4. Potential implications discuss why this study is important. Limitations discuss the study's flaws.

5.1. Summary

In conclusion, the banking sector is primarily oriented towards increasing profit with decreasing risks. Therefore, they offer personal loans to people who have deposits in the bank because they have no risk in paying loans back, but most offers are rejected. This paper tried to predict clients who will accept this offer using demographic data and financial data. In addition to finding our high-profile client. The sample dataset contained five thousand observations and thirteen features.

Before applying the algorithms, preprocessing has been done, and EDA. Missing values there were not any nulls, handling datatypes from integer to category, exploring and handling unexpected values to be in the same period, encoding categorical data, splitting the data into two training and testing. Then, exploring correlations between features and feature distribution. After that models have been fitted, tested, tuned, and retched the champion model which is Gradient Boosting.

The first research question is ‘what is the optimal machine learning algorithm that can classify personal loan clients?’ Gradient Boosting is the champion model in this study. It predicts responds of the clients with 99% Accuracy.

Second research question is ‘What are the highest features that indicate the clients may accept the loans?’ Customers with Higher income have higher mortgages and higher monthly average spending. They also have a certificate of deposit with the bank. They are our high-profile clients.

5.2. Limitations

This paper faced limitations regarding the dataset because the sample was just five thousand observations. The target column (Personal Loan) is not balanced therefore SMOT oversampling is used for handling this issue.

Regarding income and spending columns peoples have accounts in different banks then, the true income and spending will be shown clearly.

5.3. Potential Implications

This paper aids sales teams in the bank department to save time and money by highlighting the potential clients who are more likely to accept the offer of taking a personal loan. However, this paper is introduced to the banking sector also, it can be used as an example of machine learning usage in the finance sector such as investment companies who wants to provide loans offers to the investors who have high credit risks.

References

Abbasniya, M. R., Sheikholeslamzadeh, S. A., Nasiri, H., & Emami, S. (2022). Classification of Breast Tumors Based on Histopathology Images Using Deep Features and Ensemble of Gradient Boosting Methods. *Computers and Electrical Engineering*, *103*. https://doi.org/10.1016/J.COMPELECENG.2022.108382

Agarwal, K., Jain, M., & Kumawat, A. (2022). Comparing Classification Algorithms on Predicting Loans. *Lecture Notes in Networks and Systems*, *303 LNNS*, 240–249. https://doi.org/10.1007/978-3-030-86223-7_21

Al-Manaseer, H., Abualigah, L., Alsoud, A. R., Zitar, R. A., Ezugwu, A. E., & Jia, H. (2023). A Novel Big Data Classification Technique for Healthcare Application Using Support Vector Machine, Random Forest and J48. *Studies in Computational Intelligence*, *1071*, 205–215. https://doi.org/10.1007/978-3-031-17576-3_9

Duggirala, A. (2020). Bank Loan Personal Modelling using Classification Algorithms of Machine Learning. *International Journal of Scientific Research*, *8*, 178–183.

Esteki, S., & Naghsh-Nilchi, A. R. (2022). Frequency component Kernel for SVM. *Neural Computing and Applications*, *34*(24), 22449–22464. https://doi.org/10.1007/S00521-022-07632-4

Fan, Z., Yang, Q., Xu, Z., Sun, K., Yang, M., Yin, R., Zhao, D., Fan, J., Ma, H., Shen, Y., & Xia, H. (2022). Construct a classification decision tree model to select the optimal equation for estimating glomerular filtration rate and estimate it more accurately. *Scientific Reports*, *12*(1). https://doi.org/10.1038/S41598-022-19185-6

Galindo, J., & Tamayo, P. (2000). Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications. *Computational Economics*, *15*(1–2), 107–143. https://doi.org/10.1023/A:1008699112516

Gupta, D., Richhariya, B., & Borah, P. (2019). A fuzzy twin support vector machine based on information entropy for class imbalance learning. *Neural Computing and Applications*, *31*(11), 7153–7164. https://doi.org/10.1007/S00521-018-3551-9

Ismaeil, H., Kholeif, S., & Abdel-Fattah, M. A. (2022). Using Decision Tree Classification Model to Predict Payment Type in NYC Yellow Taxi. *International Journal of Advanced Computer Science and Applications*, *13*(3), 238–244. https://doi.org/10.14569/IJACSA.2022.0130330

Jai Vignesh, P. S., Kaliswar Adhish, R., Rithik, R., Sanjeev, S., & Rajesh, C. B. (2023). *Model Validation to Enhance Precision Agriculture Using DeepDream and Gradient Mapping Techniques*. 359–372. https://doi.org/10.1007/978-981-19-4960-9_28

Ke, T., Liao, Y., Wu, M., Ge, X., Huang, X., Zhang, C., & Li, J. (2023). Maximal margin hyper-sphere SVM for binary pattern classification. *Engineering Applications of Artificial Intelligence*, *117*. https://doi.org/10.1016/J.ENGAPPAI.2022.105615

Lee, Y. H., Won, J. H., Kim, S., Auh, Q. S., & Noh, Y. K. (2022). Advantages of deep learning with convolutional neural network in detecting disc displacement of the temporomandibular joint in

magnetic resonance imaging. *Scientific Reports*, *12*(1). https://doi.org/10.1038/S41598-022-15231-5

Marois, T. (2022). A Dynamic Theory of Public Banks (and Why it Matters). *Review of Political Economy*, *34*(2), 356–371. https://doi.org/10.1080/09538259.2021.1898110

Muksin, U., Riana, E., Rudyanto, A., Bauer, K., Simanjuntak, A. V. H., & Weber, M. (2023). Neural network-based classification of rock properties and seismic vulnerability. *Global Journal of Environmental Science and Management*, *9*(1), 15–30. https://doi.org/10.22034/GJESM.2023.01.02

Rathore, R., Kumar, P., & Singhi, R. (2023). Decision Tree Algorithm for Diagnosis and Severity Analysis of COVID-19 at Outpatient Clinic. *Lecture Notes in Networks and Systems*, *421*, 163–178. https://doi.org/10.1007/978-981-19-1142-2_13

Shahsavar, A., Goodarzi, A., Baniasad Askari, I., Jamei, M., Karbasi, M., & Afrand, M. (2022). The entropy generation analysis of the influence of using fins with tip clearance on the thermal management of the batteries with phase change material: Application a new gradient-based ensemble machine learning approach. *Engineering Analysis with Boundary Elements*, *140*, 432–446. https://doi.org/10.1016/J.ENGANABOUND.2022.04.024

Uttam, A. K., & Mangal, A. (2020). Application of extreme gradient boosting ensemble model for sleep quality prediction on personalized wearable device data. *International Journal of Advanced Science and Technology*, *29*(5), 3755–3762.

Vadivukkarasi, S., & Santhi, S. (2021). A novel hybrid learning based Ada Boost (HLBAB) classifier for channel state estimation in cognitive networks. *International Journal of Dynamics and Control*, *9*(1), 299–307. https://doi.org/10.1007/S40435-020-00633-Y

Venkateswarlu, T., & Anmala, J. (2023). Application of Random Forest Model in the Prediction of River Water Quality. *Lecture Notes in Networks and Systems*, *447*, 525–535. https://doi.org/10.1007/978-981-19-1607-6_47

Walker, J., Questa, H., Raman, A., Ahmed, M., Mohammadpour, M., Bewsher, S. R., & Offner, G. (2023). Application of Tribological Artificial Neural Networks in Machine Elements. *Tribology Letters*, *71*(1). https://doi.org/10.1007/S11249-022-01673-5

Xing, Z., He, Y., Chen, J., Wang, X., & Du, B. (2023). Health evaluation of power transformer using deep learning neural network. *Electric Power Systems Research*, *215*. https://doi.org/10.1016/J.EPSR.2022.109016

Zeng, Q., Qing, Z., Zhu, M., Zhang, F., Wang, H., Liu, Y., Shi, Z., & Yu, Q. (2022). Application of Random Forest Algorithm on Tornado Detection. *Remote Sensing*, *14*(19). https://doi.org/10.3390/RS14194909

Appendix I

Personal Loans Predictive Modelling Using Machine Learning Techniques 🏦 💰 💸

Imports

In [7]:
```
import scipy.stats as stats
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
import statsmodels.api as sm
```

In [8]:
```
#--Sklearn library--
# Sklearn package's randomized data splitting function
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV #for choosing best parameters
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn import metrics
#AUC ROC curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve

from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay #to plot confusion matric
from sklearn.metrics import plot_confusion_matrix

from sklearn.svm import SVC#to build the model
from sklearn.tree import DecisionTreeClassifier #to build the model
from sklearn.ensemble import RandomForestClassifier#to build the model
from sklearn.ensemble import GradientBoostingClassifier#to build the model
from sklearn.neural_network import MLPClassifier#to build the model
from imblearn.over_sampling import SMOTE #handling immbalance data
```

In [9]:
```
pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.max_rows', 300)
pd.set_option('display.max_colwidth',400)
pd.set_option('display.float_format', lambda x: '%.5f' % x)
# To supress numerical display in scientific notations
warnings.filterwarnings('ignore') # To supress warnings
 # set the background for the graphs
plt.style.use('ggplot')
```

Data Collection

In [10]:
```
# read the data using pd.read_csv and converting it to dataframe
data = pd.read_csv("Bank_Personal_Loan_Modelling.csv")
data
```

Out[10]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.60000	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.50000	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.00000	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.70000	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.00000	2	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	4996	29	3	40	92697	1	1.90000	3	0	0	0	0	1	0
4996	4997	30	4	15	92037	4	0.40000	1	85	0	0	0	1	0
4997	4998	63	39	24	93023	2	0.30000	3	0	0	0	0	0	0
4998	4999	65	40	49	90034	3	0.50000	2	0	0	0	0	1	0
4999	5000	28	4	83	92612	3	0.80000	1	0	0	0	0	1	1

5000 rows × 14 columns

Data Prepocessing

In [11]:
```
data.shape
```

Out[11]:
```
(5000, 14)
```

In [12]:
```
data.head(5)
```

Out[12]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.60000	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.50000	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.00000	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.70000	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.00000	2	0	0	0	0	0	1

In [13]:
```
data.tail(5)
```

Out[13]:

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
4995	4996	29	3	40	92697	1	1.90000	3	0	0	0	0	1	0
4996	4997	30	4	15	92037	4	0.40000	1	85	0	0	0	1	0
4997	4998	63	39	24	93023	2	0.30000	3	0	0	0	0	0	0
4998	4999	65	40	49	90034	3	0.50000	2	0	0	0	0	1	0
4999	5000	28	4	83	92612	3	0.80000	1	0	0	0	0	1	1

In [14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   ID                  5000 non-null   int64  
 1   Age                 5000 non-null   int64  
 2   Experience          5000 non-null   int64  
 3   Income              5000 non-null   int64  
 4   ZIP Code            5000 non-null   int64  
 5   Family              5000 non-null   int64  
 6   CCAvg               5000 non-null   float64
 7   Education           5000 non-null   int64  
 8   Mortgage            5000 non-null   int64  
 9   Personal Loan       5000 non-null   int64  
 10  Securities Account  5000 non-null   int64  
 11  CD Account          5000 non-null   int64  
 12  Online              5000 non-null   int64  
 13  CreditCard          5000 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 547.0 KB
```

In [15]:
```
data.rename(columns={"ZIP Code":"ZIP_Code","Personal Loan":"Personal_Loan","Securities Account":"Securities_Account","CD Account"
data.columns
```

Out[15]:
```
Index(['ID', 'Age', 'Experience', 'Income', 'ZIP_Code', 'Family', 'CCAvg',
       'Education', 'Mortgage', 'Personal_Loan', 'Securities_Account',
       'CD_Account', 'Online', 'CreditCard'],
      dtype='object')
```

In [16]:
```
data.drop(['ID','ZIP_Code'],axis=1,inplace=True) #droping id & zip code
```

In [17]:
```
data.isnull().sum()
```

Out[17]:
```
Age                   0
Experience            0
Income                0
Family                0
CCAvg                 0
Education             0
Mortgage              0
Personal_Loan         0
Securities_Account    0
CD_Account            0
Online                0
CreditCard            0
dtype: int64
```

In [18]:
```
# converting categorical varaible to category type
category_col = ['Personal_Loan', 'Securities_Account','Family', 'CD_Account', 'Online', 'CreditCard', 'Education']
data[category_col] = data[category_col].astype('category')
```

In [19]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype   
---  ------              --------------  -----   
 0   Age                 5000 non-null   int64   
 1   Experience          5000 non-null   int64   
 2   Income              5000 non-null   int64   
 3   Family              5000 non-null   category
 4   CCAvg               5000 non-null   float64 
 5   Education           5000 non-null   category
 6   Mortgage            5000 non-null   int64   
 7   Personal_Loan       5000 non-null   category
 8   Securities_Account  5000 non-null   category
 9   CD_Account          5000 non-null   category
 10  Online              5000 non-null   category
 11  CreditCard          5000 non-null   category
dtypes: category(7), float64(1), int64(4)
memory usage: 230.6 KB
```

In [20]:
```
data.describe()
```

Out[20]:

	Age	Experience	Income	CCAvg	Mortgage
count	5000.00000	5000.00000	5000.00000	5000.00000	5000.00000
mean	45.33840	20.10460	73.77420	1.93794	56.49880
std	11.46317	11.46795	46.03373	1.74766	101.71380
min	23.00000	-3.00000	8.00000	0.00000	0.00000
25%	35.00000	10.00000	39.00000	0.70000	0.00000
50%	45.00000	20.00000	64.00000	1.50000	0.00000
75%	55.00000	30.00000	98.00000	2.50000	101.00000
max	67.00000	43.00000	224.00000	10.00000	635.00000

In [21]:
```
'in Experience columns there are negative values (-3) which should not to be cause there is no negative experience:)'
```

Out[21]:
```
'in Experience columns there are negative values (-3) which should not to be cause there is no negative experience:)'
```

In [22]:
```
data['Experience'][data['Experience'] < 0] = data['Experience'].mean()
data.describe()
```

Out[22]:

	Age	Experience	Income	CCAvg	Mortgage
count	5000.00000	5000.00000	5000.00000	5000.00000	5000.00000
mean	45.33840	20.32869	73.77420	1.93794	56.49880
std	11.46317	11.25301	46.03373	1.74766	101.71380
min	23.00000	0.00000	8.00000	0.00000	0.00000
25%	35.00000	11.00000	39.00000	0.70000	0.00000
50%	45.00000	20.10460	64.00000	1.50000	0.00000
75%	55.00000	30.00000	98.00000	2.50000	101.00000
max	67.00000	43.00000	224.00000	10.00000	635.00000

In [23]:
```
' CCAvg is the average spending on credit cards per month & the Income is per year so we have to convert one to another.'
```

Out[23]:
```
' CCAvg is the average spending on credit cards per month & the Income is per year so we have to convert one to another.'
```

In [24]:
```
data['ann_CC'] = data['CCAvg'] * 12
data.drop('CCAvg', axis = 1, inplace = True)
```

In [25]:
```
df_loan = data
```

In [26]:
```
df = pd.get_dummies(data[['Family','Education']])#dummy catigorical fields
data = pd.concat([data, df], axis=1)
data.drop(['Family','Education'],axis=1,inplace=True)
data.columns
```

Out[26]:
```
Index(['Age', 'Experience', 'Income', 'Mortgage', 'Personal_Loan',
       'Securities_Account', 'CD_Account', 'Online', 'CreditCard', 'ann_CC',
       'Family_1', 'Family_2', 'Family_3', 'Family_4', 'Education_1',
       'Education_2', 'Education_3'],
      dtype='object')
```

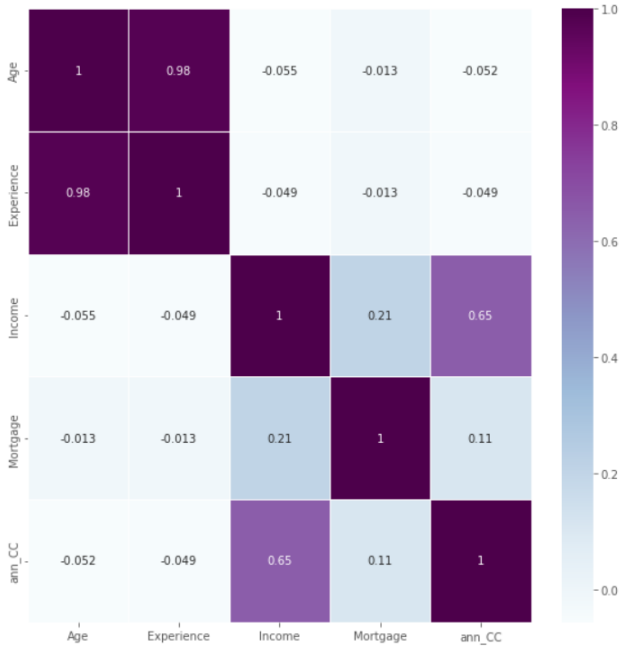
Spliting the data into training and test set in the ratio of 80:20

In [27]:
```
X = data.drop('Personal_Loan', axis = 1).values #splitting dataset 80:20
y = data['Personal_Loan'].values.reshape((-1, 1))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
```

EDA

In [28]:
```
fig, ax = plt.subplots(figsize = (10, 10))
sns.heatmap(df_loan.corr(),cmap='BuPu',cbar=True,annot=True,linewidths=0.8)
plt.show
```

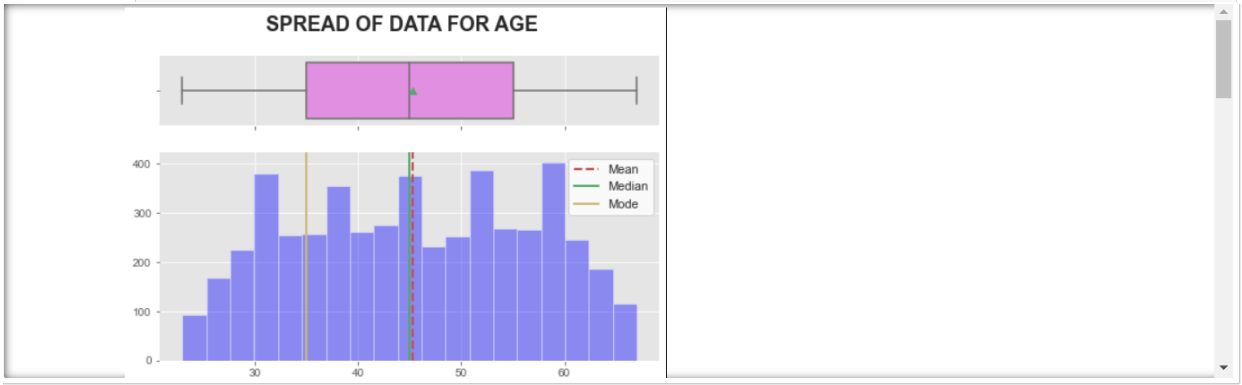
Out[28]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

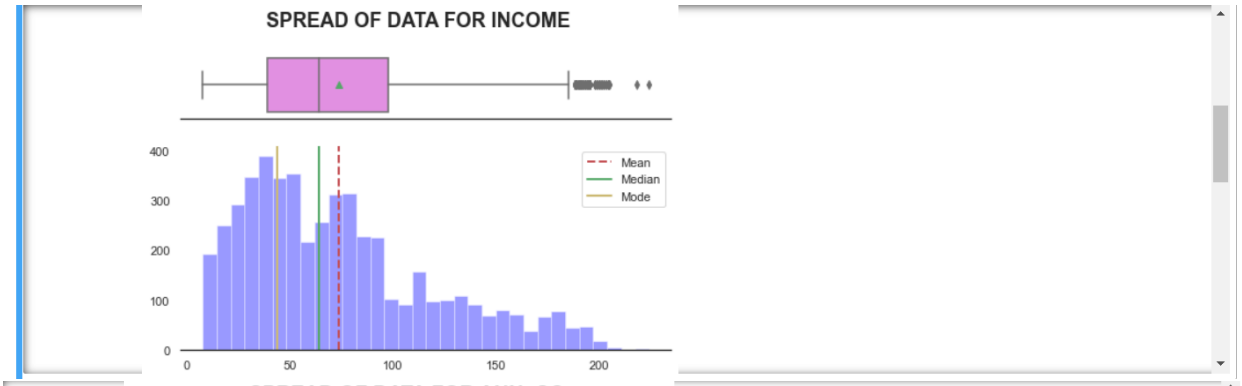


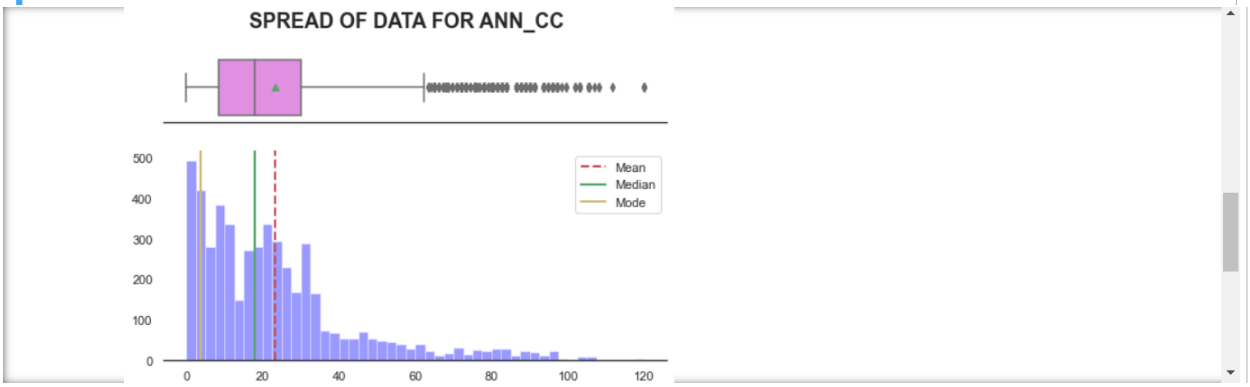
In [29]:
```
data.drop(['Experience'],axis=1,inplace=True)
```

In [30]:
```
def dist_box(data):
 # function plots a combined graph for univariate analysis of continous variable
 #to check spread, central tendency , dispersion and outliers
    Name=data.name.upper()
    fig,(ax_box,ax_dis)  =plt.subplots(nrows=2,sharex=True,gridspec_kw = {"height_ratios": (.25, .75)},figsize=(8, 5))
    mean=data.mean()
    median=data.median()
    mode=data.mode().tolist()[0]
    sns.set_theme(style="white")
    fig.suptitle("SPREAD OF DATA FOR "+ Name  , fontsize=18, fontweight='bold')
    sns.boxplot(x=data,showmeans=True, orient='h',color="violet",ax=ax_box)
    ax_box.set(xlabel='')
     # just trying to make visualisation better. This will set background to white
    sns.despine(top=True,right=True,left=True) # to remove side line from graph
    sns.distplot(data,kde=False,color='blue',ax=ax_dis)
    ax_dis.axvline(mean, color='r', linestyle='--',linewidth=2)
    ax_dis.axvline(median, color='g', linestyle='-',linewidth=2)
    ax_dis.axvline(mode, color='y', linestyle='-',linewidth=2)
    plt.legend({'Mean':mean,'Median':median,'Mode':mode})
```

In [31]:
```
#select all quantitative columns for checking the spread
list_col=  ['Age','Income','ann_CC','Mortgage']
for i in range(len(list_col)):
    dist_box(data[list_col[i]])
```





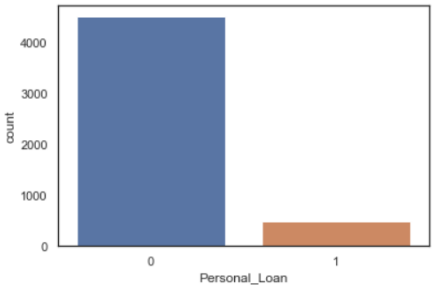


In [32]:
```
data["Personal_Loan"].value_counts()
```

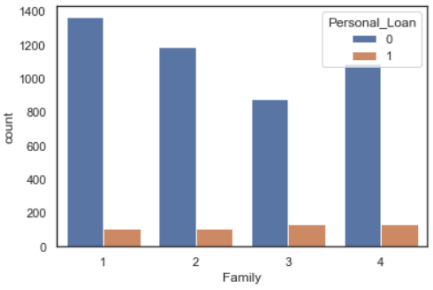
Out[32]:
```
0    4520
1     480
Name: Personal_Loan, dtype: int64
```

In [ ]:
```

```

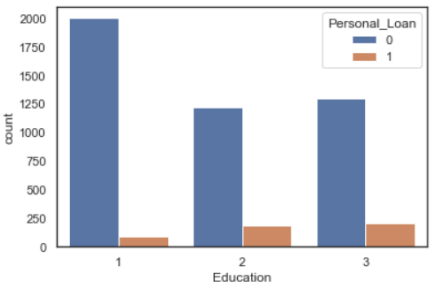
In [40]:
```
#checking class skewness (data imbalance)
p = sns.countplot(x=data['Personal_Loan'])
```



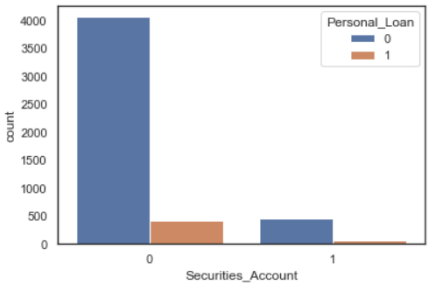
In [29]:
```
sns.countplot(x='Family', hue = 'Personal_Loan', data = df_loan);
```



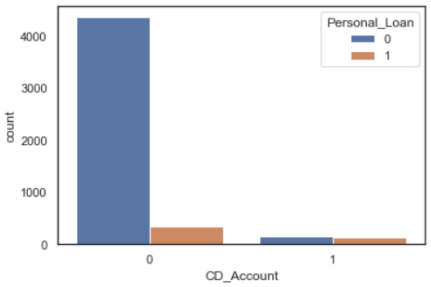
In [30]:
```
sns.countplot(x='Education', hue = 'Personal_Loan', data = df_loan);
```



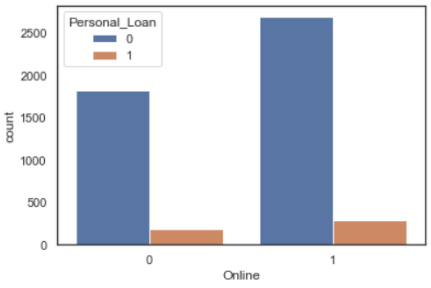
In [31]:
```
sns.countplot(x='Securities_Account', hue = 'Personal_Loan', data = df_loan);
```



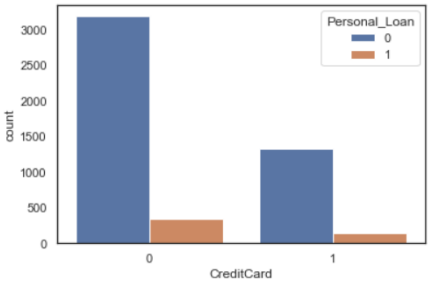
In [32]:
```
sns.countplot(x='CD_Account', hue = 'Personal_Loan', data = df_loan);
```



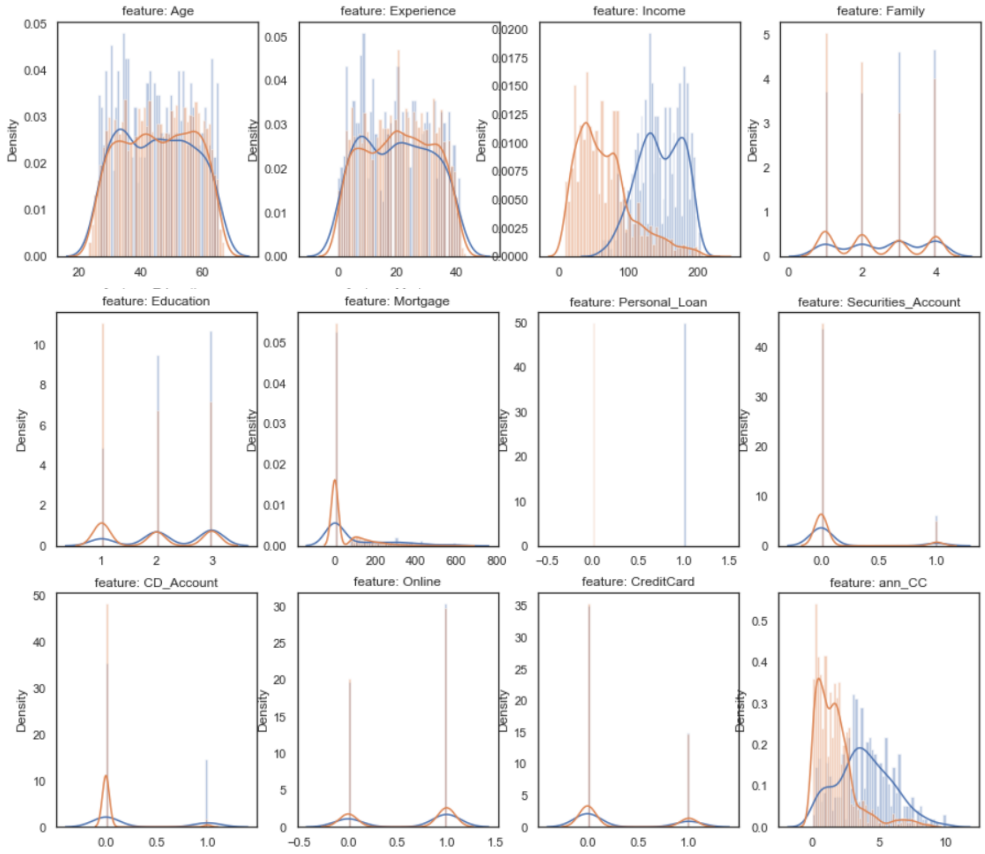
In [33]:
```
sns.countplot(x='Online', hue = 'Personal_Loan', data = df_loan);
```



In [34]:
```
sns.countplot(x='CreditCard', hue = 'Personal_Loan', data = df_loan);
```



In [35]:
```
import matplotlib.gridspec as gridspec
v_features = df_loan.columns
plt.figure(figsize=(15,8*4))
gs = gridspec.GridSpec(7, 4)
for i, cn in enumerate(df_loan[v_features]):
    ax = plt.subplot(gs[i])
    sns.distplot(df_loan[cn][df_loan.Personal_Loan == 1], bins=50)
    sns.distplot(df_loan[cn][df_loan.Personal_Loan == 0], bins=50)
    ax.set_xlabel('')
    ax.set_title('feature: ' + str(cn))
plt.show()
```



In [62]:
```
conf_matrix_all = {}

def personal_loan_prediction(name, algo, training_x, testing_x, training_y, testing_y, plot) :
    algo.fit(training_x,training_y)                          # Fit the training data set to the algorithm passed.
    predictions = algo.predict(testing_x)                    # Get all predictions
    probabilities = algo.predict_proba(testing_x)            # Get probablities of predictions

    conf_matrix = confusion_matrix(testing_y, predictions)   # Get confusion matrix using the predictions
    tn, fp, fn, tp = conf_matrix.ravel()

    conf_matrix_all[name] = conf_matrix                      # Save confusion matrix values to a dictionary

    print("Classification report:")                          # Print the classification report
    print(classification_report(testing_y, predictions))

    model_roc_auc = roc_auc_score(testing_y, predictions)          # Get the Area under the curve number
    fpr,tpr,thresholds = roc_curve(testing_y, probabilities[:,1])  # Get False postive rate and true positive rate

    print ("Area under the curve: ", model_roc_auc)


    if plot:
        fig, axes = plt.subplots(1,2, figsize=(20, 7))
        conf_matrix = np.flip(conf_matrix)

        labels = np.array([['\nTP','\nFN'],['\nFP','\nTN']])
        labels = np.core.defchararray.add(conf_matrix.astype(str), labels)
        sns.heatmap(conf_matrix, fmt='', annot = labels, ax=axes[0], cmap="YlGnBu", xticklabels=[1, 0], yticklabels=[1, 0]);
        axes[0].set(xlabel='Predicted', ylabel='Actual')

        plt.title('Receiver Operating Characteristic')
        sns.lineplot(fpr, tpr, ax=axes[1])                                  # Plot the ROC curve
        plt.plot([0, 1], [0, 1],'--')                                       # Plot the diagonal line
        axes[1].set_xlim([0, 1])                                            # Set x-axis limit to 0 and 1
        axes[1].set_ylim([0, 1])                                            # Set y-axis limit to 0 and 1
        axes[1].set(xlabel = 'False Positive Rate', ylabel = 'True Positive Rate');
        plt.show();
```

In [63]:
```
def evalution(model,X_train, X_test, y_train, y_test ): #for evaluating the model
    print("Training Accuracy :\t ",model.score(X_train, y_train))
    print("Testing Accuracy :\t  ",  model.score(X_test, y_test))
    y_pred = model.predict(X_test)
    r=recall_score( y_test,y_pred)
    print("recall Accuracy :\t  ", r )
    print('Confusion matrix:\n ', confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
    return r
```

Support Vector Machine

In [58]:
```
from sklearn import svm

prams= {'kernel':['linear', 'poly']}
svc=SVC()
clf = GridSearchCV(svc, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

In [84]:
```
prams= {'kernel':['linear'], 'gamma':('scale', 'auto'), 'C': [1,10]}
svc=SVC()
clf = GridSearchCV(svc, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

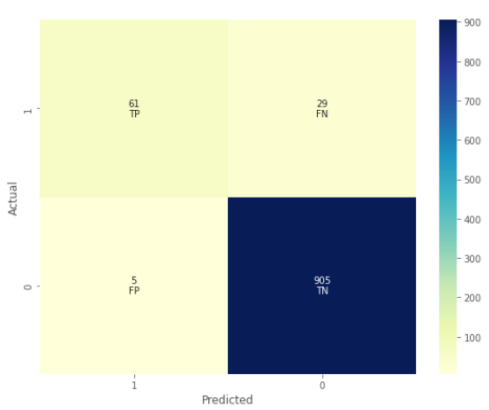
```
{'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
0.658974358974359
```

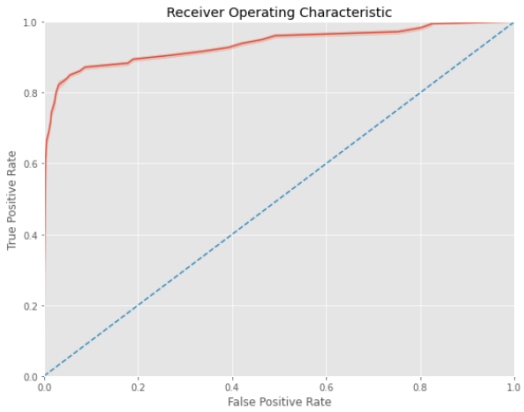
In [59]:
```
str(type(svm).__name__)
```

Out[59]:
```
'module'
```

In [64]:
```
svm= SVC(kernel = 'linear',C= 5, gamma= 'scale', random_state = 0,probability=True)
svm.fit(X_train,y_train)
evalution (svm,X_train, X_test, y_train, y_test )
personal_loan_prediction("SVM", svm, X_train, X_test, y_train, y_test, plot = True)
```

```
Training Accuracy :       0.96275
Testing Accuracy :         0.966
recall Accuracy :          0.6777777777777778
Confusion matrix:
  [[905   5]
 [ 29  61]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       910
           1       0.92      0.68      0.78        90

    accuracy                           0.97      1000
   macro avg       0.95      0.84      0.88      1000
weighted avg       0.96      0.97      0.96      1000

Classification report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       910
           1       0.92      0.68      0.78        90

    accuracy                           0.97      1000
   macro avg       0.95      0.84      0.88      1000
weighted avg       0.96      0.97      0.96      1000

Area under the curve:  0.8361416361416362
```





Decision Tree Classifier

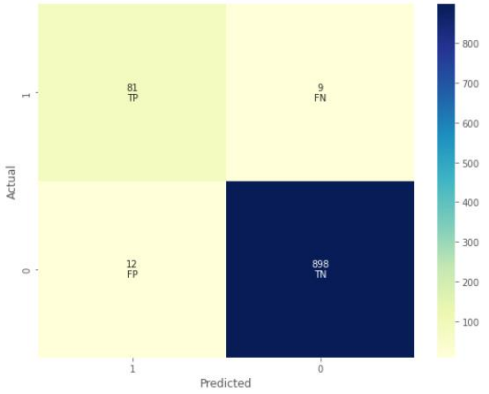
In [23]:
```
from sklearn.model_selection import GridSearchCV

prams= {'criterion':['gini', 'entropy','log_loss'], 'max_depth':[2,4,6,8,10,20],'splitter':['best', 'random']}
dt=DecisionTreeClassifier()
clf = GridSearchCV(dt, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

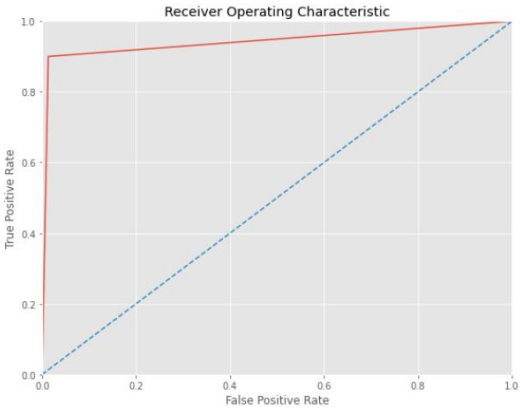
```
{'criterion': 'gini', 'max_depth': 20, 'splitter': 'best'}
0.9102564102564102
```

In [65]:
```
dt  = DecisionTreeClassifier(criterion='gini', max_depth=20, splitter='best', random_state=0)
dt.fit(X_train,y_train)
evalution (dt,X_train, X_test, y_train, y_test )
personal_loan_prediction("Decision Tree", dt, X_train, X_test, y_train, y_test, plot = True)
```

```
Training Accuracy :       1.0
Testing Accuracy :         0.979
recall Accuracy :          0.9
Confusion matrix:
  [[898  12]
 [  9  81]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       910
           1       0.87      0.90      0.89        90
```

```
    accuracy                           0.98      1000
   macro avg       0.93      0.94      0.94      1000
weighted avg       0.98      0.98      0.98      1000

Classification report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       910
           1       0.87      0.90      0.89        90

    accuracy                           0.98      1000
   macro avg       0.93      0.94      0.94      1000
weighted avg       0.98      0.98      0.98      1000

Area under the curve:  0.9434065934065935
```





Random Forest Classifier

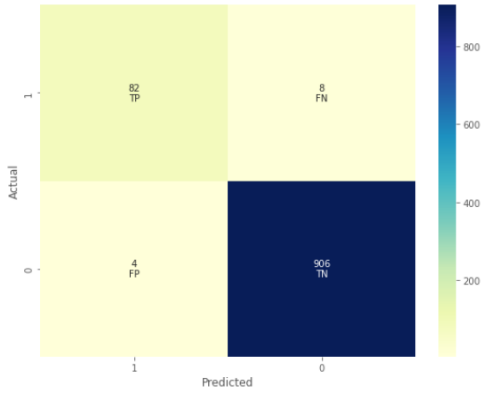
In [44]:
```
from sklearn.model_selection import GridSearchCV

prams= {'criterion':['gini', 'entropy'], 'n_estimators':[10,20,40,50,100,200], 'max_depth':[2,4,6,8,10,20]}
rd=RandomForestClassifier()
clf = GridSearchCV(rd, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

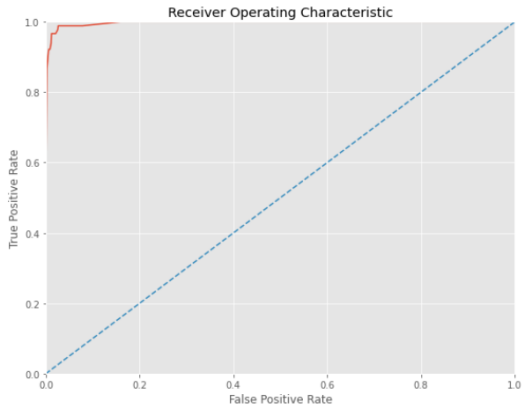
```
{'criterion': 'gini', 'max_depth': 20, 'n_estimators': 100}
0.9051282051282052
```

In [66]:
```
rd  = RandomForestClassifier(n_estimators=40, criterion='entropy', max_depth=20, random_state = 0)
rd.fit(X_train,y_train)
evalution (rd,X_train, X_test, y_train, y_test )
personal_loan_prediction("Random Forest", rd, X_train, X_test, y_train, y_test, plot = True)
```

```
Training Accuracy :       1.0
Testing Accuracy :         0.988
recall Accuracy :          0.9111111111111111
Confusion matrix:
  [[906   4]
 [  8  82]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       910
           1       0.95      0.91      0.93        90

    accuracy                           0.99      1000
   macro avg       0.97      0.95      0.96      1000
weighted avg       0.99      0.99      0.99      1000
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       910
           1       0.95      0.91      0.93        90

    accuracy                           0.99      1000
   macro avg       0.97      0.95      0.96      1000
weighted avg       0.99      0.99      0.99      1000

Area under the curve:  0.9533577533577533
```





Gradient Boosting

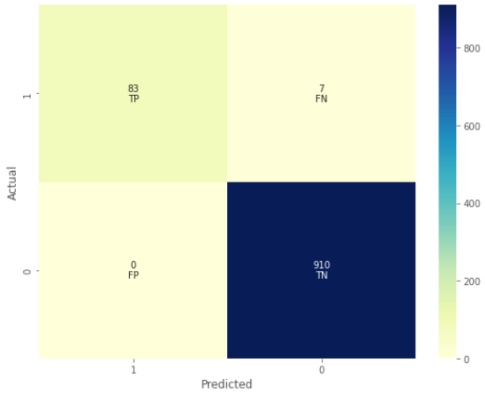
In [46]:
```
prams= {'loss':['log_loss', 'deviance', 'exponential'], 'n_estimators':[100,1000],'criterion':['friedman_mse', 'squared_error',
gr=GradientBoostingClassifier()
clf = GridSearchCV(gr, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

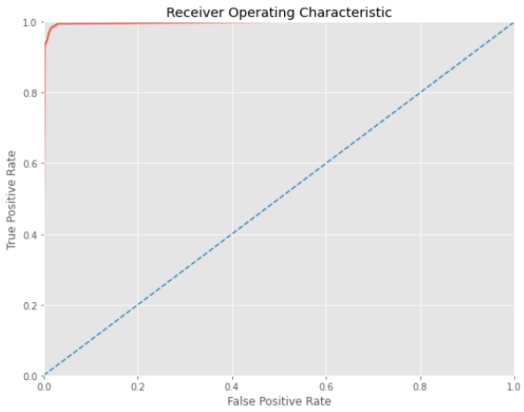
```
{'criterion': 'friedman_mse', 'loss': 'deviance', 'n_estimators': 1000}
0.9051282051282051
```

In [67]:
```
gr  = GradientBoostingClassifier(loss ='deviance' ,n_estimators=1000 ,criterion= 'friedman_mse', random_state=0)
gr.fit(X_train,y_train)
evalution (gr,X_train, X_test, y_train, y_test )
personal_loan_prediction("Gradient Boosting", gr, X_train, X_test, y_train, y_test, plot = True)
```

```
Training Accuracy :       1.0
Testing Accuracy :         0.993
recall Accuracy :          0.9222222222222223
Confusion matrix:
  [[910   0]
 [  7  83]]
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       910
           1       1.00      0.92      0.96        90

    accuracy                           0.99      1000
   macro avg       1.00      0.96      0.98      1000
weighted avg       0.99      0.99      0.99      1000
```

```
Classification report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       910
           1       1.00      0.92      0.96        90

    accuracy                           0.99      1000
   macro avg       1.00      0.96      0.98      1000
weighted avg       0.99      0.99      0.99      1000

Area under the curve:  0.9611111111111111
```





Neural Network

In [71]:
```
prams= {'activation':['identity', 'logistic', 'tanh', 'relu'], 'batch_size':[10,1000],'hidden_layer_sizes':[10,1000],
        'solver':['lbfgs', 'sgd', 'adam'], 'learning_rate':['constant', 'invscaling', 'adaptive']}
mlp=MLPClassifier()
clf = GridSearchCV(mlp, prams,scoring='recall',cv=5)
clf.fit(X_train,y_train)
print(clf.best_params_)
print(clf.best_score_)
```

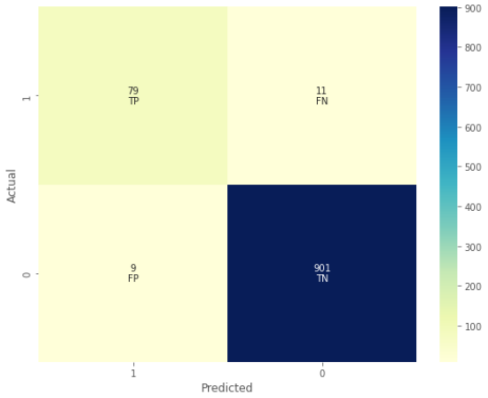
```
{'activation': 'logistic', 'batch_size': 10, 'hidden_layer_sizes': 100, 'learning_rate': 'invscaling', 'solver': 'adam'}
0.8948717948717949
```

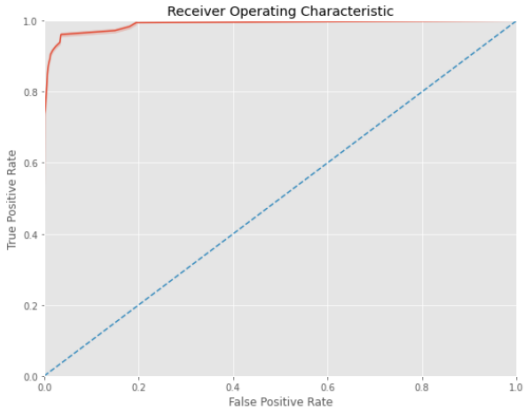
In [68]:
```
mlp=MLPClassifier(activation='logistic',batch_size= 10,solver= 'adam', hidden_layer_sizes= 100,learning_rate= 'invscaling' ,rando
mlp.fit(X_train, y_train)
```

Out[68]:
```
MLPClassifier(activation='logistic', batch_size=10, hidden_layer_sizes=100,
              learning_rate='invscaling', random_state=0)
```

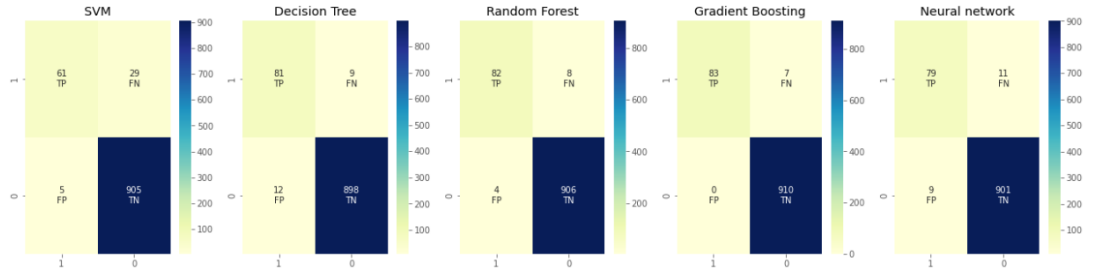
In [69]:
```
evalution (mlp,X_train, X_test, y_train, y_test )
personal_loan_prediction("Neural network", mlp, X_train, X_test, y_train, y_test, plot = True)
```

```
Training Accuracy :       0.9855
Testing Accuracy :         0.98
recall Accuracy :          0.8777777777777778
Confusion matrix:
  [[901   9]
 [ 11  79]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       910
           1       0.90      0.88      0.89        90


    accuracy                           0.98      1000
   macro avg       0.94      0.93      0.94      1000
weighted avg       0.98      0.98      0.98      1000

Classification report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       910
           1       0.90      0.88      0.89        90

    accuracy                           0.98      1000
   macro avg       0.94      0.93      0.94      1000
weighted avg       0.98      0.98      0.98      1000

Area under the curve:  0.933943833943834
```





In [71]:
```
#Plotting the confusion matrix for all our models
import math
fig, axes = plt.subplots(1,5, figsize = (22, 5))

cnt = 0
for c in range(5):
    try:
        conf_matrix = np.flip(list(conf_matrix_all.values())[cnt])
        labels = np.array([['\nTP','\nFN'],['\nFP','\nTN']])
        labels = np.core.defchararray.add(conf_matrix.astype(str), labels)

        sns.heatmap(conf_matrix, fmt='', annot = labels, ax=axes[c], cmap="YlGnBu", xticklabels=[1, 0], yticklabels=[1, 0]);
        axes[c].set(title=list(conf_matrix_all.keys())[cnt])
        cnt += 1
    except:
        pass
```



handling imbalanced in data effect

In [51]:
```
sm = SMOTE(random_state=1)
X_sm_train , y_sm_train = sm.fit_resample(X_train,y_train)
```

Support Vector Machine Modified

In [52]:
```
svm= SVC(kernel = 'linear',C= 5, gamma= 'scale', random_state = 0)
svm.fit(X_sm_train,y_sm_train)
```

Out[52]:
```
SVC(C=5, kernel='linear', random_state=0)
```

In [53]:
```
evalution (svm,X_sm_train, X_test, y_sm_train, y_test )
```

```
Training Accuracy :       0.9216066481994459
Testing Accuracy :         0.924
recall Accuracy :          0.8777777777777778
Confusion matrix:
  [[845  65]
 [ 11  79]]
              precision    recall  f1-score   support

           0       0.99      0.93      0.96       910
           1       0.55      0.88      0.68        90

    accuracy                           0.92      1000
   macro avg       0.77      0.90      0.82      1000
weighted avg       0.95      0.92      0.93      1000
```

Out[53]:
```
0.8777777777777778
```

Decision Tree Classifier

In [54]:
```
tree_class = DecisionTreeClassifier(criterion='gini', max_depth=20, splitter='best', random_state=0)
tree_class.fit(X_sm_train, y_sm_train)
```

Out[54]:
```
DecisionTreeClassifier(max_depth=20, random_state=0)
```

In [55]:
```
evalution (tree_class,X_sm_train, X_test, y_sm_train, y_test )
```

```
Training Accuracy :       1.0
Testing Accuracy :         0.98
recall Accuracy :          0.9222222222222223
Confusion matrix:
  [[897  13]
 [  7  83]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       910
           1       0.86      0.92      0.89        90

    accuracy                           0.98      1000
   macro avg       0.93      0.95      0.94      1000
weighted avg       0.98      0.98      0.98      1000
```

Out[55]:
```
0.9222222222222223
```

Random Forest Classifier

In [56]:
```
Random_class = RandomForestClassifier(n_estimators=40, criterion='entropy', max_depth=20, random_state = 0)
Random_class.fit(X_sm_train, y_sm_train)
```

Out[56]:
```
RandomForestClassifier(criterion='entropy', max_depth=20, n_estimators=40,
                       random_state=0)
```

In [57]:
```
evalution (Random_class,X_sm_train, X_test, y_sm_train, y_test )
```

```
Training Accuracy :       0.9998614958448754
Testing Accuracy :         0.988
recall Accuracy :          0.9111111111111111
Confusion matrix:
  [[906   4]
 [  8  82]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       910
           1       0.95      0.91      0.93        90

    accuracy                           0.99      1000
   macro avg       0.97      0.95      0.96      1000
weighted avg       0.99      0.99      0.99      1000
```

Out[57]:
```
0.9111111111111111
```

Gradient Boosting

In [58]:
```
gr  = GradientBoostingClassifier(loss ='deviance' ,n_estimators=1000 ,criterion= 'friedman_mse', random_state=0)
gr.fit(X_sm_train, y_sm_train)
```

Out[58]:
```
GradientBoostingClassifier(n_estimators=1000, random_state=0)
```

In [62]:
```
evalution (gr,X_sm_train, X_test, y_sm_train, y_test)
```

```
Training Accuracy :       1.0
Testing Accuracy :         0.99
recall Accuracy :          0.9333333333333333
Confusion matrix:
  [[906   4]
 [  6  84]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       910
           1       0.95      0.93      0.94        90

    accuracy                           0.99      1000
   macro avg       0.97      0.96      0.97      1000
weighted avg       0.99      0.99      0.99      1000
```

Out[62]:
```
0.9333333333333333
```

Neural Network

In [77]:
```
mlp=MLPClassifier(activation='logistic',batch_size= 100,solver= 'adam' ,random_state=1, max_iter=300)
mlp.fit(X_sm_train, y_sm_train)
```

Out[77]:
```
MLPClassifier(activation='logistic', batch_size=100, max_iter=300,
              random_state=1)
```

In [79]:
```
mlp=MLPClassifier(activation='logistic',batch_size= 10,solver= 'adam', hidden_layer_sizes= 100,learning_rate= 'invscaling' ,rand
mlp.fit(X_sm_train, y_sm_train)
```

Out[79]:
```
MLPClassifier(random_state=0)
```

In [80]:
```
evalution (mlp,X_sm_train, X_test, y_sm_train, y_test )
```

```
Training Accuracy :       0.9685595567867036
Testing Accuracy :         0.97
recall Accuracy :          0.8555555555555555
Confusion matrix:
  [[893  17]
 [ 13  77]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       910
           1       0.82      0.86      0.84        90

    accuracy                           0.97      1000
   macro avg       0.90      0.92      0.91      1000
weighted avg       0.97      0.97      0.97      1000
```

Out[80]:
```
0.8555555555555555
```