LOAN RISK PREDICTION

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

SAVARAM POKHRAN ROYAL

Regd. No.: 20B91A05R6

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July, 2022 to 6th September, 2022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SAGI RAMA KRISHNAM RAJUENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

Table of Contents

Abstract

The project titled as Loan Risk Flag. In finance, a loan is the lending of money by one or more individuals, organizations, or other entities to other individuals, organizations etc. The recipient (i.e., the borrower) incurs a debt and is usually liable to pay interest on that debt until it is repaid as well as to repay the principal amount borrowed.In this project we check whether one or more individuals, organizations, or other entities to other individuals, organizations etc are efficient to take loans.This means whether the one or more individuals, organizations, or other entities to other individuals, organizations etc will be going to pay their debt or they cause any problems while repaying their debt.

So we check the one or more individuals, organizations, or other entities to other individuals, organizations etc with their personal information with our trained data information by this we can say whether one or more individuals, organizations, or other entities to other individuals, organizations etc An organization wants to predict who possible defaulters are for the consumer loans product. They have data about historic customer behavior based on what they have observed. Hence when they acquire new customers they want to predict who is riskier and who is not.

1.0 Introduction

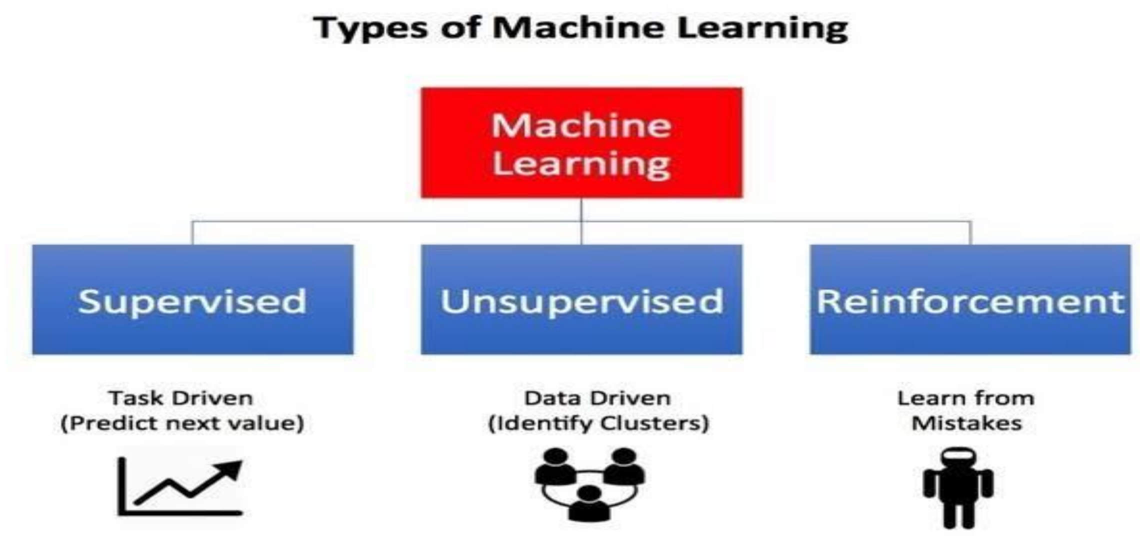
With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at reduced cost. The amount of available data is increasing exponentially and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularitiesor patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies.

A key challenge for the insurance industry is to charge each customer an appropriate price for the risk they represent. Risk varies widely from customer to customer and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this program is to see how well various statistical methods perform in predicting loan Insurance claims based on the characteristics of the driver, vehicle and driver / vehicle coverage details.

A number of factors will determine BI claims prediction among them a driver's age, past accident history, and domicile, etc. However, this contest focused on the relationship between claims and vehicle characteristics well as other characteristics associated with the loan insurance policies.

1.1. What are the different types of Machine Learning?


Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement
Task Driven (Predict next value)
Data Driven (Identify Clusters)
Learn from Mistakes

Machine Learning is defined as the study of computer programs that leverage algorithms and statistical models to learn through inference and patterns without being explicitly programed. Machine Learning field has undergone significant developments in the last decade

As with any method, there are different ways to train machine learning algorithms, each with their own advantages and disadvantages. To understand the pros and cons of each type of machine learning, we must first look at what kind of data they ingest. In ML, there are two kinds of data — labeled data and unlabeled data.

Labeled data has both the input and output parameters in a completely machine-readable pattern, but requires a lot of human labor to label the data, to begin with. Unlabeled data only has one or none of the parameters in a machine-readable form. This negates the need for human labor but requires more complex solutions.

There are also some types of machine learning algorithms that are used in very specific use-cases, but three main methods are used today.

Supervised Learning

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances.

In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem.

The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output. This solution is then deployed for use with the final dataset, which it learns from in the same way as the training dataset. This means that supervised machine learning algorithms will continue to improve even after being deployed, discovering new patterns and relationships as it trains itself on new data.

Unsupervised Learning

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program.

In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.

The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

Reinforcement learning directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial-and-error method. Favorable outputs are encouraged or ‘reinforced’, and non-favorable outputs are discouraged or ‘punished’.

Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not.

In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.

1.2. Benefits of Using Machine Learning in Banking Sector

Benefits of AI and ML in the Banking Industry

Artificial intelligence and Machine Learning in the banking sector will have a deep impact on banking operations and the way banks interact with the customer. The impact will be experienced both by the bank and the customer, with both eventually gaining from it. The banking sector already extensively uses AI and ML to automate many processes and we will see an exponential jump in its usage as we charge into the future. The major areas where we see it being used are as follows:

Fraud Detection

All major banks are embracing artificial intelligence and machine learning as a technique to reduce fraud. Banks are particularly targeted because of money transactions. Credit card frauds are most common. Any bank with a huge client base is susceptible to fraud and hence must keep leveraging the latest technologies to help their clients. They are achieving this with artificial intelligence and deep learning techniques.

Customer Service

Customer service is a core part of banking, and a high customer service rating often influences decisions about where you do your banking. Conversational AI and machine learning are now changing customer experience by accommodating chatbots, real-time feedback, and hyper-personalization customer support is getting reimagined. Virtual assistants such as Alexa, Siri, Google, and so on, upheld by AI, utilize deep behavioral learning to up-sell, cross-sell and decide on the best next steps to retain customers and maximize revenue for the bank

Credit service and loan decisions

By leveraging Machine learning and Artificial Intelligence, banks get better insight into both credit and market risk to be able to reduce loan underwriting risk. Credit and loan decisions are now more often being made by automated underwriting engines which churn through millions of data points and historical data to determine creditworthiness. In addition, by leveraging credit default models using historical data, the ‘Loan Approval’ model is constantly updated to make it better and reduce risk. This is allowing the banks to service sub-prime customers and increase their market share.

Regulatory Compliance

Automated transaction monitoring is a key application of Machine Learning in the banking sector. ML-powered Predictive Analytics platforms have already made an impact and help in monitoring AML transactions and help reduce false positives. KYC (Know-Your-Customer) is another area that has benefited from AI and ML with the usage of facial biometric and ML-based scoring to ensure higher compliance.

1.3. About Industry

A loan is a form of debt incurred by an individual or other entity. The lender—usually a corporation, financial institution, or government—advances a sum of money to the borrower. In return, the borrower agrees to a certain set of terms including any finance charges, interest, repayment date, and other conditions.

1.4. AI / ML Role in Banking sector(loan)

This article will look at some of the newest and most interesting attempts to use AI and machine learning in the lending business. It will not look at every single use of AI in the sector but will give an overview of major applications. Specifically, we'll cover:

- How AI is being used to determine creditworthiness, particularly for those without credit histories

- How AI is being used to streamline the loan process

- How AI is being used to improve customer experience for borrowers

It is also worth nothing all of these goals also feed into each other. For example, the better you can determine an individual's creditworthiness, the more easily you can streamline the internal processes. Similarly, the quicker and less hassle a process is, the more appealing it is to customers.

2.0 Loan Risk Prediction

In this post, we will talk about my most recent project and how to use Machine Learning to Predict the Risk of Loan.One of the most powerful ***Machine Learning Applications*** is Loan Risk Prediction. Probably, all banks are using Machine Learning to decide who can take a loan and who cannot.And actually, build a ***Loan Risk Model*** isn't complex and can be a very intuitive task.The theory is simple: Get historical data of a big variety of people who took a loan and their features, and give to our Model identify the patterns and be able to predict the risk of Loan to a specific person.Let's talk about what you need…

2.1. Main Drivers for AI loan Risk Analysis

Today's BFSI organizations need unparalleled data control, organization, and insight to ensure they're making confident, well-informed decisions. Our suite of no-code, AI-powered data analytics solutions help you streamline the lending and servicing workflow, predict credit and default risk, and give you unmatched data insight so you're making smart, timely decisions. With Altair, you can usher in a whole new world of data insight.

2.2. Internship Project - Data Link

The internship project data has taken from Kaggle and the link is

https://www.kaggle.com/code/mihirpaghdal/default-loan-predictor/data

3.0 AI / ML Modelling and Results

3.1. Your Problem of Statement

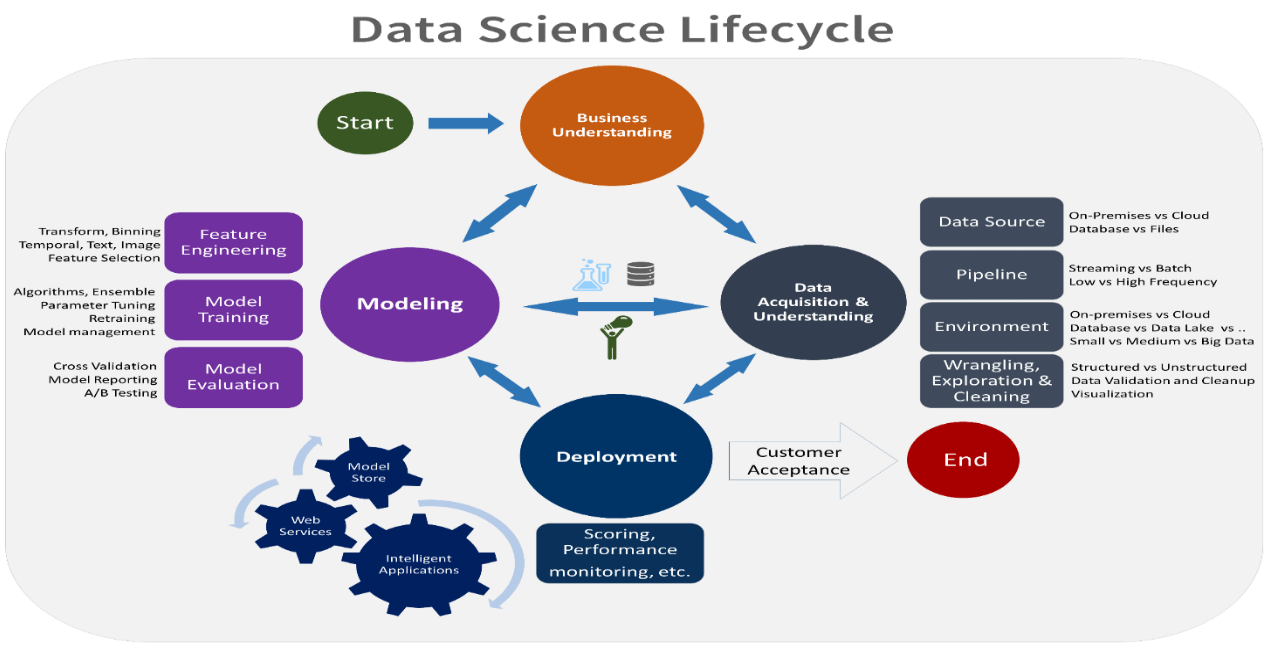
Predictive models are most effective when they are constructed using acompany's own historical claims data since this allows the model to recognize the specific nature of a company's exposure as well as its claims practices.The construction of the model also involves input from the companythroughout the process, as well as consideration of industry leading claimspractices and benchmarks.

Predictive modelling can be used to quantify the impact to the claimsdepartment resulting from the failure to meet or exceed claim service leadingpractices. It can also be used to identify the root cause of claim leakage.Proper use of predictive modelling will allow for potential savings across twodimensions:

- Early identification of claims with the potential for high leakage, thereby allowing for the proactive management of the claim
- Recognition of practices that are unnecessarily increasing claims settlement payments

3.2. Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data


Data Science Lifecycle
Start
Business Understanding
Feature Engineering
Transform, Binning Temporal, Text, Image Feature Selection
Model Training
Algorithms, Ensemble Parameter Tuning Retraining Model management
Model Evaluation
Cross Validation Model Reporting A/B Testing
Modeling
Data Acquisition & Understanding
Data Source
On-Premises vs Cloud Database vs Files
Pipeline
Streaming vs Batch Low vs High Frequency
Environment
On-premises vs Cloud Database vs Data Lake vs .. Small vs Medium vs Big Data
Wrangling, Exploration & Cleaning
Structured vs Unstructured Data Validation and Cleanup Visualization
Deployment
Customer Acceptance
End
Model Store
Web Services
Intelligent Applications
Scoring, Performance monitoring, etc.

3.2.1 Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

The exploratory analysis is done for loan Quote / Policy Conversion with different parameters and all the charts are presented in Appendices 6.2 - List of charts (6.2.1 to 6.2.9)

3.2.2 Data Pre-processing

We removed variables which does not affect our target variable (Claimed_Target) as they may add noise and also increase our computation time, we checked the data for anomalous data points and outliers. we did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1. Check the Duplicate and low variation data

Feature Selection is the process of reducing the number of input variables when developing a predictive model. After an extensive Feature Engineering step, you would end up with a large number of features. You may not use all the features in your model. You would be interested in feeding your model only those significant features or remove the ones that do not have any predictive power. It reduces the computational cost of model training and also improves the performance of the model. This post is for identifying all the duplicate features. These can be of two types:

1. Duplicate Values: When two features have the same set of values

2. Duplicate Index: When the value of two features are different, but they occur at the same index

3. To find whether their exists any duplicates in our data set we use the dataset name Df.duplicated().any()

   If return true there exist duplicate in our dataset otherwise their exist no duplicate in our dataset.

3.2.2.2. Identify and address the missing variables

Missing data (or missing values) is defined as the data value that is not stored for a variable in the observation of interest. The problem of missing data is relatively common in almost all research and can have a significant effect on the conclusions that can be drawn from the data [1]. Accordingly, some studies have focused on handling the missing data, problems caused by missing data, and the methods to avoid or minimize such in medical research.However, until recently, most researchers have drawn conclusions based on the assumption of a complete data set. The general topic of missing data has attracted little attention in the field of anesthesiology.

Missing data present various problems. First, the absence of data reduces statistical power, which refers to the probability that the test will reject the null hypothesis when it is false. Second, the lost data can cause bias in the estimation of parameters. Third, it can reduce the representativeness of the samples. Fourth, it may complicate the analysis of the study. Each of these distortions may threaten the validity of the trials and can lead to invalid conclusions.

3.2.2.3. Handling of Outliers

An outlier is a data point in a data set that is distant from all other observations. A data point that lies outside the overall distribution of dataset (95% of customer behaviour / claims/ spending nature of customer)Many people get confused between Extreme values & Outliers.

What is an Extreme Value?

An Extreme value is just a minimum or a maximum, it need not be much different from of the data.

What is difference between Extreme value & Outlier?

An Extreme value is just a minimum or a maximum, it need not be much different from the data & a point that is far away from the other points called as outlier.

What is the reason for an outlier to exist in dataset?

- Variability in the data

- An Experimental measurement errors

How can we Identify an outlier?

- Using Box plots
- Using Scatter plot
- Using Z score

3.2.2.4. Categorical data and Encoding Techniques

Since we are going to be working on categorical variables in this article, here is a quick refresher on the same with a couple of examples. Categorical variables are usually represented as 'strings' or 'categories' and are finite in number. Here are a few examples:

1. The city where a person lives: Delhi, Mumbai, Ahmedabad, Bangalore, etc.
2. The department a person works in: Finance, Human resources, IT, Production.
3. The highest degree a person has: High school, Diploma, Bachelors, Masters, PhD.
4. The grades of a student: A+, A, B+, B, B- etc.

In the above examples, the variables only have definite possible values. Further, we can see there are two kinds of categorical data-

- **Ordinal Data:** The categories have an inherent order
- **Nominal Data:** The categories do not have an inherent order

In Ordinal data, while encoding, one should retain the information regarding the order in which the category is provided. Like in the above example the highest degree a person possesses, gives vital information about his qualification. The degree is an important feature to decide whether a person is suitable for a post or not.

While encoding Nominal data, we have to consider the presence or absence of a feature. In such a case, no notion of order is present. For example, the city a person lives in. For the data, it is important to retain where a person lives. Here, We do not have any order or sequence. It is equal if a person lives in Delhi or Bangalore.

For encoding categorical data, we have a python package category_encoders. The following code helps you install easily.

```
pip install category_encoders
```

3.2.2.5. Feature Scaling

Here's the curious thing about feature scaling – it improves (significantly) the performance of some machine learning algorithms and does not work at all for others. What could be the reason behind this quirk?

Also, what's the difference between normalization and standardization? These are two of the most commonly used feature scaling techniques in machine learning but a level of ambiguity exists in their understanding. When should you use which technique?

I will answer these questions and more in this article on feature scaling. We will also implement feature scaling in Python to give you a practice understanding of how it works for different machine learning algorithms.

3.2.3 Selection of Dependent and Independent variables

The dependent or target variable here is Claimed Target which tells us a particular policy holder has filed a claim or not the target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4 Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1. Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.

It can be performed using strata function from the library sampling.

3.2.4.2. Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5 Models Used for Development

We built our predictive models by using the following ten algorithms

3.2.5.1. Model 01

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class. The also gives us p-value of the variables which tells us about significance of each independent variable.

3.2.5.2. Model 02

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Breiman and Adele Cutler. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.

- The strength of each tree in the forest. A strong tree is a tree with low error. Byusing trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.3. Model 03

Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. In contrast to Adaboost, the weights of the training instances are not tweaked, instead, each predictor is trained using the residual errors of predecessor as labels. There is a technique called the Gradient Boosted Trees whose base learner is CART (Classification and Regression Trees).

3.2.5.4. Model 04

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

Syntax=From sklearn.tree import DecisionTreeClassifier

classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)

classifier.fit(x_train, y_train)

3.2.5.5. Model 05

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

Syntax=from sklearn.neighbors import KNeighborsClassifier

classifier= KNeighborsClassifier(n_neighbors=5,metric='minkowski', p=2)

classifier.fit(x_train, y_train)

3.2.5.6. Model 06

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

Syntax=from sklearn.svm import SVC # "Support vector classifier"

classifier = SVC(kernel='linear', random_state=0)

classifier.fit(x_train, y_train)

3.2.5.7. Model 07

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset, where N is the size of the original training set. The training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

3.2.5.8 Model 08

Extremely Randomized Trees Classifier(Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a “forest” to output it’s classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, Each tree is provided with a random sample of k features from the feature-set from which each decision tree must select the best feature to split the data based on some mathematical criteria (typically the Gini Index). This random sample of features leads to the creation of multiple de-correlated decision trees.

3.2.5.9. Model 09

LightGBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage.

It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks

3.2.5.10. Model 10

Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data. We have explored the idea behind Gaussian Naive Bayes along with an example. Before going into it, we shall go through a brief overview of Naive Bayes.

3.3. AI / ML Models Analysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given loan dataset of size ~ 252000 policies.

3.3.1 Random Forest Python Code

- from sklearn.ensemble import RandomForestClassifier
- models = RandomForestClassifier()

models = models.fit(x_train,y_train)

```
y_pred = models.predict(x_test)

print('Model Name: ', models)

actual = y_test

predicted = y_pred

matrix = confusion_matrix(actual,predicted)

print('Confusion matrix : \n', matrix)

tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)

print('Outcome values : \n', tp, fn, fp, tn)

C_Report = classification_report(actual,predicted,labels=[1,0])

print('Classification report:\n', classification_report(y_test,y_pred))

from sklearn.metrics import roc_curve, roc_auc_score

print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

from sklearn.metrics import roc_auc_score

from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y_test, y_pred)

fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")
```

```
#plt.savefig('Log_ROC')

plt.show()
```

3.3.2 Extra Trees Python code

- from sklearn.ensemble import ExtraTreesClassifier
- models = ExtraTreesClassifier()

```
models = models.fit(x_train,y_train)

    y_pred = models.predict(x_test)

    print('Model Name: ', models)

    actual = y_test

    predicted = y_pred

    matrix = confusion_matrix(actual,predicted)

    print('Confusion matrix : \n', matrix)

    tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)

    print('Outcome values : \n', tp, fn, fp, tn)

    C_Report = classification_report(actual,predicted,labels=[1,0])

    print('Classification report:\n', classification_report(y_test,y_pred))

    from sklearn.metrics import roc_curve, roc_auc_score

    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

    from sklearn.metrics import roc_auc_score

    from sklearn.metrics import roc_curve

    logit_roc_auc = roc_auc_score(y_test, y_pred)

    fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

    plt.figure()

    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)
```

```
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
#plt.savefig('Log_ROC')
plt.show()
```

3.3.3 Logistic Regression

- from sklearn.linear_model import LogisticRegression
- models = LogisticRegression()

```
models = models.fit(x_train,y_train)
y_pred = models.predict(x_test)
print('Model Name: ', models)
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted)
print('Confusion matrix : \n', matrix)
tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)
print('Outcome values : \n', tp, fn, fp, tn)
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report:\n', classification_report(y_test,y_pred))
from sklearn.metrics import roc_curve, roc_auc_score
```

```
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

from sklearn.metrics import roc_auc_score

from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y_test, y_pred)

fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

#plt.savefig('Log_ROC')

plt.show()
```

3.3.4 Decision Tree Classifier

- from sklearn.tree import DecisionTreeClassifier
- models = DecisionTreeClassifier()

```
models = models.fit(x_train,y_train)

   y_pred = models.predict(x_test)

   print('Model Name: ', models)

   actual = y_test

   predicted = y_pred

   matrix = confusion_matrix(actual,predicted)
```

```
    print('Confusion matrix : \n', matrix)

    tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)

    print('Outcome values : \n', tp, fn, fp, tn)

    C_Report = classification_report(actual,predicted,labels=[1,0])

    print('Classification report:\n', classification_report(y_test,y_pred))

    from sklearn.metrics import roc_curve, roc_auc_score

    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

    from sklearn.metrics import roc_auc_score

    from sklearn.metrics import roc_curve

    logit_roc_auc = roc_auc_score(y_test, y_pred)

    fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

    plt.figure()

    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)

    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

    plt.plot([0, 1], [0, 1],'r--')

    plt.xlim([0.0, 1.0])

    plt.ylim([0.0, 1.05])

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.title('Receiver operating characteristic')

    plt.legend(loc="lower right")

    plt.show()
```

3.3.5 XGBClassifier

- from xgboost import XGBClassifier
- models=XGBClassifier(n_estimators=10,max_depth=3,eval_metric='mloglo ss')

models = models.fit(x_train,y_train)

```
    y_pred = models.predict(x_test)
    print('Model Name: ', models)
    actual = y_test
    predicted = y_pred
    matrix = confusion_matrix(actual,predicted)
    print('Confusion matrix : \n', matrix)
    tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)
    print('Outcome values : \n', tp, fn, fp, tn)
    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report:\n', classification_report(y_test,y_pred))
    from sklearn.metrics import roc_curve, roc_auc_score
    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(y_test, y_pred)
    fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])
    plt.figure()
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
```

3.3.6 KNeighborsClassifier

- from sklearn.neighbors import KNeighborsClassifier
- models = KNeighborsClassifier()

```
models = models.fit(x_train,y_train)
    y_pred = models.predict(x_test)
    print('Model Name: ', models)
    actual = y_test
    predicted = y_pred
    matrix = confusion_matrix(actual,predicted)
    print('Confusion matrix : \n', matrix)
    tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)
    print('Outcome values : \n', tp, fn, fp, tn)
    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report:\n', classification_report(y_test,y_pred))
    from sklearn.metrics import roc_curve, roc_auc_score
    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(y_test, y_pred)
    fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])
    plt.figure()
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
```

3.3.7 GaussianNB

- from sklearn.naive_bayes import GaussianNB
- models = GaussianNB()

```
models = models.fit(x_train,y_train)
    y_pred = models.predict(x_test)
    print('Model Name: ', models)
    actual = y_test
    predicted = y_pred
    matrix = confusion_matrix(actual,predicted)
    print('Confusion matrix : \n', matrix)
    tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)
    print('Outcome values : \n', tp, fn, fp, tn)
    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report:\n', classification_report(y_test,y_pred))
    from sklearn.metrics import roc_curve, roc_auc_score
    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(y_test, y_pred)
    fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])
    plt.figure()
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
```

3.3.8 SVM

- from sklearn.svm import SVC
- models = SVC(C=1.0, kernel='linear', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=True, tol=0.001, cache_size=200, class_weight=None, verbose=False,max_iter=-1,decision_function_shape='ovr',break_ties=False, random_state=None)

```
models = models.fit(x_train,y_train)
y_pred = models.predict(x_test)
print('Model Name: ', models)
actual = y_test
predicted = y_pred
matrix = confusion_matrix(actual,predicted)
print('Confusion matrix : \n', matrix)
tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)
print('Outcome values : \n', tp, fn, fp, tn)
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report:\n', classification_report(y_test,y_pred))
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
```

```
   logit_roc_auc = roc_auc_score(y_test, y_pred)

   fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

   plt.figure()

   # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)

   plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

   plt.plot([0, 1], [0, 1],'r--')

   plt.xlim([0.0, 1.0])

   plt.ylim([0.0, 1.05])

   plt.xlabel('False Positive Rate')

   plt.ylabel('True Positive Rate')

   plt.title('Receiver operating characteristic')

   plt.legend(loc="lower right")

   plt.show()
```

3.3.9 LGBMClassifier

- import lightgbm as lgb
- models =lgb.LGBMClassifier()

```
models = models.fit(x_train,y_train)

   y_pred = models.predict(x_test)

   print('Model Name: ', models)

   actual = y_test

   predicted = y_pred

   matrix = confusion_matrix(actual,predicted)

   print('Confusion matrix : \n', matrix)

   tp, fn, fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)

   print('Outcome values : \n', tp, fn, fp, tn)
```

```
C_Report = classification_report(actual,predicted,labels=[1,0])

print('Classification report:\n', classification_report(y_test,y_pred))

from sklearn.metrics import roc_curve, roc_auc_score

print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

from sklearn.metrics import roc_auc_score

from sklearn.metrics import roc_curve

logit_roc_auc = roc_auc_score(y_test, y_pred)

fpr, tpr, thresholds = roc_curve(y_test,models.predict_proba(x_test)[:,1])

plt.figure()

# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)

plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

plt.plot([0, 1], [0, 1],'r--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver operating characteristic')

plt.legend(loc="lower right")

plt.show()
```

3.3.10 BaggingClassifier

- from sklearn.ensemble import BaggingClassifier
- models = BaggingClassifier()

```
models = models.fit(x_train,y_train)

    y_pred = models.predict(x_test)

    print('Model Name: ', models)
    actual = y_test

    predicted = y_pred

    matrix = confusion_matrix(actual,predicted)

    print('Confusion matrix : \n', matrix)

    tp, fn,
fp,tn=confusion_matrix(actual,predicted,labels=[1,0]).reshape(1)

    print('Outcome values : \n', tp, fn, fp, tn)

    C_Report = classification_report(actual,predicted,labels=[1,0])

    print('Classification report:\n', classification_report(y_test,y_pred))

    from sklearn.metrics import roc_curve, roc_auc_score

    print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))

    from sklearn.metrics import roc_auc_score

    from sklearn.metrics import roc_curve

    logit_roc_auc = roc_auc_score(y_test, y_pred)

    fpr, tpr, thresholds =
roc_curve(y_test,models.predict_proba(x_test)[:,1])

    plt.figure()

    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc)

    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)

    plt.plot([0, 1], [0, 1],'r--')

    plt.xlim([0.0, 1.0])

    plt.ylim([0.0, 1.05])

    plt.xlabel('False Positive Rate')

    plt.ylabel('True Positive Rate')

    plt.title('Receiver operating characteristic')

    plt.legend(loc="lower right")
plt.show()
```

Stratified Sampling:

Random Forest model performance is good, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0). This is because random forest uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with claims dataset.

Simple Random Sampling:

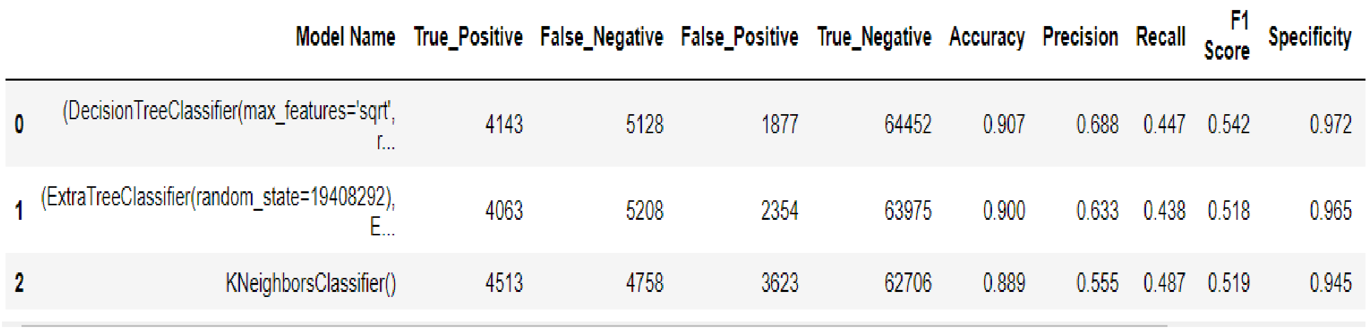
Artificial Neural Networks / Random Forest are out performed Logistic Regression model, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0). This is because Artificial Neural Networks have hidden and complex patterns between different variables and can leads to good predictions with claims dataset.

4.0 Conclusions and Future work

The model results in the following order by considering the model accuracy, F1 score and RoC AUC score.

1) **Random Forest** with Stratified and Random Sampling
2) **Extra Trees Classifier** with Simple Random Sampling
3) **K Neighbors Classifier** with Simple Random Sampling

We recommend model - **Random Forest** with Stratified and Random Sampling technique as a best fit for the given loan prediction dataset. We considered Random Forest because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with claims dataset.



	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity
0	(DecisionTreeClassifier(max_features='sqrt', r...	4143	5128	1877	64452	0.907	0.688	0.447	0.542	0.972
1	(ExtraTreeClassifier(random_state=19408292), E...	4063	5208	2354	63975	0.900	0.633	0.438	0.518	0.965
2	KNeighborsClassifier()	4513	4758	3623	62706	0.889	0.555	0.487	0.519	0.945

Note: Add results screen snapshot here

The future work to evaluate the "Other Types Claims" in loan Insurance by using classification methods.

5.0 References

1)The dataset taken for the project is from Kaggle.

2)The information above was collected from many websites like sklearn,W3schools etc……,

Author's name/Dataset Owner

Nathan Xiang (Commercial Real Estate Analyst at The Corcoran Group New York, New York, United States)

Kaggle account link

https://www.kaggle.com/nathanxiang

Dataset is about

Loan default risk prediction is indispensable to banks, lenders and credit agencies as instead of passively reacting to insolvency losses, it allows these entities to proactively stop loan defaults before they actually happen.

The dataset consists of customer profiling by analyzing the income, asset, occupation, marriage status... of both the normal customers and their in-default counterparts and then use machine learning models to see if we can effectively keep loan defualts at bay.

This makes a major help to the banking sector where the banks can verify the applicant person for loan by their personal information. By this the able to know whether the person is good for the loan or not. If there is no risk we can provide loan to the applicant.

The Information Above Is Taken From The Websites

1)sklearn

https://scikit-learn.org/stable/

2)geeksforgeeks

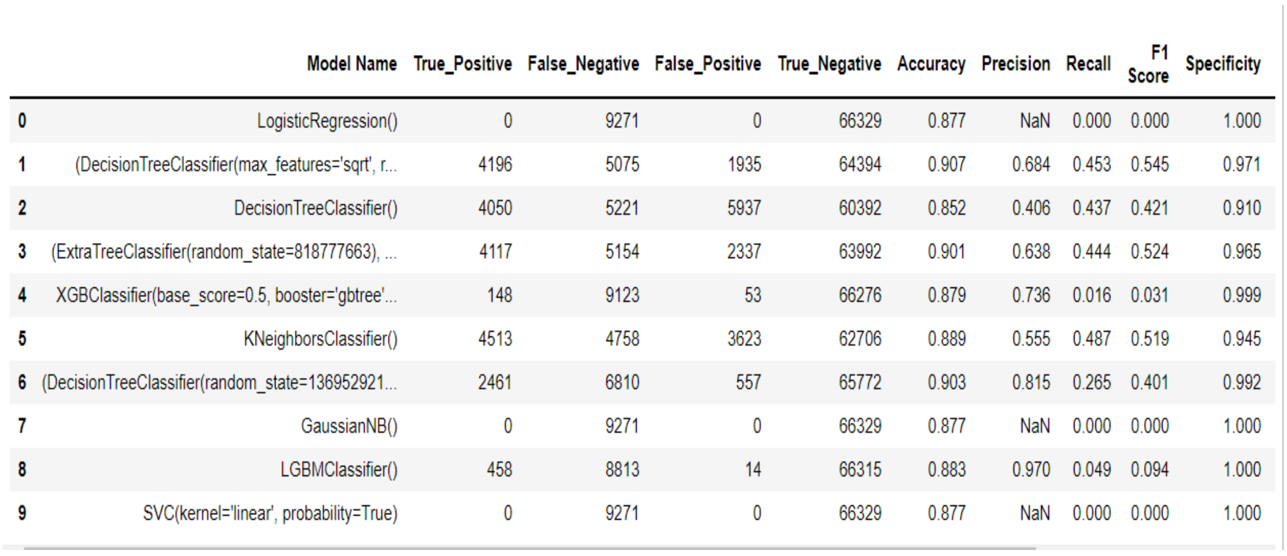
https://www.geeksforgeeks.org/

3)W3schools

https://www.w3schools.in/

6.0 Appendices

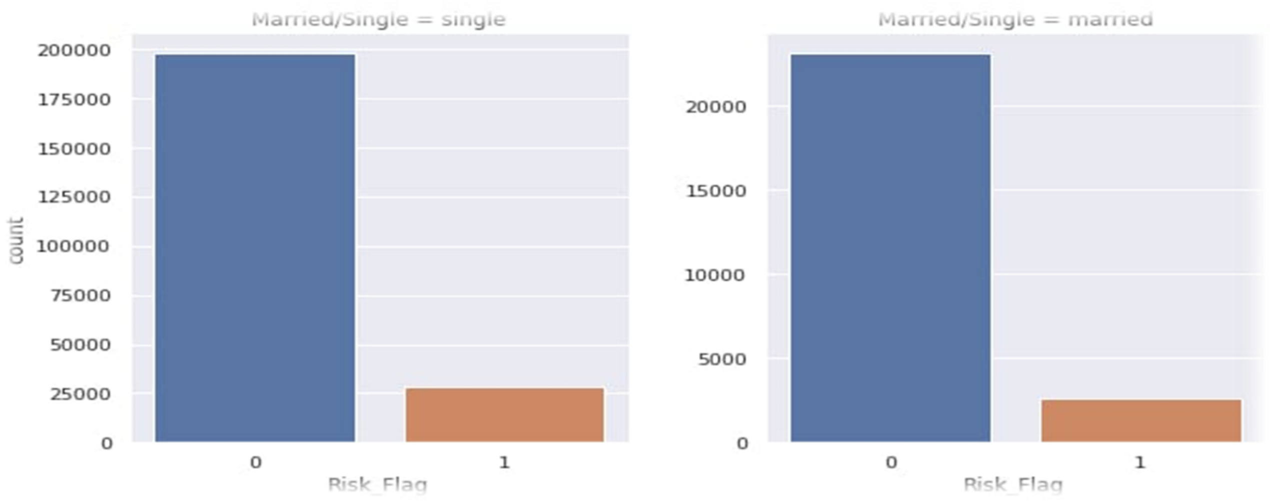
6.1. Python code Results



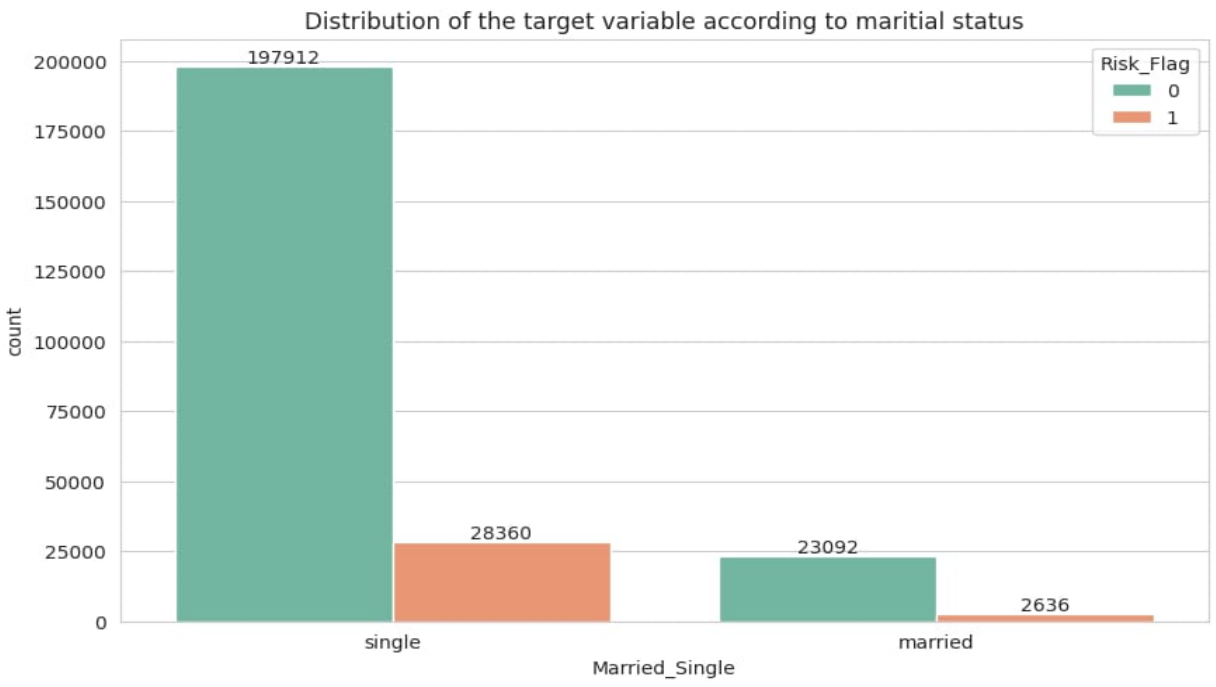
	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity
0	LogisticRegression()	0	9271	0	66329	0.877	NaN	0.000	0.000	1.000
1	(DecisionTreeClassifier(max_features='sqrt', r...	4196	5075	1935	64394	0.907	0.684	0.453	0.545	0.971
2	DecisionTreeClassifier()	4050	5221	5937	60392	0.852	0.406	0.437	0.421	0.910
3	(ExtraTreeClassifier(random_state=818777663), ...	4117	5154	2337	63992	0.901	0.638	0.444	0.524	0.965
4	XGBClassifier(base_score=0.5, booster='gbtree'...	148	9123	53	66276	0.879	0.736	0.016	0.031	0.999
5	KNeighborsClassifier()	4513	4758	3623	62706	0.889	0.555	0.487	0.519	0.945
6	(DecisionTreeClassifier(random_state=136952921...	2461	6810	557	65772	0.903	0.815	0.265	0.401	0.992
7	GaussianNB()	0	9271	0	66329	0.877	NaN	0.000	0.000	1.000
8	LGBMClassifier()	458	8813	14	66315	0.883	0.970	0.049	0.094	1.000
9	SVC(kernel='linear', probability=True)	0	9271	0	66329	0.877	NaN	0.000	0.000	1.000

6.2. List of Charts

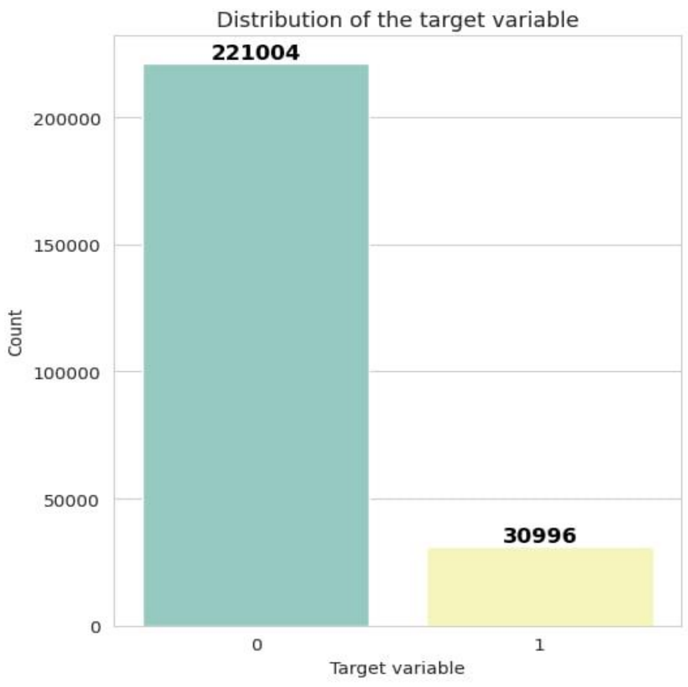
6.2.1 Chart 01: Marriage Status vs Default Risk

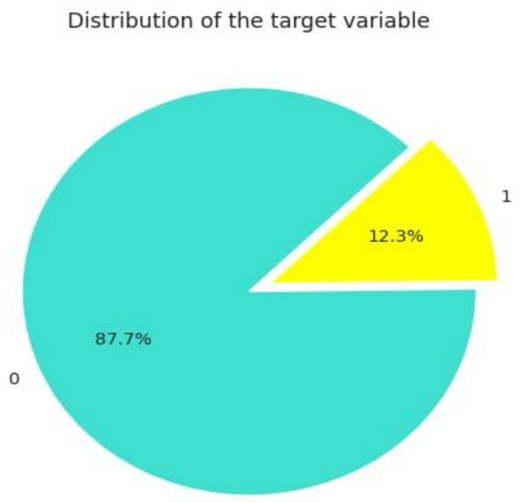


6.2.2 Chart 02: Distribution of the target variable according to maritial status



6.2.3 Chart 03: Distribution of the target variable


Distribution of the target variable
Count
Target variable


Distribution of the target variable