Claims Approval Time prediction Machine Learning Project

Aniruddha Maity (11100116052), GCETTB

Atanu Jana (11100117012) , GCETTB

Bibhas Gayen (11100116045) , GCETTB

Lopamudra Roy (11100116037) , GCETTB

Purnendu Das (11100116031) , GCETTB

❖Acknowledgement

We take this opportunity to express our profound gratitude and deep regards to our faculty **Mr. Titas Roy Chowdhury** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark. We are obliged to our project team members for the valuable information provided by them in their respective fields. We are grateful for our cooperation during the period of this assignment.

Aniruddha Maity

Atanu Jana

Bibhas Gayen

Lopamudra Roy

Purnendu Das

❖Table of Contents

❖Chapter 1: Introduction

▪ 1.1 Introduction

An insurance claim is "a formal request to an insurance company for coverage or compensation for a covered loss or policy event. Once initiated, the claim often goes through a complex process with one of two possible outcomes — the claim is either accepted, leading to a settlement, or rejected. Customer need to submit various number of documents regarding his/her details about insurance before approval of his/her claims. Sometimes what happen is that many customers forget to submit many documents to insurance office and that lead to delay in insurance claims. It's an inconvenient for customer as well as insurance company. This makes customer to visit insurance office several times and makes customer unhappy and agitated toward insurance company's policy. Customer unsatisfaction have great impact on insurance company's business policy.

Our project is all about to overcome such problems. It will predict weather a customer who has applied for insurance claim, has to wait and submit additional documents before granting claims or will granted insurance first.

This project is based on 'Naïve Bayes' a machine learning algorithm to predict the outcomes. It will output 1 if a customer could get claims faster and 0 is if a customer has to wait before granted his/her claims.

If programme output is 0 then insurance office could take a look at customer's submitted documents and inform customer what document need to submit before granting claims.

This will lead to faster payment to the customers claims, reduce human effort, reduce customer inconvenience and increase company reliability upon customer.

1.2 Objective

There are certain benefits of using machine learning approach in claim prediction problem:

- **The algorithms can discover patterns from data that is not used in traditional approaches.**
- **Due to the automated approach, it can be predicted easily weather a customer can get claims faster or it could delay without using man power, effectively, more accurately.**
- **They catch non-linear effects in the data.**
- **This prediction could reduce customer harassment.**
- **Increase work efficiency.**
- **Reduce the pressure on the employee of the insurance company.**
- **Increase the reliability of the company upon customer.**

All of these bullet points translate into improved predictability and greater stability of the claim modelling. Moreover, using artificial intelligence can greatly automate the process, so that specialists can dedicate their time on the results analysis, instead of taking care of sloppy excel calculations. As there is a speedup in the process, the sales representatives can immediately get the expected claim cost based on the past data leading to better sales decisions.

❖Chapter 2: Background Study

▪ 2.1 Brief History of Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "Training Data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.
Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Since the initial standpoint of science, technology and AI, scientists following Blaise Pascal and Von Leibniz ponder about a machine that is intellectually capable as much as humans. Famous writers like Jules Verne, Frank Baum (Wizard of OZ), Marry Shelly (Frankenstein), George Lucas (Star Wars) dreamed artificial beings resembling human behaviors or even more, swamp humanized skills in different contexts. From these books to the real Machine Learning was start about early 1940's during the second world war Alan Mathison Turing the British mathematician, computer scientist, logician, cryptanalyst, philosopher

and theoretical biologist designed and invented a machine to break the German encrypted machine called “Enigma”.
After that, the Machine Learning trend was just started through the very first Machine Learning Algorithm “The Perceptron” by Rosenblatt in 1957. The perceptron is designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in the special, and frequently unknown, conditions which hold for particular biological organisms.
After 1957’s single layer perceptron to multi-layer perceptron, Bayesian methods are introduced for probabilistic inference in machine learning in 1960, then in the year 1967 “The nearest neighbor”algorithm was created, which is the start of basic
pattern recognition. The algorithm was used to map route (Nearest Neighbor), then Kunihiko Fukushima first publishes his work on the “neocognitron”, a type of “artificial neural network” (ANN) “neocognition” later inspires “convolutional neural networks” (CNNs) in 1979. Christopher Watkins develops Q_learning, which greatly improves the practicality and feasibility of reinforcement learning in 1989.Corinna Cortes and Vladimir Vapnik publish their work on support vector machines as well as Tin Kam Ho publishes a paper describing random decision forest in the same year of 1995.
From the early 1940’s to the present 2019 Machine Learning is still developed itself.

- ## 2.2 The Future

  Depending on how well we understand machine learning, some of these insights may sound familiar, others may expose we to new era. Some of the futuristic Machine Learning approaches:

- **Fine-tuned personalization**

Machine learning may be a method of data analysis; however, it's steadily influencing the lives those who own IoT devices like smartwatches, phones, cars, and more. Here's what Ben had to say about the unique relationship between machine learning and consumers.

- **Better search engine experiences**

Search engines are going to improve both the user and the admin experience by leaps and bounds over the next few years. With further development of neural networks and deep learning, search engines of the future will be way better at delivering answers and insights that are highly relevant to the user that is searching.

Right now, we're really good at understanding what results should be served based on the user's profile and the query. However, this process still requires manual configurations and understanding of how search engines work. In the future, results will be tailored even closer to the individual based on their past interactions, preferences and the words they used without any manual administration. We'll also get proactive about alerting people on potential

issues before they even happen and provide actionable recommendations to ensure a smooth operation and excellent search experience.

- **Evolution of data teams**

It's nearly impossible to predict the future of ML and AI. If you've told technology experts 20 years ago what we could do with ML today, they would probably be skeptical, to say the least.

There are, however, certain trends in how ML is being used today and how those cases will evolve in the near-future. ML will be one of the foundational tools for developing and maintaining digital applications in the coming years. This means IT/data teams will spend less time programming and updating applications, but rather have them learn and keep improving their operations continually.

- **Rise of quantum computing**

Quantum computing is going to play a huge part in the future of machine learning. Integration of quantum computing into machine learning will transform the field as we'll see faster processing, accelerated learning and increased capabilities. This means that complex problems that we don't have the ability to solve with current methods could be done so in a small fraction of time. The potential for this is huge and could impact millions of lives for the better – notably in healthcare and medicine.

- **Neural networks running on our mobile devices**

Mobile device may have the ability to conduct machine learning tasks locally, opening up a wide range of opportunities for object recognition, speech, face detection, and other innovations for mobile platforms.

- **Real-time speech translation.**

In late 2014 Skype launched Skype Translator. It's been improving the service since then, and currently provides real-time audio translation among seven languages. If this technology continues to develop, it could significantly improve the quality of international communication or even eradicate language barriers.

❖Chapter 3: Pre-Requirements

■ 3.1 Essential Libraries and Tools

We all know from our childhood the soldiers need proper training with the latest weapons. Then, they can win a war over their opposition party. As the same way, data scientists need an efficient and effective machine learning software, tools or framework whatever we say as a weapon. For developing the system with the required training data to erase the drawbacks and make the machine or device intelligent. Only, a well-defined software can build up a fruitful machine. However, nowadays we develop our machine such a way that we no need to give any instruction about the surroundings. The machine can act by itself, and also it can understand the environment. Therefore, we don't need to guide it. As an instance, a self-driving car. Why is a machine so dynamic at present? It's only for developing the system by utilizing machine learning tools.

Tools are a big part of machine learning and choosing the right tool can be as important as working with the best algorithms.

In this topic, we will take a closer look at machine learning tools used for our project. Discover why they are important and the types of tools that we could choose from.

▪ Why Use Tools

Machine learning tools make applied machine learning faster, easier and more fun.

- **Faster**: Good tools can automate each step in the applied machine learning process. This means that the time from ideas to results is greatly

shortened. The alternative is that you have to implement each capability yourself. From scratch. This can take significantly longer than choosing a tool to use off the shelf.

- **Easier**: You can spend your time choosing the good tools instead of researching and implementing techniques to implement. The alternative is that you have to be an expert in every step of the process in order to implement it. This requires research, deeper exercise in order to understand the techniques, and a higher level of engineering to ensure it is implemented efficiently.
- **Fun**: There is a lower barrier for beginners to get good results. You can use the extra time to get better results or work on more projects. The alternative is that you will spend most of your time building your tools rather than on getting results.

Tools with a Purpose

You do not want to study and use machine learning tools for their own sake. They must serve a strong purpose.

Machine learning tools provide capabilities that you can use to deliver results in a machine learning project. You can use this as a filter when you are trying to decide whether or not to learn a new tool or new feature on your tool. You can ask the question:

"How does this serve me in delivering results in a machine learning project?"

Machine learning tools are not just implementations of machine learning algorithms. They can be, but they can also provide capabilities that you can use at any step in the process of working through a machine learning problem.

- **Good Versus Great Tools**

You want to use the best tools for the problems that you are working on. How to do tell the difference between good and great machine learning tools?

- **Intuitive Interface**: Great machine learning tools provide an intuitive interface onto the sub-tasks of the applied machine learning process. There's a good mapping and suitability in the interface for the task.
- **Best Practice**: Great machine learning tools embody best practices for process, configuration and implementation. Examples include automatic configuration of machine learning algorithms and good process built into the structure of the tool.
- **Trusted Resource**: Great machine learning tools are well maintained, updated frequently and have a community of people around it. Look for activity around a tool as a sign it is being used.

- **When to Use Machine Learning Tools**

  Machine learning tools can save you time and help you consistency deliver good results across projects. Some examples of when you may get the most benefit from using machine learning tools include:

  - **Getting Starting**: When you are just getting started machine learning tools guide you through the process of delivering good results quickly and give you confidence to continue on with your next project.
  - **Day-to-Day**: When you need to get good results to a question quickly machine learning tools can allow you to focus on the specifics of your problem rather than on the depths of the techniques you need to use to get an answer.

- **Project Work**: When you are working on a large project, machine learning tools can help you to prototype a solution, figure out the requirements and give you a template for the system that you may want to implement.

- **Platforms Versus Libraries**

  - There are a lot of machine learning tools. Enough that a google search can leave you feeling overwhelmed.
  - One useful way to think about machine learning tools it so separate them into **Platforms** and **Libraries**. A platform provides all you need to run a project, whereas a library only provides discrete capabilities or parts of what you need to complete a project.
  - This is not a perfect distinction because some machine learning platforms are also libraries or some libraries provide a graphical user interface. Nevertheless, this provides a good point of comparison to differentiate genera case purpose from specific purpose tools.
  -

Machine Learning Platform

- A machine learning platform provides capabilities to complete a machine learning project from beginning to end. Namely, some data analysis, data preparation, modelling and algorithm evaluation and selection.
- Features of machine learning platforms are:
- They provide capabilities required at each step in a machine learning project.

- The interface may be graphical, command line, programming all of these or some combination.
- They provide a lose coupling of features, requiring that you tie the pieces together for your specific project.
- They are tailored for general purpose use and exploration rather than speed, scalability or accuracy.

Essential Libraries & Tools

Programming Tool Used

We have used Python as programming tool for our project.

There are a number of reasons why the python programming language is being used by us.

One of the most commonly cited reasons is the syntax of Python, which has been described as both “elegant” and also “math-like”. Experts point out that the semantics of Python have a particular correspondence to many common mathematical ideas, so that it doesn’t take as much of a learning curve to apply those mathematical ideas in the Python language.

Python is also often described as simple and easy to learn, which is a big part of its appeal for any applied use, including machine learning systems. Some programmers describe Python as having a favourable “complexity/performance trade-off” and describe how using Python is more intuitive than some other languages, because of its accessible syntax.

Other users point out that Python also has particular tools that are extremely helpful in working with machine learning systems. Then there's the libraries. We can use Pandas to wrangle our data, matplotlib to visualize, Keras to build anns, XGBoost to build traditional models, SciKit-Learn for data segmentation… the list goes on and on.

IDE Used

We have used **Jupyter Notebook** as IDE for our project.

Jupyter notebook is a very popular and flexible tool which lets us put our code, output of the code and any kind of visualization or plot etc. in the same document.

It was earlier known as IPython notebook.

The name Jupyter is an acronym which stands for the three languages it was designed for: **JU**lia, **PYT**hon, and **R.**

- **Libraries Used**

One of Python's greatest assets is its extensive set of libraries.

Libraries are sets of routines and functions that are written in a given language. A robust set of libraries can make it easier for developers to perform complex tasks without rewriting many lines of code.

Machine learning is largely based upon mathematics. Specifically, mathematical optimization, statistics and probability. Python libraries help researchers/mathematicians who are less equipped with developer knowledge to easily "do machine learning".

Below are the libraries used in our project:

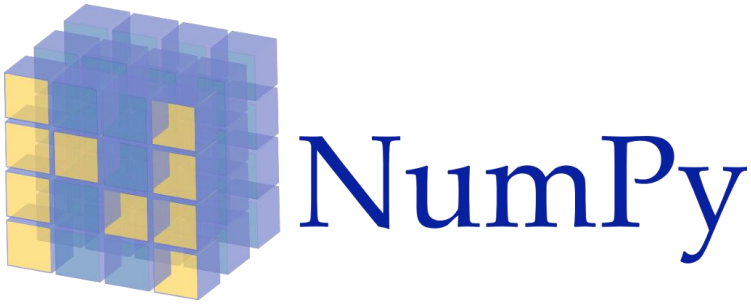
- **Scikit-learn for working with classical ML algorithms**

Scikit-learn is one the most popular ML libraries. It supports many supervised and unsupervised learning algorithms. Examples include linear and logistic regressions, decision trees, clustering, k-means and so on.

It builds on two basic libraries of Python, NumPy and SciPy. It adds a set of algorithms for common machine learning and data mining tasks, including clustering, regression and classification. Even tasks like transforming data, feature selection and ensemble methods can be implemented in a few lines.

- **Data extraction and preparation Libraries**

- **Numpy**



The most fundamental package, around which the scientific computation stack is built, is NumPy (stands for Numerical Python). It provides an abundance of useful features for operations on n-arrays and matrices in Python. The library provides vectorization of mathematical operations on the NumPy array type, which ameliorates performance and accordingly speeds up the execution.

This interface can be utilized for expressing images, sound waves, and other binary raw streams as an array of real numbers in N-dimensional. For implementing this library for machine learning having knowledge of Numpy is important for developers.

- **Pandas**

Pandas is a very popular library that provides high-level data structures which are simple to use as well as intuitive.

It has many inbuilt methods for grouping, combining data and filtering as well as performing time series analysis.

Pandas can easily fetch data from different sources like SQL databases, CSV, Excel, JSON files and manipulate the data to perform operations on it. There are two main structures in the library:

- “Series” — one dimensional

Series	
A	6
B	3
C	4
D	6

- “Data Frames” — two dimensional.

DataFrame				
	A	B	C	D
x0	18	90	32	37
x1	65	33	13	13
x2	62	40	59	71
x3	74	94	83	29
x4	7	30	90	34
x5	67	60	22	5

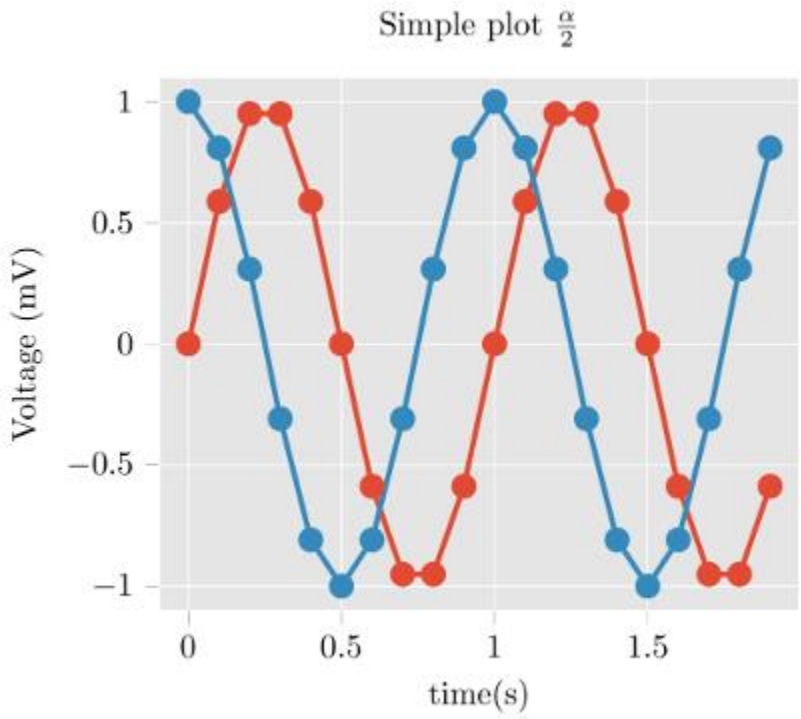
Here is just a small list of things that we can do with Pandas:

- Easily delete and add columns from DataFrame
- Convert data structures to DataFrame objects

- Handle missing data, represents as NaNs
- Powerful grouping by functionality

Data Visualization Library

- **Matplotlib**



The best and most sophisticated ML is meaningless if you can't communicate it to other people.

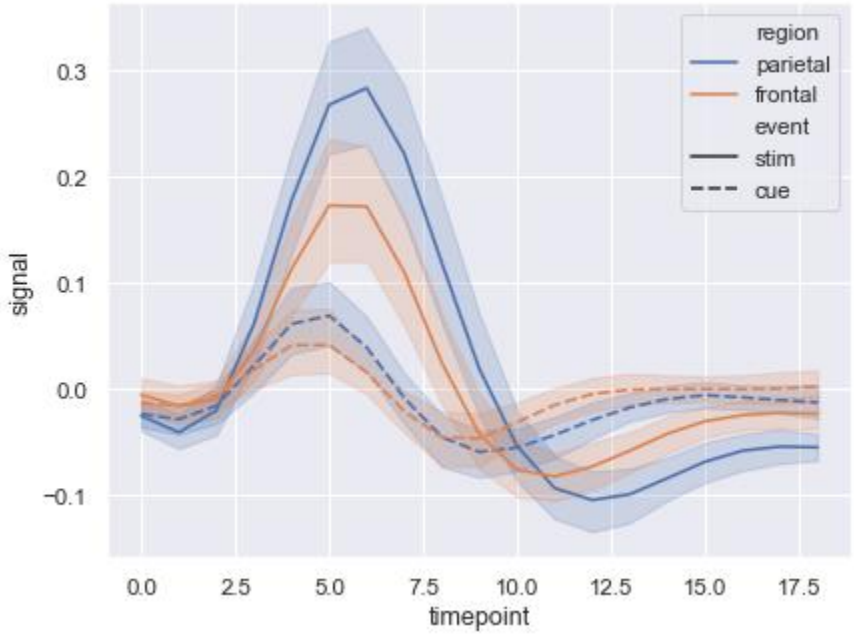
So how do you actually turn around value from all this data that you have? How do you inspire your business analysts and tell them “stories” full of “insights”?

This is where Matplotlib comes to the rescue. It is a standard Python library used by every data scientist for creating 2D plots and graphs. It’s pretty low-level, meaning it requires more commands to generate nice-looking graphs and figures than with some advanced libraries.

However, the flip side of that is flexibility. With enough commands, you can make just about any kind of graph you want with Matplotlib. You can build diverse charts, from histograms and scatterplots to non-Cartesian coordinates graphs.

It supports different GUI backends on all operating systems, and can also export graphics to common vector and graphic formats like PDF, SVG, JPG, PNG, BMP, GIF, etc.

- **Seaborn**



Seaborn is a popular visualization library that builds on Matplotlib's foundations. It is a higher-level library, meaning it's easier to generate certain kinds of plots, including heat maps, time series, box plots and violin plots.

❖Chapter 4: Dataset Preparation

■ 4.1 Dataset Collection

In this project we have a file or data set named **train.csv.** Which size is 116 MB. This CSV file contains the whole data. It contains 114321 rows and 133 columns.

- **Data observation**

At first we import the data via pandas library. We observed……

Shape of the data is = (114321, 133)

columns are : ID,Target,v1,v2,v3,v4,v5……v131

In this project our **y** is the **Target** column of the dataset.

we found many columns contains string values. Those columns are:
v3', 'v22', 'v24', 'v30', 'v31', 'v47', 'v52', 'v56', 'v66','v71', 'v74', 'v75', 'v79', 'v91', 'v107', 'v110', 'v112', 'v113','v125'

And **numerical columns** are :

'v1', 'v2', 'v4', 'v5', 'v6', 'v7', 'v8', 'v9', 'v10', 'v11',
'v12', 'v13', 'v14', 'v15', 'v16', 'v17', 'v18', 'v19', 'v20',
'v21', 'v23', 'v25', 'v26', 'v27', 'v28', 'v29', 'v32', 'v33',
'v34', 'v35', 'v36', 'v37', 'v38', 'v39', 'v40', 'v41', 'v42',
'v43', 'v44', 'v45', 'v46', 'v48', 'v49', 'v50', 'v51', 'v53',
'v54', 'v55', 'v57', 'v58', 'v59', 'v60', 'v61', 'v62', 'v63',
'v64', 'v65', 'v67', 'v68', 'v69', 'v70', 'v72', 'v73', 'v76',
'v77', 'v78', 'v80', 'v81', 'v82', 'v83', 'v84', 'v85', 'v86',

'v87', 'v88', 'v89', 'v90', 'v92', 'v93', 'v94', 'v95', 'v96',
'v97', 'v98', 'v99', 'v100', 'v101', 'v102', 'v103', 'v104',
'v105', 'v106', 'v108', 'v109', 'v111', 'v114', 'v115', 'v116'
'v117', 'v118', 'v119', 'v120', 'v121', 'v122', 'v123',
'v124','v126', 'v127', 'v128', 'v129', 'v130', 'v131'

We found that many columns has **categorical** values. They are :
v3 v22 v24 v31 v38 v47 v52 v56 v62 v66 v71 v72 v74 v75 v79 v91 v107 v110 v112 v125 v129

We have found that many columns have **Nan** values. Reports is:

v1 Total NaN = 49832 Percentage = 43.59 %
v2 Total NaN = 49796 Percentage = 43.56 %
v3 Total NaN = 3457 Percentage = 3.02 %
v4 Total NaN = 49796 Percentage = 43.56 %
v5 Total NaN = 48624 Percentage = 42.53 %
v6 Total NaN = 49832 Percentage = 43.59 %
v7 Total NaN = 49832 Percentage = 43.59 %
v8 Total NaN = 48619 Percentage = 42.53 %
v9 Total NaN = 49851 Percentage = 43.61 %
v10 Total NaN = 84 Percentage = 0.07 %
v11 Total NaN = 49836 Percentage = 43.59 %
v12 Total NaN = 86 Percentage = 0.08 %
v13 Total NaN = 49832 Percentage = 43.59 %
v14 Total NaN = 4 Percentage = 0.00 %
v15 Total NaN = 49836 Percentage = 43.59 %
v16 Total NaN = 49895 Percentage = 43.64 %
v17 Total NaN = 49796 Percentage = 43.56 %
v18 Total NaN = 49832 Percentage = 43.59 %
v19 Total NaN = 49843 Percentage = 43.60 %
v20 Total NaN = 49840 Percentage = 43.60 %

v21 Total NaN = 611 Percentage = 0.53 %

v22 Total NaN = 500 Percentage = 0.44 %
v23 Total NaN = 50675 Percentage = 44.33 %
v24 Total NaN = 0 Percentage = 0.00 %
v25 Total NaN = 48619 Percentage = 42.53 %
v26 Total NaN = 49832 Percentage = 43.59 %
v27 Total NaN = 49832 Percentage = 43.59 %
v28 Total NaN = 49832 Percentage = 43.59 %
v29 Total NaN = 49832 Percentage = 43.59 %
v30 Total NaN = 60110 Percentage = 52.58 %
v31 Total NaN = 3457 Percentage = 3.02 %
v32 Total NaN = 49832 Percentage = 43.59 %
v33 Total NaN = 49832 Percentage = 43.59 %
v34 Total NaN = 111 Percentage = 0.10 %
v35 Total NaN = 49832 Percentage = 43.59 %
v36 Total NaN = 48624 Percentage = 42.53 %
v37 Total NaN = 49843 Percentage = 43.60 %
v38 Total NaN = 0 Percentage = 0.00 %
v39 Total NaN = 49836 Percentage = 43.59 %
v40 Total NaN = 111 Percentage = 0.10 %
v41 Total NaN = 49832 Percentage = 43.59 %
v42 Total NaN = 49832 Percentage = 43.59 %
v43 Total NaN = 49836 Percentage = 43.59 %
v44 Total NaN = 49796 Percentage = 43.56 %
v45 Total NaN = 49832 Percentage = 43.59 %
v46 Total NaN = 48619 Percentage = 42.53 %
v47 Total NaN = 0 Percentage = 0.00 %
v48 Total NaN = 49796 Percentage = 43.56 %
v49 Total NaN = 49832 Percentage = 43.59 %
v50 Total NaN = 86 Percentage = 0.08 %
v51 Total NaN = 50678 Percentage = 44.33 %
v52 Total NaN = 3 Percentage = 0.00 %
v53 Total NaN = 49836 Percentage = 43.59 %
v54 Total NaN = 48619 Percentage = 42.53 %
v55 Total NaN = 49832 Percentage = 43.59 %
v56 Total NaN = 6882 Percentage = 6.02 %
v57 Total NaN = 49832 Percentage = 43.59 %
v58 Total NaN = 49836 Percentage = 43.59 %
v59 Total NaN = 49796 Percentage = 43.56 %

v60 Total NaN = 49832 Percentage = 43.59 %
v61 Total NaN = 49796 Percentage = 43.56 %
v62 Total NaN = 0 Percentage = 0.00 %
v63 Total NaN = 48619 Percentage = 42.53 %
v64 Total NaN = 49796 Percentage = 43.56 %
v65 Total NaN = 49840 Percentage = 43.60 %
v66 Total NaN = 0 Percentage = 0.00 %
v67 Total NaN = 49832 Percentage = 43.59 %
v68 Total NaN = 49836 Percentage = 43.59 %
v69 Total NaN = 49895 Percentage = 43.64 %
v70 Total NaN = 48636 Percentage = 42.54 %
v71 Total NaN = 0 Percentage = 0.00 %
v72 Total NaN = 0 Percentage = 0.00 %
v73 Total NaN = 49836 Percentage = 43.59 %
v74 Total NaN = 0 Percentage = 0.00 %
v75 Total NaN = 0 Percentage = 0.00 %
v76 Total NaN = 49796 Percentage = 43.56 %
v77 Total NaN = 49832 Percentage = 43.59 %
v78 Total NaN = 49895 Percentage = 43.64 %
v79 Total NaN = 0 Percentage = 0.00 %
v80 Total NaN = 49851 Percentage = 43.61 %
v81 Total NaN = 48624 Percentage = 42.53 %
v82 Total NaN = 48624 Percentage = 42.53 %
v83 Total NaN = 49832 Percentage = 43.59 %
v84 Total NaN = 49832 Percentage = 43.59 %
v85 Total NaN = 50682 Percentage = 44.33 %
v86 Total NaN = 49832 Percentage = 43.59 %
v87 Total NaN = 48663 Percentage = 42.57 %
v88 Total NaN = 49832 Percentage = 43.59 %
v89 Total NaN = 48619 Percentage = 42.53 %
v90 Total NaN = 49836 Percentage = 43.59 %
v91 Total NaN = 3 Percentage = 0.00 %
v92 Total NaN = 49843 Percentage = 43.60 %
v93 Total NaN = 49832 Percentage = 43.59 %
v94 Total NaN = 49832 Percentage = 43.59 %
v95 Total NaN = 49843 Percentage = 43.60 %
v96 Total NaN = 49832 Percentage = 43.59 %
v97 Total NaN = 49843 Percentage = 43.60 %
v98 Total NaN = 48654 Percentage = 42.56 %
v99 Total NaN = 49832 Percentage = 43.59 %

v100 Total NaN = 49836 Percentage = 43.59 %
v101 Total NaN = 49796 Percentage = 43.56 %
v102 Total NaN = 51316 Percentage = 44.89 %
v103 Total NaN = 49832 Percentage = 43.59 %
v104 Total NaN = 49832 Percentage = 43.59 %
v105 Total NaN = 48658 Percentage = 42.56 %
v106 Total NaN = 49796 Percentage = 43.56 %
v107 Total NaN = 3 Percentage = 0.00 %
v108 Total NaN = 48624 Percentage = 42.53 %
v109 Total NaN = 48624 Percentage = 42.53 %
v110 Total NaN = 0 Percentage = 0.00 %
v111 Total NaN = 49832 Percentage = 43.59 %
v112 Total NaN = 382 Percentage = 0.33 %
v113 Total NaN = 55304 Percentage = 48.38 %
v114 Total NaN = 30 Percentage = 0.03 %
v115 Total NaN = 49895 Percentage = 43.64 %
v116 Total NaN = 49836 Percentage = 43.59 %
v117 Total NaN = 48624 Percentage = 42.53 %
v118 Total NaN = 49843 Percentage = 43.60 %
v119 Total NaN = 50680 Percentage = 44.33 %
v120 Total NaN = 49836 Percentage = 43.59 %
v121 Total NaN = 49840 Percentage = 43.60 %
v122 Total NaN = 49851 Percentage = 43.61 %
v123 Total NaN = 50678 Percentage = 44.33 %
v124 Total NaN = 48619 Percentage = 42.53 %
v125 Total NaN = 77 Percentage = 0.07 %
v126 Total NaN = 49832 Percentage = 43.59 %
v127 Total NaN = 49832 Percentage = 43.59 %
v128 Total NaN = 48624 Percentage = 42.53 %
v129 Total NaN = 0 Percentage = 0.00 %
v130 Total NaN = 49843 Percentage = 43.60 %
v131 Total NaN = 49895 Percentage = 43.64 %

We store the data into a dataframe name **data**.

4.2 Data Cleaning

We have created another dataset which is **data2**. Where we allow only 2 Nan value in a row. and magically the percentage of Nan in a column also be minimised.

In Data2 dataframe data is:

Total Nan

v1 Total= 0 0.00 %
v2 Total= 0 0.00 %
v3 Total= 192 0.32 %
v4 Total= 0 0.00 %
v5 Total= 0 0.00 %
v6 Total= 0 0.00 %
v7 Total= 0 0.00 %
v8 Total= 0 0.00 %
v9 Total= 0 0.00 %
v10 Total= 0 0.00 %
v11 Total= 0 0.00 %
v12 Total= 0 0.00 %
v13 Total= 0 0.00 %
v14 Total= 0 0.00 %
v15 Total= 0 0.00 %
v16 Total= 0 0.00 %
v17 Total= 0 0.00 %
v18 Total= 0 0.00 %
v19 Total= 0 0.00 %
v20 Total= 0 0.00 %
v21 Total= 108 0.18 %
v22 Total= 101 0.17 %
v23 Total= 0 0.00 %
v24 Total= 0 0.00 %
v25 Total= 0 0.00 %
v26 Total= 0 0.00 %

v27 Total= 0 0.00 %
v28 Total= 0 0.00 %
v29 Total= 0 0.00 %
v30 Total= 21021 34.79 %
v31 Total= 192 0.32 %
v32 Total= 0 0.00 %
v33 Total= 0 0.00 %
v34 Total= 0 0.00 %
v35 Total= 0 0.00 %
v36 Total= 0 0.00 %
v37 Total= 0 0.00 %
v38 Total= 0 0.00 %
v39 Total= 0 0.00 %
v40 Total= 0 0.00 %
v41 Total= 0 0.00 %
v42 Total= 0 0.00 %
v43 Total= 0 0.00 %
v44 Total= 0 0.00 %
v45 Total= 0 0.00 %
v46 Total= 0 0.00 %
v47 Total= 0 0.00 %
v48 Total= 0 0.00 %
v49 Total= 0 0.00 %
v50 Total= 0 0.00 %
v51 Total= 0 0.00 %
v52 Total= 0 0.00 %
v53 Total= 0 0.00 %
v54 Total= 0 0.00 %
v55 Total= 0 0.00 %
v56 Total= 3375 5.59 %
v57 Total= 0 0.00 %
v58 Total= 0 0.00 %
v59 Total= 0 0.00 %
v60 Total= 0 0.00 %
v61 Total= 0 0.00 %
v62 Total= 0 0.00 %
v63 Total= 0 0.00 %
v64 Total= 0 0.00 %
v65 Total= 0 0.00 %
v66 Total= 0 0.00 %

v67 Total= 0 0.00 %
v68 Total= 0 0.00 %
v69 Total= 0 0.00 %
v70 Total= 0 0.00 %
v71 Total= 0 0.00 %
v72 Total= 0 0.00 %
v73 Total= 0 0.00 %
v74 Total= 0 0.00 %
v75 Total= 0 0.00 %
v76 Total= 0 0.00 %
v77 Total= 0 0.00 %
v78 Total= 0 0.00 %
v79 Total= 0 0.00 %
v80 Total= 0 0.00 %
v81 Total= 0 0.00 %
v82 Total= 0 0.00 %
v83 Total= 0 0.00 %
v84 Total= 0 0.00 %
v85 Total= 0 0.00 %
v86 Total= 0 0.00 %
v87 Total= 10 0.02 %
v88 Total= 0 0.00 %
v89 Total= 0 0.00 %
v90 Total= 0 0.00 %
v91 Total= 0 0.00 %
v92 Total= 0 0.00 %
v93 Total= 0 0.00 %
v94 Total= 0 0.00 %
v95 Total= 0 0.00 %
v96 Total= 0 0.00 %
v97 Total= 0 0.00 %
v98 Total= 8 0.01 %
v99 Total= 0 0.00 %
v100 Total= 0 0.00 %
v101 Total= 0 0.00 %
v102 Total= 460 0.76 %
v103 Total= 0 0.00 %
v104 Total= 0 0.00 %
v105 Total= 10 0.02 %
v106 Total= 0 0.00 %

v107 Total= 0 0.00 %
v108 Total= 0 0.00 %
v109 Total= 0 0.00 %
v110 Total= 0 0.00 %
v111 Total= 0 0.00 %
v112 Total= 30 0.05 %
v113 Total= 31339 51.87 %
v114 Total= 0 0.00 %
v115 Total= 0 0.00 %
v116 Total= 0 0.00 %
v117 Total= 0 0.00 %
v118 Total= 0 0.00 %
v119 Total= 0 0.00 %
v120 Total= 0 0.00 %
v121 Total= 0 0.00 %
v122 Total= 0 0.00 %
v123 Total= 0 0.00 %
v124 Total= 0 0.00 %
v125 Total= 0 0.00 %
v126 Total= 0 0.00 %
v127 Total= 0 0.00 %
v128 Total= 0 0.00 %
v129 Total= 0 0.00 %
v130 Total= 0 0.00 %
v131 Total= 0 0.00 %

- In data and data2 dataframe we use to fll up the Nan value with Zero/Median/Mean all three values of the columns.

- We have also Fill the Nan categorical column with most frequent value of the column.

- Then we have label encoded the categorical columns. Because it is necessary to convert string to a numerical value.

- We drop the "v30","v113","v102","v105" columns from data and data2 dataframes due to huge percentage of Nan values.

- We split the continues column's dataset into Xntrain, Xntest, yntrain, yntest

- We perform Variance Threshold method to check columns with Threshold = (.8 * (1 - .8)).
  we found few columns is eliminated in this Threshold. so keep the appropriate columns.

- Then we go for feature selection methods :

Data – Continuous columns :

- ANOVA TEST : we got columns 'v10', 'v114', 'v119', 'v12', 'v123', 'v14', 'v21', 'v34', 'v4', 'v50'

- From mutual info classif We got : 'v10', 'v103', 'v104', 'v114', 'v12', 'v14', 'v20', 'v21', 'v29', 'v50'

- Select From Model in Random Forest Classifier model with n_estimators=10 we got : 'v10', 'v114', 'v12', 'v14', 'v21', 'v34', 'v40', 'v50'

- we finally selected from them : 'v12', 'v50', 'v123', 'v4', 'v104', 'v114', 'v21', 'v103', 'v29', 'v34'

Data – Categorical columns :

- ANOVA TEST : we got columns 'v56', 'v31', 'v72', 'v62', 'v38', 'v129', 'v66', 'v110', 'v47', 'v79'

- From mutual info classif We got : 'v56', 'v22', 'v31', 'v72', 'v62', 'v129', 'v66', 'v110', 'v47', 'v79'

- Select From Model in Random Forest Classifier model with n_estimators=10 we got 'v56', 'v22', 'v112', 'v52', 'v125'

- we finally selected from them 'v22', 'v79', 'v47', 'v110', 'v52', 'v66', 'v112', 'v38', 'v62', 'v129'

- **Data2 – Continuous columns :**

  - ANOVA TEST : we got columns 'v119', 'v21', 'v4', 'v114', 'v50', 'v12', 'v10', 'v14', 'v34', 'v123'

    - From mutual info classif We got : 'v29', 'v64', 'v50', 'v123', 'v13', 'v45', 'v10', 'v14', 'v42', 'v88'

    - Select From Model in Random Forest Classifier model with n_estimators=10 we got 'v40', 'v4', 'v2', 'v12', 'v6', 'v120', 'v10', 'v68', 'v85', 'v70'

    - we finally selected from them 'v50', 'v7', 'v123', 'v2', 'v4', 'v13', 'v64', 'v42', 'v6', 'v120'

- **Data2 – Categorical columns :**

  - ANOVA TEST : we got columns 'v56', 'v31', 'v72', 'v62', 'v38', 'v129', 'v66', 'v110', 'v47', 'v79'

    - From mutual info classif We got : 'v56', 'v22', 'v31', 'v72', 'v62', 'v129', 'v66', 'v110', 'v47', 'v79'

- Select From Model in Random Forest Classifier model with n_estimators=10 we got 'v56', 'v22', 'v112', 'v52', 'v125'

- we finally selected from them 'v22', 'v79', 'v47', 'v110', 'v52', 'v66', 'v112', 'v38', 'v62', 'v129

▪ Final feature selection based on performances :

Data-Continuous columns: "v50","v4","v14","v29"

Data-Categorical columns: 'v66', 'v31', 'v56', 'v72', 'v79', 'v22'

Data2-Continuous columns: 'v50', 'v7', 'v4', 'v13'

Data2-Categorical columns: 'v22', 'v47', 'v52', 'v66','v38', 'v129'

❖Chapter 5: Proposed Methodology

▪ 5.1 Model Development

In this section we will see how our model is developed. We know that we have selected some number of columns out of 131 number of columns form feature selection techniques.

For model developing we have used selected columns. The columns are

Continues value columns : 'v50', 'v7', 'v4', 'v13'

Categorical value columns : 'v22', 'v47', 'v52', 'v66','v38', 'v129'

Previously we have create two type of data sets.

- This data set has considered all Nan in a row and fills with Median/Zero/Mean values separately in continues columns and categorical columns with most frequent values. Named as : **data** And this dataset's shape is = (114321, 133)

- This data set has considered only 2 Nan in a row and fills with Median/Zero/Mean values separately in continues columns and categorical columns with most frequent values. Named as : **data2** And this dataset's shape is = (60423, 133)

So we performs all operations with both data sets. And record all results according to the performance.

After getting the selected continues and categorical columns we need to merge those columns and create a dataset.
So, we created 2 new data sets with those columns **x1_m** from dataset **data**. Here m denotes the Nan continues columns filled with median value.
Then shape of the X1_m is = (114321, 10)
we created another new data set with those columns **x2_m** from dataset **data2**. Here m denotes the Nan continues columns filled with median value.
Then shape of the X2_m is = (60423, 10)

So, we are going to split the dataset into Xtrain, Xtest, ytrain,ytest. After splitting we are ready to testing the performance to model creation.

In this project those variables are in dataset **data** X_all_train, X_all_test, ytrain, ytest. The parameter we passed as test_size=.25 and random_state=11.

X_all_train shape is => (85740, 10)
X_all_test shape is => (28581, 10)
ytrain shape is => (85740, 1)
ytest shape is => (28581, 1)

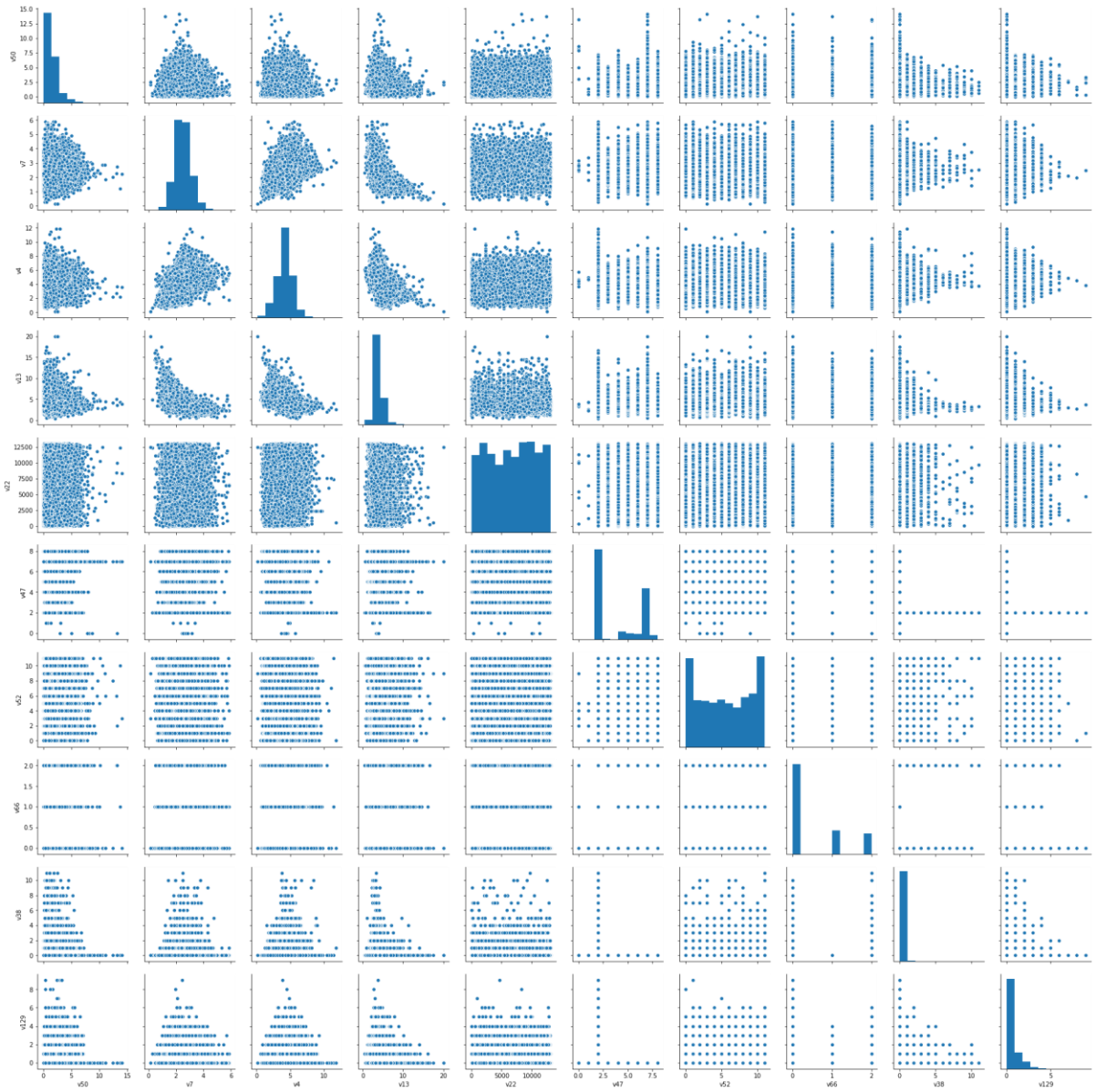
We have also done with dataset **data2**. And the variables are Xn_all_train, Xn_all_test, yntrain, yntest. The parameter we passed as test_size=.25 and random_state=11.

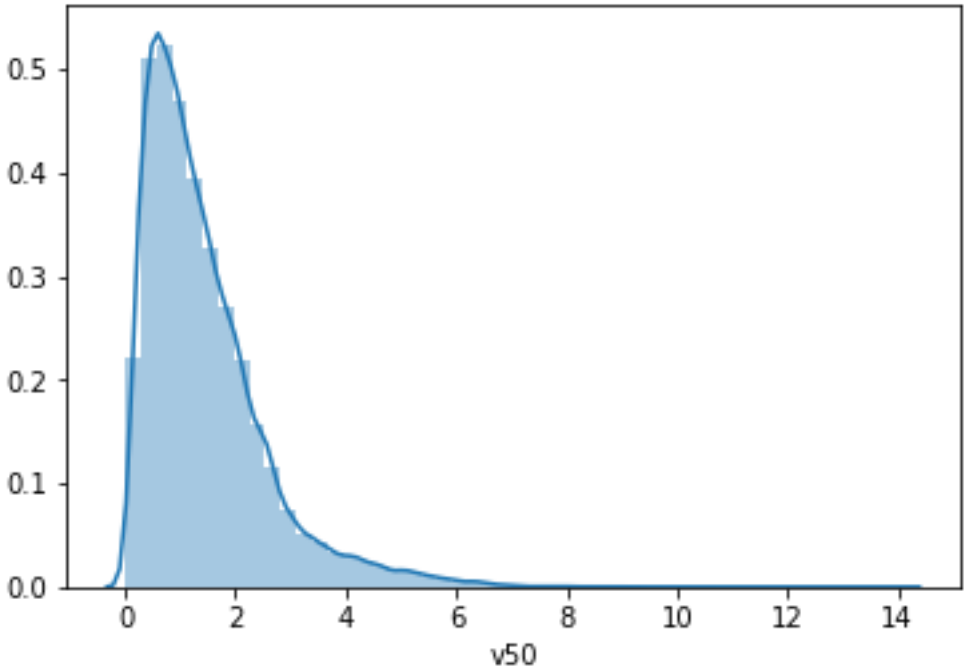
Xn_all_train shape is => (45317, 10)
Xn_all_test shape is => (15106, 10)
yntrain shape is => (45317, 1)
yntest shape is => (15106, 1)

- **5.2 Data Visualization**

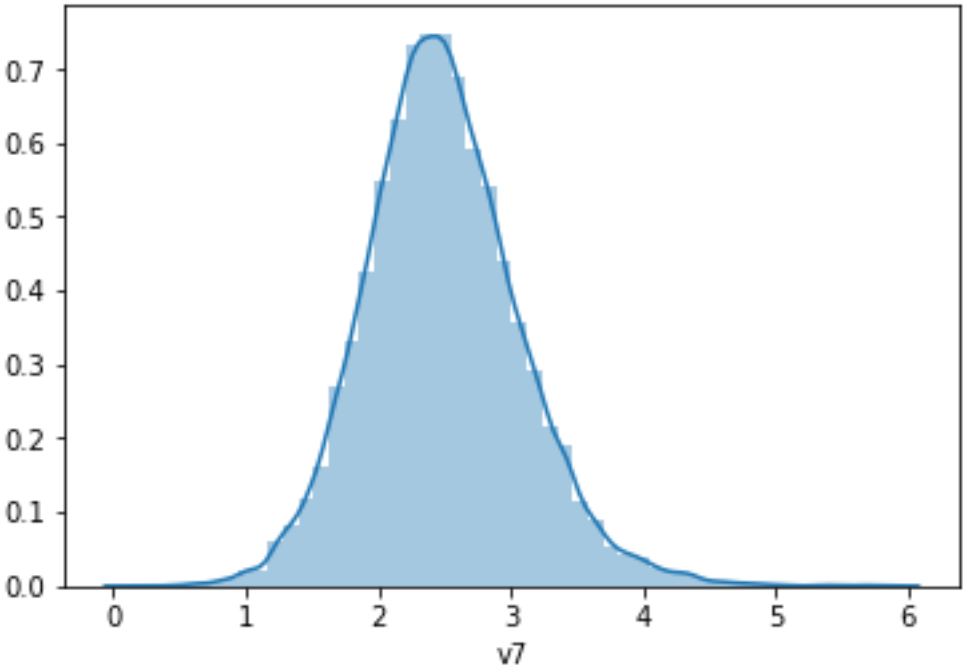
In this section we can visualized our datas.

Pairplot of our data :

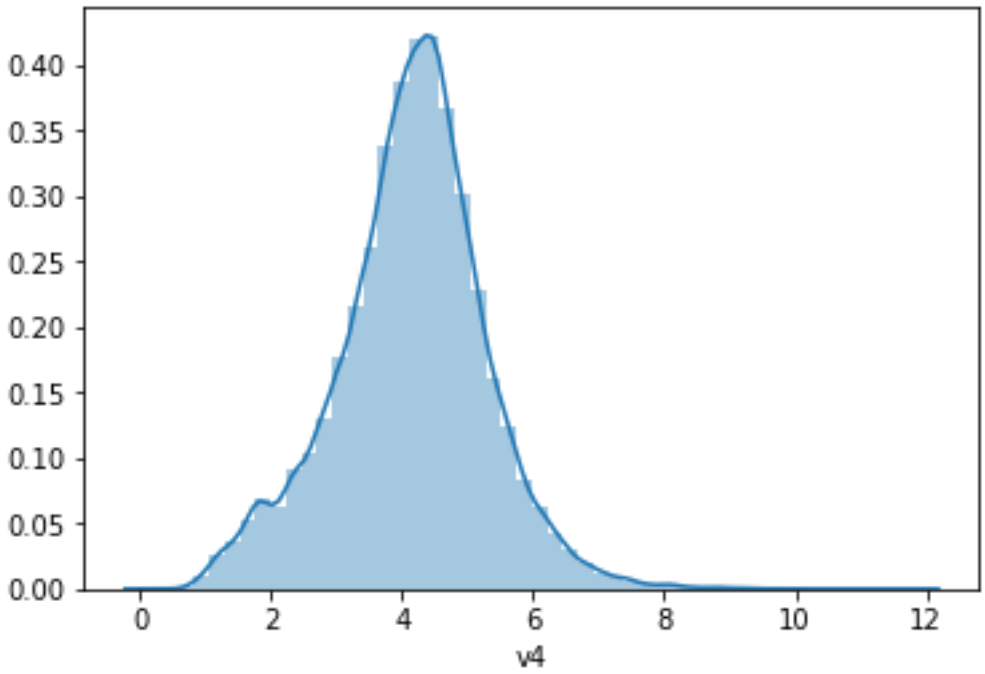




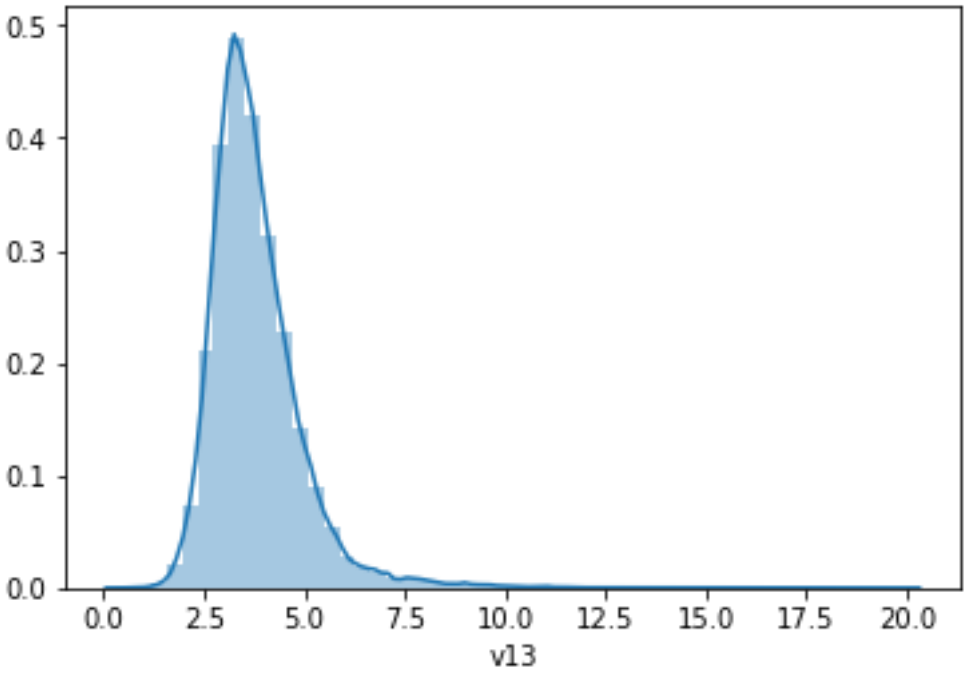
v50 distribution plot



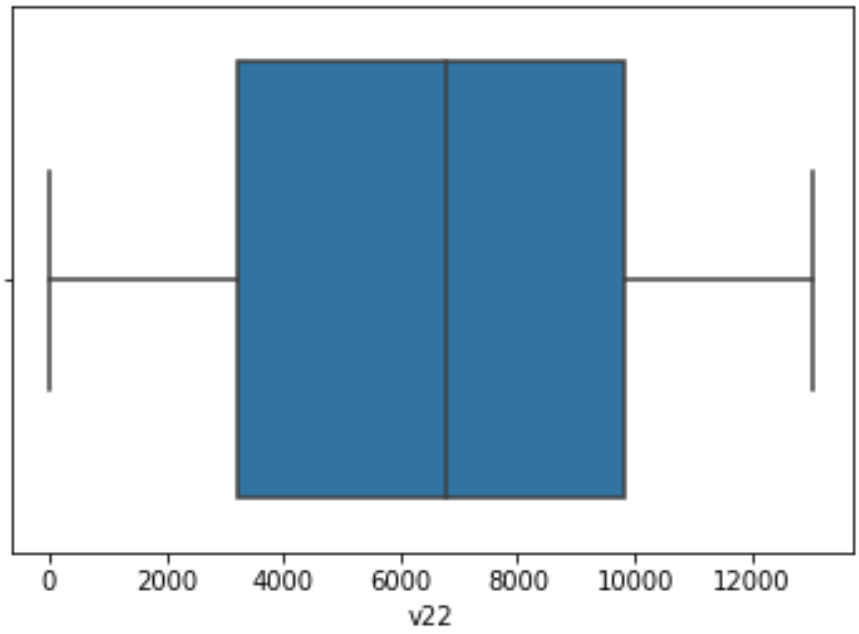
v7 distribution plot



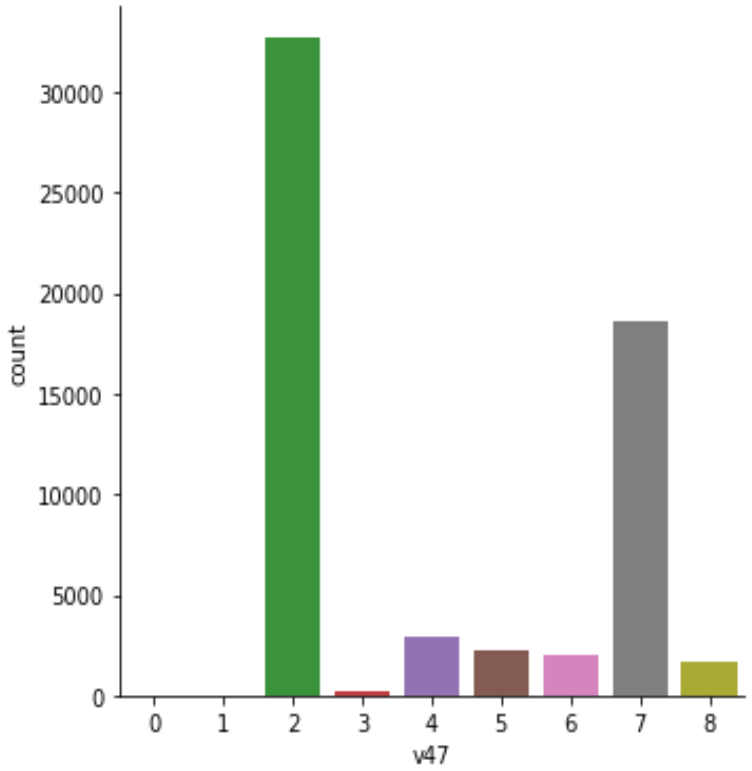
v4 distribution plot



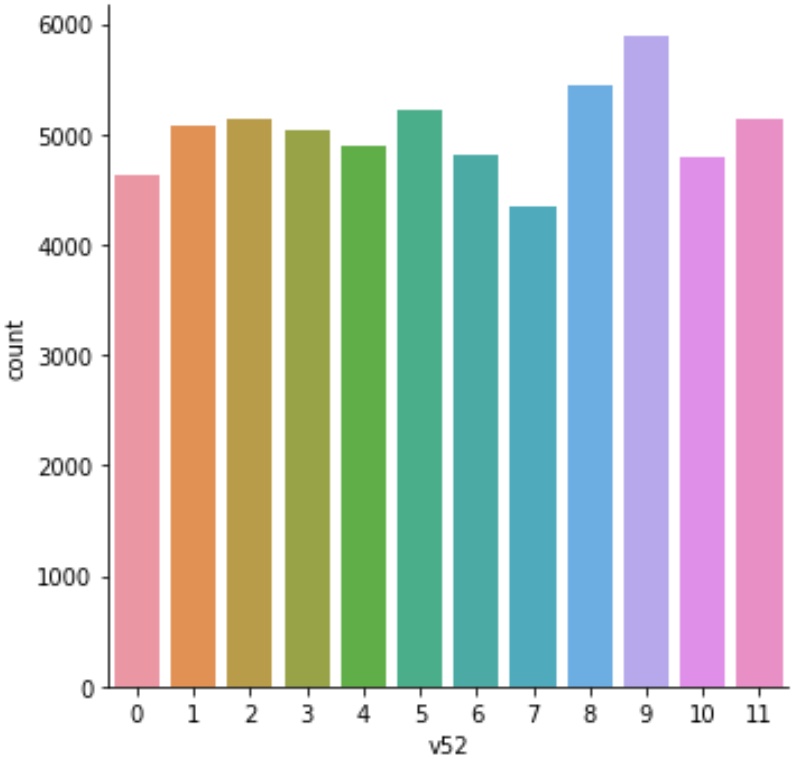
v13 distribution plot



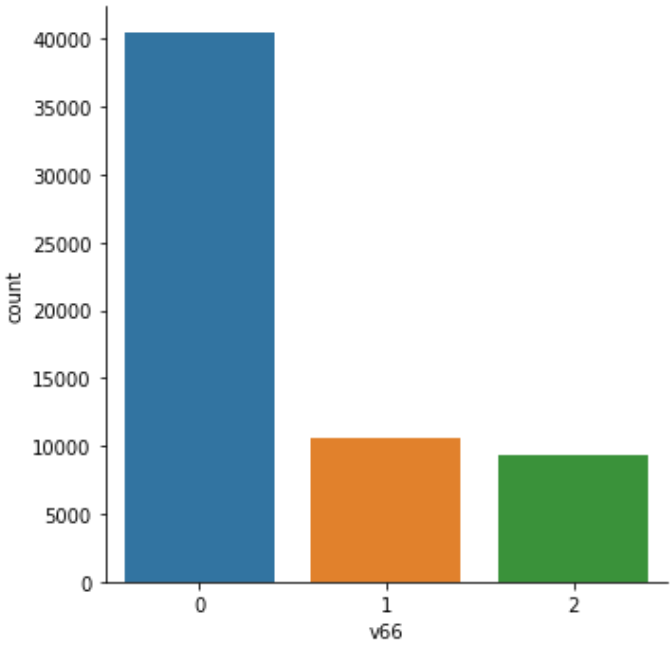
v22 box plot



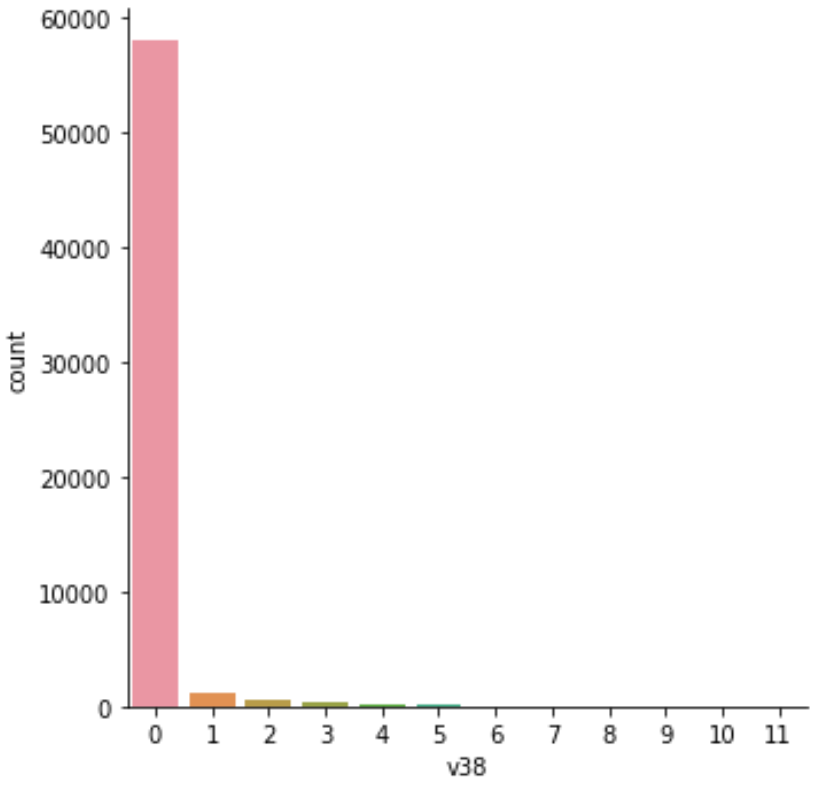
V47 bar plot



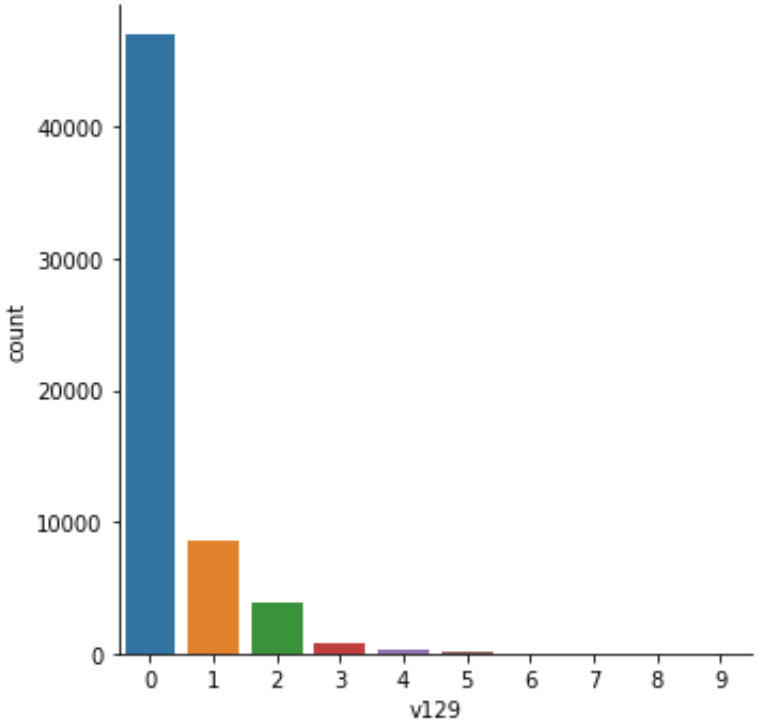
v52 bar plot



v66 bar plot



v38 bar plot



v129 bar plot

▪ 5.3 Algorithm Used

We have built many classifier models to predict the result.

In this project we have used many algorithms. The algorithms list bellow :

1. Logistic Regression
2. Decision Tree Classifier
3. K Nearest Neighbour's Classifier
4. Gaussian NB
5. Logistic Regression
6. Random Forest Classifier

Observation for Dataset-1:

MODELNAME	ACCURACY	PRECISION	RECALL	AUC
LR-train	0.765769	0.769875	0.987719	0.522072
LR-test	0.762814	0.767805	0.986000	0.521474
DecisionTree-train	0.999988	1.000000	0.999985	0.999992
DecisionTree-test	0.685245	0.798526	0.783376	0.579133
KNN(15)-train	0.778272	0.795350	0.954491	0.584787
KNN(15)-test	0.750184	0.778500	0.938107	0.546974
NB-train	0.570655	0.870649	0.512449	0.634565
NB-test	0.570239	0.870871	0.509970	0.635410
Random Forest	0.74521535	0.805098	0.8769514	0.6027635

Observation for Dataset-2:

MODELNAME	ACCURACY	PRECISION	RECALL	AUC
LR-train	0.754882	0.759388	0.985952	0.522206
LR-test	0.755660	0.761125	0.984619	0.520743
DecisionTree-train	1.000000	1.000000	1.000000	1.000000
DecisionTree-test	0.681650	0.794653	0.778520	0.582259
KNN(15)-train	0.757596	0.765375	0.976519	0.537152
KNN(15)-test	0.739971	0.757678	0.962647	0.511501
NB-train	0.583159	0.865540	0.526670	0.640040
NB-test	0.585397	0.871029	0.527685	0.644610
Random Forest	0.73116642	0.79645085	0.86386008	0.5950201

We observed that our model is perform very well in Gaussian Naive Bayes algorithm.

And our data is overfitted on Decision Tree algorithm.

Among the two data set we prefer 2nd one. Because this second data set has more AUC score which is **0.64**. Which is best on our project.

We also want the high precision score in this algorithm we can also get the high precision score as **0.87** and we also want low recall score we can get as **0.52** which is great for our project.

So, our selected Dataset is dataset2, selected algorithm is Gaussian Naive Bayes.

▪ 5.3 Model Building Algorithm

▪ Naive Bayes Classifiers

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

▪ Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ? 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, ..., x_n)$$

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)

y = No

So basically, P(X|y) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

- **Naive assumption**

Now, its time to put a naive assumption to the Bayes' theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

Hence, we reach to the result:

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

which can be expressed as:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

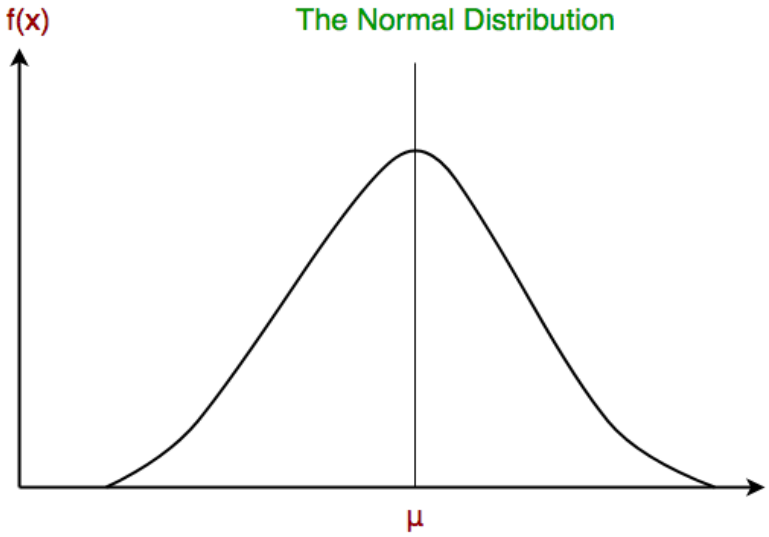
So, finally, we are left with the task of calculating P(y) and $P(x_i \mid y)$.

Please note that P(y) is also called **class probability** and $P(x_i \mid y)$ is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

- **Gaussian Naive Bayes classifier**

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values as shown below:



The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Advantages of this algorithm:

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction

- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (bell curve, which is a strong assumption).

Disadvantages of this algorithm:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency". To solve this, we can use the smoothing technique. One of the simplest smoothing techniques is called Laplace estimation.

- On the other side naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.

- Another limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

Methods of increase the prediction power of a Naive Bayes model

- If continuous features do not have normal distribution, we should use transformation or different methods to convert it in normal distribution.

- If test data set has zero frequency issue, apply smoothing techniques "Laplace Correction" to predict the class of test data set.

- Remove correlated features, as the highly correlated features are voted twice in the model and it can lead to over inflating importance.

- Naive Bayes classifiers has limited options for parameter tuning like alpha=1 for smoothing, fit_prior=[True|False] to learn class prior probabilities or not and some other options (look at detail here). I would recommend to focus on your pre-processing of data and the feature selection.

- You might think to apply some classifier combination technique like ensembling, bagging and boosting but these methods would not help. Actually, "ensembling, boosting, bagging" won't help since their purpose is to reduce variance. Naive Bayes has no variance to minimize.

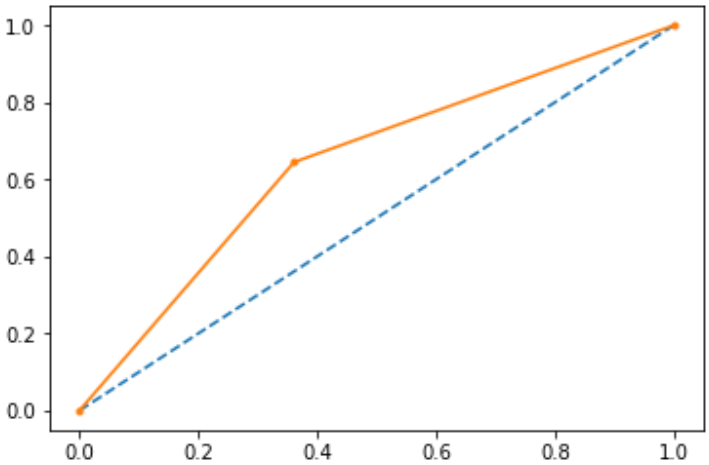
❖Chapter 6: Result Analysis

In this section we will analysis our model result. We select both data sets dataset-1,dataset-2 as data,data2.

We selected Continuous Columns on Data set-1:
"v50","v4","v14","v29"

We selected Categorical Columns on Data set-1:
'v66', 'v31', 'v56', 'v72', 'v79', 'v22'

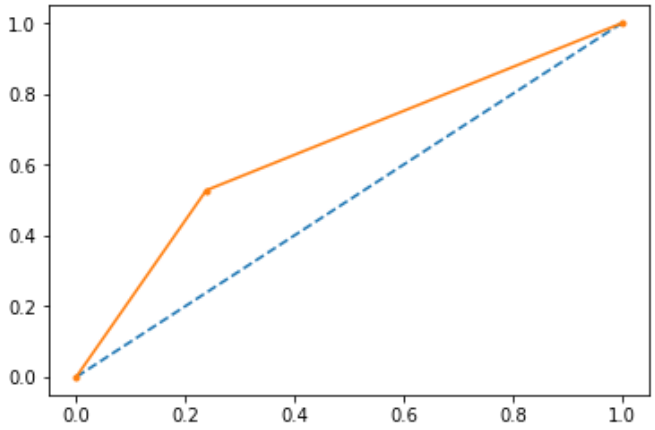
Observation : Naive Bayes Algorithm
Accuracy : 0.6435044260172842
Precision : 0.8495269286754003
Recall : 0.6450379921713102
Logloss : 12.31299032535493
AUC : 0.6418461152234355
Confusion Matrix :
[[4385 2481]
 [7708 14007]]



AUC Curve Graph

We selected Continuous Columns on Data set-2:
'v50', 'v7', 'v4', 'v13'
We selected Categorical Columns on Data set-2:
'v22', 'v47', 'v52', 'v66','v38', 'v129'

Observation : Naive Bayes Algorithm
Accuracy : 0.5853965311796637
Precision : 0.8710285797185551
Recall : 0.5276850061522236
Logloss : 14.319943559078583
AUC : 0.644609670458086 (** Best AUC of our project)
Confusion Matrix :
[[2839 889]
[5374 6004]]



AUC Curve Graph

We selected the Gaussian Naive Bayes Algorithm. And Dataset-2
We got final AUC score which is **0.64**. Which is best on our project.
We also want the high precision score in this project, we can also get the high precision score as **0.87** and we also want low recall score we can get as **0.52** which is great for our objective of this project.

❖ Source Code Of Our Project

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import model_selection
from sklearn import metrics
from sklearn import preprocessing
from sklearn import feature_selection
from sklearn import ensemble
from sklearn.feature_selection import      chi2
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import roc_curve

from sklearn import linear_model
from sklearn import tree
from sklearn import neighbors
from sklearn import naive_bayes

from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB

get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


data = pd.read_csv("train.csv")
```

In[3]:

```
data[data.isna().sum(axis=1)<101].shape
```

In[4]:

```
data[data.isna().sum(axis=1)<100].shape
```

In[5]:

```
data[data.isna().sum(axis=1)<3].shape
```

So We consider only 2 Nan in a Row

In[6]:

```
data2= data[data.isna().sum(axis=1)<3]
```

In[7]:

```
data.shape
```

In[8]:

```
data2.shape
```

In[9]:

```
data.isna().sum()
```

In[10]:

```
X= data.iloc[:,2:]
y= data.iloc[:,1:2]
```

In[11]:

```
X_nr = data2.iloc[:,2:]
y_nr = data2.iloc[:,1:2]
```

In[ ]:

In[12]:

```
X.drop(columns=["v30","v113","v102","v105"],inplace=True)
X_nr.drop(columns=["v30","v113","v102","v105"],inplace=True)
```

Finding Categorical columns

In[13]:

```
cat_list=[]
for i in X.columns:
        a=X[i].value_counts().shape
        if a[0]<50 or X[i].value_counts().iloc[1]>500:
                cat_list.append(i)


# In[14]:


print("Total Categorical Columns =",len(cat_list))
print(*cat_list)


# In[15]:


numeric=["int16","int32","int64","float16","float32","float64"]
numcols=X.select_dtypes(include=numeric).columns.values
strcols=X.select_dtypes(include="object").columns.values

print("Numeric Columns =",len(numcols))
print("String Columns =",len(strcols))


# In[16]:


num_cols = [ i for i in numcols if i not in cat_list]

print("Only Contineus Numeric Columns = ",len(num_cols))


# In[17]:


Xn= X[num_cols]

Xn_med =Xn.iloc[:,:]
```

```
Xn_mn =Xn.iloc[:,:]
Xn_z =Xn.iloc[:,:]

Xn_med.fillna(Xn.median(),inplace=True)
Xn_mn.fillna(Xn.mean(),inplace=True)
Xn_z.fillna(0,inplace=True)


# In[ ]:




# #### Fill with Zero - Median - Mean

# In[ ]:




# In[18]:


Xnr= X_nr[num_cols]

Xnr_med =Xnr.iloc[:,:]
Xnr_mn =Xnr.iloc[:,:]
Xnr_z =Xnr.iloc[:,:]

Xnr_med.fillna(Xnr.median(),inplace=True)
Xnr_mn.fillna(Xnr.mean(),inplace=True)
Xnr_z.fillna(0,inplace=True)


# In[ ]:
```

In[19]:

```
Xntrain,Xntest,yntrain,yntest=model_selection.train_test_split(Xn_med,y,te
st_size=.25,random_state=11)

Xnrtrain,Xnrtest,ynrtrain,ynrtest=model_selection.train_test_split(Xnr_med
,y_nr,test_size=.25,random_state=11)
```

In[ ]:

Variance Threshold of Both Numerical columns

In[ ]:

In[20]:

```
varthobj=feature_selection.VarianceThreshold((.8 * (1 - .8)))

varthobj1=feature_selection.VarianceThreshold((.8 * (1 - .8)))
varthobj2=feature_selection.VarianceThreshold((.8 * (1 - .8)))

varthobj1.fit(Xntrain)
varthobj2.fit(Xnrtrain)

a = varthobj.fit_transform(Xntrain)
```

```
b = varthobj.fit_transform(Xnrtrain)

print(Xntrain.shape,Xnrtrain.shape)
print(a.shape,b.shape)


# In[21]:


Xntrain = Xntrain[Xntrain.columns[varthobj1.get_support(indices=True)]]
Xnrtrain =
Xnrtrain[Xnrtrain.columns[varthobj2.get_support(indices=True)]]


# ### Feature Selection

# #### ANOVA TEST

# In[22]:


skbobj1=feature_selection.SelectKBest(feature_selection.f_classif,k=10)
skbobj1.fit_transform(Xntrain,np.array(yntrain).ravel())
s1=set(Xntrain.columns[skbobj1.get_support()])
s1


# #### mutual_info_classif

# In[23]:


skbobj2=feature_selection.SelectKBest(feature_selection.mutual_info_classi
f,k=10)
skbobj2.fit_transform(Xntrain,np.array(yntrain).ravel())
s2=set(Xntrain.columns[skbobj2.get_support()])
s2
```

Select From Model

In[24]:

```
sfmobj=feature_selection.SelectFromModel(ensemble.RandomForestClassi
fier(n_estimators=10))
sfmobj.fit_transform(Xntrain,np.array(yntrain).ravel())
s4=set(Xntrain.columns[sfmobj.get_support()])
s4
```

In[26]:

```
all_union = ((s1.union(s2))).union(s4)
print(*all_union)
```

In[27]:

```
len(all_union)
```

In[28]:

```
common = ((s1.intersection(s2))).intersection(s4)
```

In[29]:

```
common
```

In[30]:

```
def f_selection(Xtrain,ytrain):

        # ANOVA TEST

skbobj1=feature_selection.SelectKBest(feature_selection.f_classif,k=10)
        skbobj1.fit_transform(Xtrain,np.array(ytrain).ravel())
        s1=set(Xtrain.columns[skbobj1.get_support()])

        # Mutual_info_classif

skbobj2=feature_selection.SelectKBest(feature_selection.mutual_info_classi
f,k=10)
        skbobj2.fit_transform(Xtrain,np.array(ytrain).ravel())
        s2=set(Xtrain.columns[skbobj2.get_support()])


        # Select From Model

sfmobj=feature_selection.SelectFromModel(ensemble.RandomForestClassi
fier(n_estimators=10))
        sfmobj.fit_transform(Xtrain,np.array(ytrain).ravel())
        s3=set(Xtrain.columns[sfmobj.get_support()])

        all_union = ((s1.union(s2)).union(s3))
        common = ((s1.intersection(s2)).intersection(s3))

        return s1,s2,s3,common,all_union


# In[31]:


ns1,ns2,ns3,ncommon,nall_union = f_selection(Xnrtrain,ynrtrain)


# In[32]:
```

```
print(ns1)
print(ns2)
print(ns3)

print(ncommon)
print(nall_union)


# In[33]:


def final_selector(Xtrain,Xtest,ytrain,ytest,cols):
        aucscores={}
        for col in cols:
                model=ensemble.RandomForestClassifier(n_estimators=10)
                model.fit(Xtrain[col].to_frame(),np.array(ytrain).ravel())
                testp=model.predict(Xtest[col].to_frame())
                aucscores[col]=np.round(metrics.roc_auc_score(ytest,testp),4)
        print(aucscores)

        aucser=pd.Series(aucscores)
        selected_cols = list(aucser.sort_values(ascending=False)[:10].index)
# Top 10 Columns

        return selected_cols


# ## Top 10 feature selection of Numerical-1

# In[34]:


final_selector(Xntrain,Xntest,yntrain,yntest,all_union)


# ## Top 10 feature selection of Numerical-2

# In[35]:
```

final_selector(Xnrtrain,Xnrtest,ynrtrain,ynrtest,nall_union)

In[36]:

dual_union = list(set(['v12', 'v50', 'v4', 'v123', 'v101', 'v103', 'v7', 'v49', 'v21', 'v29','v50', 'v4', 'v2', 'v123', 'v64', 'v29', 'v12', 'v120', 'v13', 'v45']))

In[37]:

len(dual_union)

Categorical columns Label encoding

In[38]:

train_cat1=data[cat_list]
train_cat1 = train_cat1.apply(lambda x:x.fillna(x.value_counts().index[0]))

Xcat1=train_cat1.apply(LabelEncoder().fit_transform)
X1_ctrain,X1_ctest,y1train,y1test=model_selection.train_test_split(Xcat1,y,test_size=.25,random_state=11)

In[39]:

print(X1_ctrain.shape)
print(X1_ctest.shape)

print(y1train.shape)
print(y1test.shape)

In[40]:

```
train_cat2=data2[cat_list]
train_cat2 = train_cat2.apply(lambda x:x.fillna(x.value_counts().index[0]))

Xcat2=train_cat2.apply(LabelEncoder().fit_transform)
X2_ctrain,X2_ctest,y2train,y2test=model_selection.train_test_split(Xcat2,y_
nr,test_size=.25,random_state=11)
```

In[41]:

```
print(X2_ctrain.shape)
print(X2_ctest.shape)

print(y2train.shape)
print(y2test.shape)
```

In[42]:

```
nc1=varthobj.fit_transform(X1_ctrain)
print(nc1.shape)
print(X1_ctrain.shape)
```

In[43]:

```
nc2=varthobj.fit_transform(X2_ctrain)
print(nc2.shape)
print(X2_ctrain.shape)
```

Variance Threshold of Both Categorical columns

In[44]:

```
varthobj11=feature_selection.VarianceThreshold((.8 * (1 - .8)))
varthobj22=feature_selection.VarianceThreshold((.8 * (1 - .8)))

varthobj11.fit(X1_ctrain)
varthobj22.fit(X2_ctrain)
```

```
X1_ctrain =
X1_ctrain[X1_ctrain.columns[varthobj11.get_support(indices=True)]]
X2_ctrain =
X2_ctrain[X2_ctrain.columns[varthobj11.get_support(indices=True)]]
```

In[45]:

```
print(X1_ctrain.shape)
print(X2_ctrain.shape)
```

In[ ]:

Selecting Categorical Feature 1

In[46]:

```
ac1,ac2,ac3,acommon,aall_union = f_selection(X1_ctrain,y1train)
```

In[47]:

```
print(ac1)
print(ac2)
print(ac3)

print(acommon)
print(aall_union)


# In[48]:


final_selector(X1_ctrain,X1_ctest,y1train,y1test,aall_union)


# In[ ]:




# ### Selecting Categorical Feature 2

# In[49]:


bc1,bc2,bc3,bcommon,ball_union = f_selection(X2_ctrain,y2train)


# In[50]:


print(bc1)
print(bc2)
print(bc3)

print(bcommon)
print(ball_union)
```

In[51]:

final_selector(X2_ctrain,X2_ctest,y2train,y2test,ball_union)

In[ ]:

Merging Categorical and Numerical Columns

In[ ]:

In[52]:

x1=Xn_med[['v12', 'v50', 'v4', 'v123', 'v101', 'v103', 'v7', 'v49', 'v21', 'v29']]
x1_c=Xcat1[['v66', 'v129', 'v31', 'v56', 'v72', 'v79', 'v112', 'v125', 'v110',]]
y1=y

In[53]:

x1_m=x1.join(x1_c)

In[54]:

```
x1_m.shape


# In[55]:


def reports(ytrue,predicted):
        print("Accuracy :
{}".format(metrics.accuracy_score(ytrue,predicted)))
        print("Precision :
{}".format(metrics.precision_score(ytrue,predicted)))
        print("Recall : {}".format(metrics.recall_score(ytrue,predicted)))
        print("Logloss : {}".format(metrics.log_loss(ytrue,predicted)))
        print("AUC : {}".format(metrics.roc_auc_score(ytrue,predicted)))
        print("Confusion Matrix :
\n{}".format(metrics.confusion_matrix(ytrue,predicted)))


# In[56]:


def modelstats1(Xtrain,Xtest,ytrain,ytest):
        stats=[]
        modelnames=["LR","DecisionTree","KNN","NB"]
        models=list()
        models.append(linear_model.LogisticRegression())
        models.append(tree.DecisionTreeClassifier())
        models.append(neighbors.KNeighborsClassifier())
        models.append(naive_bayes.GaussianNB())
        for name,model in zip(modelnames,models):
                if name=="KNN":
                        k=[l for l in range(5,17,2)]
                        grid={"n_neighbors":k}
                        grid_obj =
model_selection.GridSearchCV(estimator=model,param_grid=grid,scoring=
"f1")
                        grid_fit =grid_obj.fit(Xtrain,ytrain)
                        model = grid_fit.best_estimator_
                        model.fit(Xtrain,ytrain)
```

```
name=name+"("+str(grid_fit.best_params_["n_neighbors"])+")"
                print(grid_fit.best_params_)
            else:
                model.fit(Xtrain,ytrain)
            trainprediction=model.predict(Xtrain)
            testprediction=model.predict(Xtest)
            scores=list()
            scores.append(name+"-train")
            scores.append(metrics.accuracy_score(ytrain,trainprediction))
            scores.append(metrics.precision_score(ytrain,trainprediction))
            scores.append(metrics.recall_score(ytrain,trainprediction))
            scores.append(metrics.roc_auc_score(ytrain,trainprediction))
            stats.append(scores)
            scores=list()
            scores.append(name+"-test")
            scores.append(metrics.accuracy_score(ytest,testprediction))
            scores.append(metrics.precision_score(ytest,testprediction))
            scores.append(metrics.recall_score(ytest,testprediction))
            scores.append(metrics.roc_auc_score(ytest,testprediction))
            stats.append(scores)


colnames=["MODELNAME","ACCURACY","PRECISION","RECALL","AUC"]
    return pd.DataFrame(stats,columns=colnames)


# In[57]:


X_all_train,X_all_test,ytrain,ytest=model_selection.train_test_split(x1_m,y1,t
est_size=.25,random_state=11)


# In[58]:


model=ensemble.RandomForestClassifier(n_estimators=10)
model.fit(X_all_train,np.array(ytrain).ravel())
```

```
testp=model.predict(X_all_test)
print("For Random Forest\n\n")
reports(ytest,testp)


# In[59]:


nv = GaussianNB()
nv.fit(X_all_train,np.array(ytrain).ravel())
testp = nv.predict(X_all_test)
print("For Naive Bayes\n\n")
reports(ytest,testp)


# In[60]:


modelstats1(X_all_train,X_all_test,ytrain,ytest)


# In[61]:


def naive_randomf(x,y):

X_all_train,X_all_test,ytrain,ytest=model_selection.train_test_split(x,y,test_si
ze=.25,random_state=11)

        model=ensemble.RandomForestClassifier(n_estimators=10)
        model.fit(X_all_train,np.array(ytrain).ravel())
        testp=model.predict(X_all_test)
        print("\nFor Random Forest\n")
        reports(ytest,testp)


        fpr, tpr, thresholds = roc_curve(ytest,testp)
        plt.plot([0, 1], [0, 1], linestyle='--')
        plt.plot(fpr, tpr, marker='.')
```

```
        plt.show()


        print("-----------------------------------")

        nv = GaussianNB()
        nv.fit(X_all_train,np.array(ytrain).ravel())
        testp = nv.predict(X_all_test)
        print("\nFor Naive Bayes\n")
        reports(ytest,testp)




        fpr, tpr, thresholds = roc_curve(ytest,testp)
        plt.plot([0, 1], [0, 1], linestyle='--')
        plt.plot(fpr, tpr, marker='.')
        plt.show()


# ## All NAn fill with Zero/Median/Mean Data

# In[200]:


x1=Xn_mn[["v50","v4","v14","v29"]]
x1_c=Xcat1[['v66', 'v31', 'v56', 'v72', 'v79', 'v22']]
y1=y
x1_m=x1.join(x1_c)


# In[201]:


naive_randomf(x1_m,y1)


# ### Nan Removed Row Data
```

In[167]:

```
x2=Xnr_z[['v50', 'v7', 'v4', 'v13']]
x2_c=Xcat2[['v22', 'v47', 'v52', 'v66','v38', 'v129']]
y2=y_nr

x2_m=x2.join(x2_c)
```

In[168]:

```
naive_randomf(x2_m,y2)
```

In[202]:

```
Xn_all_train,Xn_all_test,yntrain,yntest=model_selection.train_test_split(x2_
m,y2,test_size=.25,random_state=11)
```

In[203]:

```
modelstats1(Xn_all_train,Xn_all_test,yntrain,yntest)
```

❖ Chapter 7: Conclusion & Future Scope

▪ CONCLUSION:

Insurance is a large investment and you will most likely purchase multiple policies throughout your lifetime. In insurance you transfer your risk (negative deviation from an future outcome) to an insurance company which will pay you at the time when the event insured happens with respect to the contract signed by both the parties .So you do not want to waste your money on policies that do not meet your needs, but the right insurance policy can protect you and your family from unforeseen disasters.

Machine learning can play a big role where the machine itself understands and analyses for future candidates and does it's work efficiently and accurately. In this study after testing all the models i.e. random forest classifier, k-nn, Nave Bayes and decision tree, we found out Naive Bayes classifier has the maximum AUC value which is 0.6356. So in our case of data that we have taken, Naive Bayes is best by which we are predicting the claim will be approved fast or not.

▪ Future Scope:

We designed a system to detect the claims for which approval could be accelerated leading to faster payments or claims for which additional information is required before approval.

In a world shaped by the emergence of new uses and lifestyles, everything is going faster and faster. When facing unexpected events, customers expect their insurer to support them as soon as possible. However, claims management may require different levels of check before a claim can be approved and a payment can be made. With the new practices and behaviors generated by the digital economy, this process needs adaptation thanks to data science to meet the new needs and expectations of customers.

Claims management is a powerful lever to activate the enhancement of the Customer Experience, ranked #1[1] among factors leading to client satisfaction; ahead of purchasing experience, call center interaction and underwriting. Insurers' claims strategy over the next two years reflect this trend: improving the customer satisfaction comes first[1], before decreasing claim unit cost or scaling up operations to absorb growth.

In practice, customers associate claims to critical moments of truth, i.e. where they see if it was worth it for them to pay premiums. However, insurers still consider claims as a cost center. For years they have taken steps to build resilient organizations, with spotlights away from customers:

- Heavy IT investments;
- Design of lean organizations for short term cost savings;
- Renegotiation of supplier and claims handling arrangements; and
- To a lesser extent, investment in digital front-end.

As a result, Claim Managers are still pressured to reach more cost optimization and higher performance expectations. Up to 32%[2] of policyholders are unsatisfied with their "claims" process (compared to 14%[2] for "purchase"). As Claims are not managed as an experience yet, insurers have to leverage the momentum launched by other industries and early birds within their industry, such as direct players.

All is not lost though. Through an explosion in data volume and sources and given outdated technologies that limit the ability to gain integrated customer views and provide responsive service, the wind of

change is blowing on Claims management, to evolve from a step in the policy lifecycle, to a part of the customer journey.
A claims adjuster is responsible for determining how much an insurance company pays to a claimant. The adjuster analyzes the property, and the damage reported to get an estimated amount of the event. He/she also visits the site of damage or going to where the wreckage of a car is stored. Other duties include examining the insured property, conduct a background search for information regarding the damage, and provide a comprehensive report of the claim. He/she also gathers all the necessary information on behalf of the insurance company from videos to audio evidence. Lastly, they also consult the police and retrieve reports in case of fire and road accidents

❖ CERTIFICATE

This is to certify that **PURNENDU DAS** of Government College of Engineering and Textile Technology, Berhampore, registration number: 161110110022, has successfully completed a project **on "Claim Management" using Machine Learning(Python)** under the guidance of **Mr. Titas RoyChowdhury**.

Mr. Titas Roy Chowdhury
Globsyn Finishing School

❖ CERTIFICATE

This is to certify that **ATANU JANA** of Government College of Engineering and Textile Technology, Berhampore, registration number: 171110120002, has successfully completed a project **on "Claim Management" using Machine Learning(Python)** under the guidance of **Mr. Titas RoyChowdhury**.

Mr. Titas Roy Chowdhury
Globsyn Finishing School

❖ CERTIFICATE

This is to certify that **ANIRUDDHA MAITY** of Government College of Engineering and Textile Technology, Berhampore, registration number: 16110110001, has successfully completed a project **on "Claim Management" using Machine Learning(Python)** under the guidance of **Mr. Titas RoyChowdhury**.

Mr. Titas Roy Chowdhury
Globsyn Finishing School

❖ CERTIFICATE

This is to certify that **Bibhas Gayen** of Government College of Engineering and Textile Technology, Berhampore, registration number: 161110110008, has successfully completed a project **on "Claim Management" using Machine Learning(Python)** under the guidance of **Mr. Titas RoyChowdhury**.

Mr. Titas Roy Chowdhury
Globsyn Finishing School

❖ CERTIFICATE

This is to certify that **LOPAMUDRA ROY** of Government College of Engineering and Textile Technology, Berhampore, registration number: 16110110016, has successfully completed a project **on "Claim Management" using Machine Learning(Python)** under the guidance of **Mr. Titas RoyChowdhury**.

Mr. Titas Roy Chowdhury
Globsyn Finishing School