A report Based on

ON JOB TRAINING

Carried out at

Bajaj Allianz Life Insurance,

Pune

SUBMITTED BY

Ms. Patil Rajshri Durvas

.

To The

Department of Statistics
School of Mathematical Sciences,
Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon
Academic Year 2024-26

Internal Mentor: Mr. Manoj C. Patil

External Mentor: Mr. Mukund Gopal Kulkarni

CERTIFICATE

This is to certify that **Ms. Rajshri Durvas Patil** (Seat No. **380658**) has successfully completed her On-Job Training **(OJT)** from 19/05/2025 to 18/06/2025 at **Bajaj Allianz Life Insurance, Pune**.

During this period, she was actively involved in data-driven projects related to life insurance policy analysis and product recommendation systems. She worked sincerely under the **Statistics and Analytics domain**, contributing to the creation of synthetic datasets, performing exploratory data analysis (EDA), and building rule-based recommendation models using Python and R.

Her training was conducted under the guidance of **Mr. Kulkarni Mukund Gopal (B06)**, who mentored her in understanding the practical aspects of customer segmentation, product design logic, and decision support systems in the insurance industry.

Her dedication, technical learning, and analytical contribution throughout the training period have been commendable. We wish her continued success in her future academic and professional endeavors.

Prof. Kirtee K. Kamalja

(*Head of Department of Statistics*)

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to everyone who contributed to the successful completion of my internship at the **Bajaj Allianz Life Insurance, Pune** particularly in the **Actuarial Department**

First and foremost, I express my heartfelt gratitude to **Bajaj Allianz Life Insurance, Pune** for providing me the opportunity to undergo a one-month OJT & Internship as a part of the M.Sc. in Statistics curriculum. It was a valuable learning experience that allowed me to explore real-life applications of statistical concepts in the insurance industry.

I am deeply thankful to **Mr. Gopal Mukund Kulkarni, Bajaj Allianz Life Insurance, Pune** , my mentor at Bajaj Allianz Life Insurance, for his constant support, guidance, and encouragement throughout the training period. His guidance, technical expertise, and passion for applying statistics in real-world insurance operations provided an inspiring and highly productive learning environment. Under his mentorship, I gained valuable exposure to practical applications of statistics & insurance, and his support and leadership were instrumental in shaping a meaningful OJT experience.

. Furthermore, I would also like to extend my sincere thanks to Mr. Manoj C. Patil Sir, my Mentor from Department of Statistics at School of Mathematical Sciences, Kavayitri Bahinabai Chaudhari North Maharshtra Universirty Jalgaon, for his academic mentorship support and guidance throughout the entire month.

I would also like to thank my teammates and fellow classmates who participated in the OJT and project at Bajaj Allianz, for their cooperation, ideas, and teamwork, which made the entire experience more enriching and collaborative.

With a background in Actuarial Science, this OJT experience has helped me apply statistical knowledge in real-world insurance scenarios and strengthened my understanding of data-driven decision-making in the life insurance domain This internship has been an enriching experience and I am grateful to everyone who played a part in it.

Your Sincerely,

Rajshri Patil

Table of Contents

INTRODUCTION

This report presents a comprehensive account of the work carried out during a one-month On-the-Job Training (OJT) at **Bajaj Allianz Life Insurance**, Pune. The internship aimed to bridge academic concepts from statistics and actuarial science with real-world data applications in the insurance sector.

The project is structured in two primary phases:

- **Data Generation and Exploratory Data Analysis (EDA):**
  A synthetic dataset was programmatically created to simulate realistic customer profiles and insurance behaviors, inspired by Bajaj Allianz Life's product mix. EDA techniques were applied to identify trends, segment behaviors, and validate data integrity. This formed the foundation for downstream modeling.

- **Development of a Rule-Based Recommendation System:**
  Using association rule mining techniques, particularly the **Apriori algorithm**, four distinct recommendation models were built iteratively. These ranged from manually defined rule tables to dynamic, data-driven systems integrated with user interfaces like Streamlit, Tkinter, and R Shiny. The goal was to recommend suitable **Benefit Terms** or **Policy Terms** for customers based on demographic and financial inputs.

Additionally, the report includes a business overview of **Bajaj Allianz Life Insurance**, its service offerings, digital initiatives, and key performance metrics. This contextual foundation helped align the data modeling with actual business objectives.

❖ Summary

This internship project not only strengthened technical proficiency in Python, R, and statistical modeling, but also provided practical insight into the insurance domain, recommendation systems, and the challenges of real-world deployment. The outcome is a robust, explainable, and scalable framework that simulates how modern insurers can leverage data to enhance customer engagement and policy personalization.

Internship Objectives and Purpose

🔷 1. Bridging Academic Knowledge with Industry Practice

• Objective:
Apply classroom knowledge of statistics and analytics to solve practical problems in the life insurance sector.
• Purpose:
Understand how customer segmentation, policy analysis, and data-driven decision-making are applied in real-time business settings.

🔷 2. Gaining Industry-Specific Exposure

• Objective:
Familiarize with the internal structure, workflow, and functional areas of a leading life insurance company.
• Purpose:
Understand key insurance operations such as premium collection, risk assessment, and policy servicing through departmental coordination.

🔷 3. Applying Analytical Tools and Statistical Techniques

• Objective:
Use analytical tools and algorithms to analyze real or simulated business data.
• Purpose:
Gain hands-on skills in Python and Excel by working on projects like the Apriori-based recommendation model.

🔷 4. Professional Development and Workplace Readiness

• Objective:
Build soft skills and understand corporate professionalism.
• Purpose:
Improve communication, time management, and teamwork through mentorship and collaborative work.

🔷 5. Contributing to Organizational Projects and Goals & Enhancing Career Preparedness

• Objective:
Engage in real-world projects that support business outcomes & Explore career paths in analytics, actuarial science, and insurance.
• Purpose:
Create value by contributing to a recommendation engine that enhances customer engagement and personalization & Align academic training with practical exposure to prepare for future roles in industry or research.

Introduction to Bajaj Allianz Life

Bajaj Allianz Life Insurance Company Limited is one of India's premier private life insurers. It is a joint venture between:

- **Bajaj Finserv Ltd.** – one of India's most respected non-banking financial companies
- **Allianz SE** – a global leader in insurance and asset management based in Germany

The company blends **global expertise** with **local market understanding** to deliver a comprehensive range of life insurance solutions designed for Indian customers.

Founded in **2001**, Bajaj Allianz Life has grown into a trusted brand in the insurance space, known for innovation, reliability, and customer-centric offerings tailored for:

- Protection
- Investment
- Savings
- Child Education
- Retirement Planning

➢ 🔷 Services Offered

Bajaj Allianz Life provides a diverse portfolio of life insurance products, including:

- ✅ **Term Insurance Plans** (e.g., *Smart Protect Goal*)
- ✅ **Unit Linked Insurance Plans (ULIPs)** (e.g., *Future Gain*, *Goal Assure*)
- ✅ **Savings and Investment Plans**
- ✅ **Child Plans**
- ✅ **Retirement and Pension Plans**
- ✅ **Health and Critical Illness Riders**

The company also leads in **digital innovation**, enabling:

- Online policy management
- Premium payment portals
- Mobile app and WhatsApp-based servicing

➢ 🔷 History and Growth

- 🗓️ **Established**: August 2001
- 📍 **Headquarters**: Pune, Maharashtra
- 🌐 **Pan-India Presence**:
  - 500+ branches across the country
  - 80,000+ agents and advisors
  - Strong reach via bancassurance, digital platforms, and broker networks

Bajaj Allianz Life was among the **early adopters of digital technology** in insurance and consistently ranks high in:

- Claim settlement ratio
- Customer service metrics
- Innovation awards

➢ 🔷 Mission and Vision

- **Mission**: To offer **value-packed**, **customer-focused** life insurance solutions that ensure the **financial well-being** of every Indian family.
- **Vision**: To be a **transparent**, **trustworthy**, and **digitally forward** insurer that exceeds customer expectations.

➢ 🔷 Recognition and CSR

- 🏅 Winner of numerous awards in **digital transformation**, **customer service**, and **innovation**
- 🤝 Active in **CSR initiatives** focused on:
  - **Health**
  - **Education**
  - **Environmental sustainability**

These efforts reflect the company's **holistic commitment** to both **business excellence** and **social responsibility**.

General Information about Bajaj Allianz Life

➢ 🔹 3.1 PRODUCT RANGE :

Bajaj Allianz Life offers products across every major insurance category:

- **Protection Plans**: *Term insurance policies like Smart Protect Goal and Life Secure.*
- **ULIPs (Unit Linked Insurance Plans)** : *Investment-linked plans such as Future Gain and Goal Assure.*
- **Traditional Endowment and Savings Plans** : *Save Assure, Guaranteed Income Goal*
- **Retirement & Pension Plans** : *Retire Rich*
- **Child Plans** : *For education and future security, such as Young Assure.*
- **Health Riders** : Critical illness and add-on benefits

These are designed for all stages of life — from income growth to retirement.

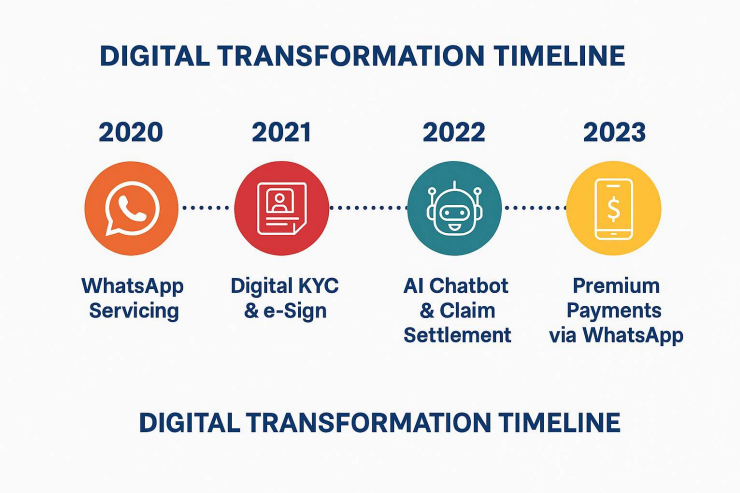
A circular chart

➢ 🔹 3.2 DIGITAL FIRST APPROACH :

Bajaj Allianz Life has adopted a 'Digital First' strategy, offering a seamless digital customer experience through:

- Online policy purchase, renewal, and tracking
- WhatsApp and mobile app servicing
- e-KYC, e-signatures, and digital documentation
- AI-powered chatbots and claims automation
- This enhances service speed, 24/7 access, and paperless convenience.
-

Timeline or roadmap showing digital milestones

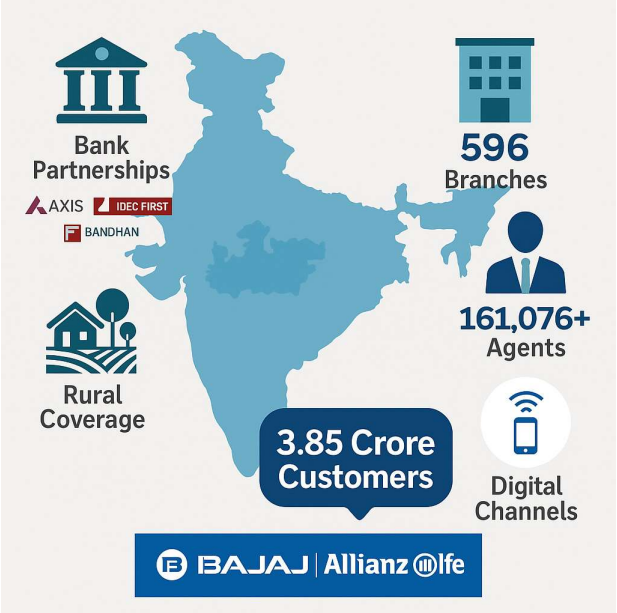


➢ 🔹 3.3 CUSTOMER BASE & DISTRIBUTION :

- Operates through over 596+ branches across India.
- Offers products through a strong network of 161,000+ agents, corporate partners, brokers, and digital channels.
- Serves 38.5 million+ customers, with a growing rural outreach.
- Partnerships with major banks for bancassurance (Axis, Bandhan, IDFC First)

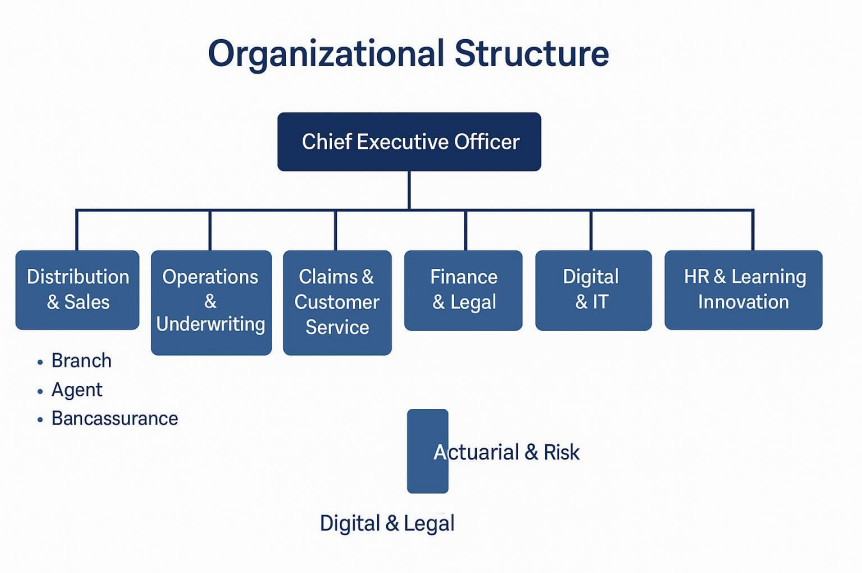
Combines rural and urban outreach with strong digital coverage.

India map



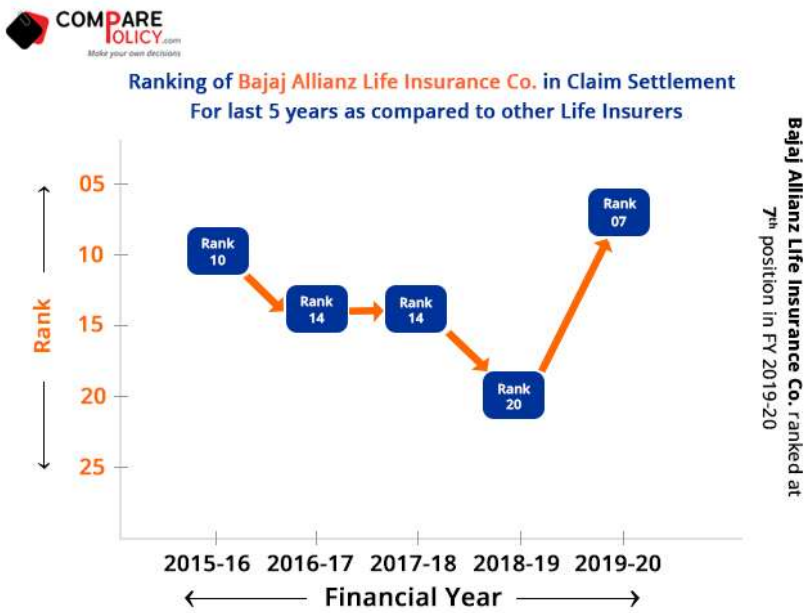
➢ **◆ 3.4 EMPLOYEE STRENGTH & WORKFORCE :**

- 23,000+ full-time employees
- Training-driven culture with upskilling for agents and staff
- Focus on wellness, innovation, and leadership development
- Strong team collaboration across departments



➢ 🔹 3.5 CLAIM SETTLEMENT & PERFORMANCE :

- **Claim Settlement Ratio (CSR)** of 99.29% in FY 2024–25
- CSR improved consistently from 98.48% (FY 2020–21)
- Outperforms private insurer average (~99%)
- Emphasizes trust, fast service, and transparency.



➢ 🔹 3.6 AWARDS & RECOGNITION :

- Recognized for:
  - Customer service excellence
  - Digital transformation
  - Claims efficiency
- Honored at ET Insurance Summit, BFSI Awards, and more
- Noted for CSR work in education, health, and environment

Special Products by Bajaj Allianz Life

Bajaj Allianz Life Insurance offers a **well-diversified product portfolio** to meet the varied financial goals and life stages of its customers. These include:

🔷 Term Plans

Provide high life cover at affordable premiums with flexibility to add riders.

- **Smart Protect Goal**
  A popular term plan with options for:
  - Critical Illness Benefit
  - Accidental Death Benefit
  - Return of Premium
  - Waiver of Premium on disability

🔷 ULIPs (Unit Linked Insurance Plans)

Offer investment-linked insurance where premium is partly invested in equity/debt funds.

- **Goal Assure**
  Features:
  - Loyalty additions
  - Return of mortality charges
  - Multiple fund choices
  - Return Enhancer on maturity
- **Future Gain**
  Balanced growth + protection ULIP with medium to long-term investment options

🔷 Savings and Endowment Plans

These plans offer guaranteed returns with life cover, ideal for long-term disciplined savings.

- **Save Assure**
  Traditional endowment with guaranteed maturity payout
- **Guaranteed Income Goal**
  Offers regular income post maturity for a fixed number of years

🔷 Child Plans

Help parents plan for a child's education and life goals while covering uncertainties.

- **Young Assure**
  Provides milestone payouts and waiver of future premiums in case of parent's death

🔷 Retirement and Pension Plans

Designed to provide **financial independence** during retirement.

- **Retire Rich**
  A guaranteed annuity plan with options like:
  - Life annuity
  - Annuity with return of purchase price
  - Joint life option for spouse

🔷 Health Riders

Optional add-ons that enhance base policy protection.

- **Critical Illness Rider**
- **Waiver of Premium Rider**
- **Accidental Death Benefit Rider**

Product Category Summary Table :

Product Category	No. of Variants	Examples
Term Plans	6	Smart Protect Goal, Life Secure
ULIPs	5	Goal Assure, Future Gain
Savings Plans	4	Save Assure, Guaranteed Income
Child Plans	2	Young Assure
Retirement Plans	3	Retire Rich
Health Riders	3	Critical Illness, ADB, Waiver

Life Stage Mapping Table :

Customer Life Stage	Product Type	Example Plan
Young Professionals	Term Plan / ULIP	Smart Protect Goal, Goal Assure
Married with Dependents	Child Plan / Endowment	Young Assure, Save Assure
Pre-retirement (40–55)	Savings / Pension	Guaranteed Income Goal
Retired Individuals	Pension Plan / Annuity	Retire Rich
Rural / Low-Income	Micro Insurance / Simple Plans	POS Goal Suraksha

Branch Distribution Insight

To support its wide-ranging product portfolio, **Bajaj Allianz Life Insurance** has built a **robust physical and digital distribution network** across India. This ensures insurance access even in underserved and remote regions.

Key Highlights:

- **596+ branches** across India (as of FY 2025)
- Presence in **urban, semi-urban, and rural locations**
- Branches strategically located in:
  - Major metros: **Mumbai, Pune, Delhi, Kolkata, Chennai, Hyderabad**
  - Tier-2 cities: **Nagpur, Jaipur, Lucknow, Bhopal, Coimbatore**
  - Rural hubs: Covered via micro-offices and agent networks

Distribution Channels:

Channel Type	Description
Physical Branches	500+ full-service branches handling sales, service & claims
Agents & Advisors	161,000+ registered agents nationwide
Bancassurance	Partners with banks like Axis, IDFC First, Bandhan, etc.
Digital Channels	Website, mobile app, WhatsApp bot for 24×7 self-service

Reach Strategy:

- Emphasis on **digital hybrid presence**: Combines online policy servicing with human advisors
- **Customer service zones (CSZs)** within branches offer walk-in support
- Integration of **POS (Point of Sale) products** for low-ticket, rural insurance sales

Pune Division - A Strategic Hub

The **Pune Division** is one of Bajaj Allianz Life's most critical operational centers. It plays a vital role in managing regional and pan-India operations, particularly in **technical, service, and digital innovation** functions.

Key Departments & Staff :

Department	Employees	Key Functions	Special Notes
Sales & Distribution	120	Policy sales, agent training	Supports entire Maharashtra
Customer Service	80	Grievances, omnichannel queries	Physical + WhatsApp + App
Underwriting	45	Medical and financial risk assessment	Medical & Financial checks
Actuarial & Risk Mgmt	35	Product pricing, data analytics	Works closely with dev teams
IT & Digital Operations	50	Website, mobile app, chatbot tools	Manages AI, servicing portals
Claims Processing	40	Claim verification & payout	Supports 99% CSR benchmark
Product Development	30	New product innovation & rollout	Uses analytics + market feedback
HR & Admin	25	Payroll, training, internship onboarding	Coordinates OJT programs
Marketing & Branding	20	Offline and online campaigns	Regional branding and outreach
Compliance & Audit	10	IRDAI compliance, internal audits	Ensures regulatory alignment

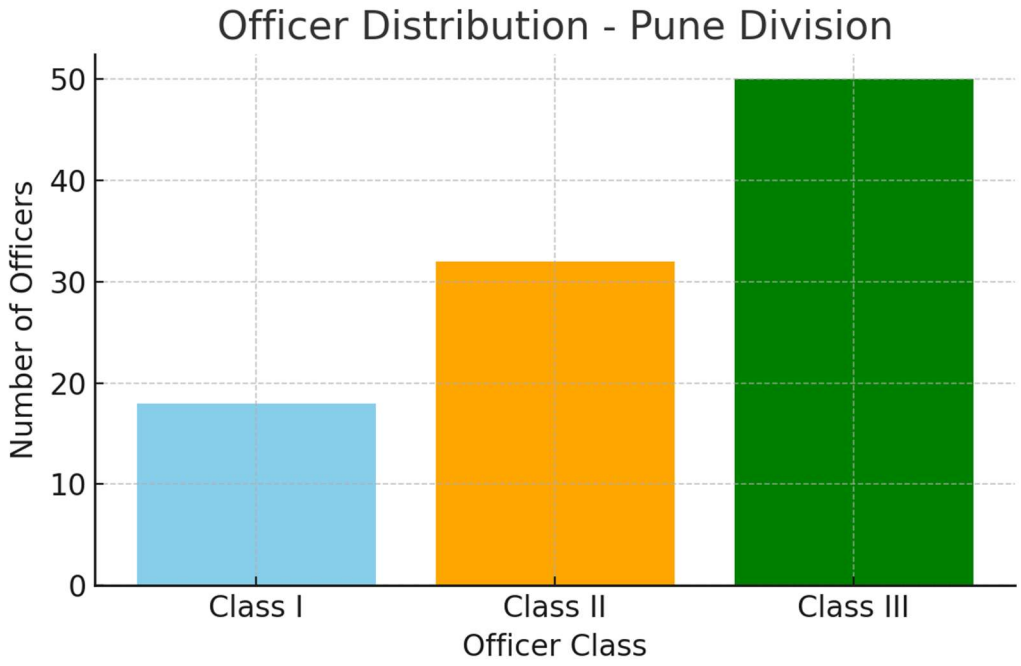
Strategic Significance :

- The **Pune branch** is not only a sales and service node but also a **center for innovation**, hosting core teams for:
    - AI chatbot development
    - Product pricing and analytics
    - Internship and training programs (like OJT)

Officer Class Structure – Pune

These **officers** are responsible for strategic decision-making, departmental leadership, and operational execution within the **Pune hub**.

Officer Class	Sample Designations	No. of Officers
Class I	Senior Manager, AVP, VP	18
Class II	Manager, Deputy Manager	32
Class III	Executive, Assistant Manager	50



Project Work

Section 1: Data Generation and EDA

1.1 Objective :

- To generate realistic synthetic life insurance data based on Bajaj Allianz Life's product mix.
- To analyze customer profiles, product behaviors, and policy performance using Exploratory Data Analysis (EDA).
- To identify patterns and correlations that can support product recommendations and customer segmentation.

1.2 Understanding the Dataset :

My dataset includes **100,000 rows** and **23 variables** representing customer profiles, policies, and outcomes.

Key Variable Categories :

Category	Variables
Demographics	Age_Group, Gender, Marital_Status, Occupation, Education_Level
Geography	Location, Region
Financial Info	Income, Declared_Income, Number_of_Dependents
Policy Details	Product, Policy_Type, Policy_Status, Benefit_Term, Premium_Term, Sum_Assured, Mode_of_Premium_Payment, Premium_per_Payment, Premium_Term_Category
Temporal	Date_of_Purchased_(years_ago), Time_to_Maturity
Outcome	Claim_History, Children

Variable Notes:

- `Product` and `Policy_Type` are aligned with Bajaj Allianz offerings (e.g., ULIPs, Term Plans).
- `Premium_Term_Category` classifies premium terms as short, medium, or long.
- `Claim_History` is a binary outcome useful for risk analysis.

1.3 Data Generation Method :

The synthetic data was created using Python with the following logic. This rule-based generation ensures that customer-product relationships are realistic and business-aligned (see Bajaj_Life_Data_Generation_Notebook.ipynb)

Tools Used:

- pandas, numpy, random, Faker for synthetic generation

  Ex. Code

```
def generate_sum_assured(product):
return {
'Smart Protect Goal': np.random.randint(500000, 1500000),
'Save Assure': np.random.randint(200000, 1000000),
'Goal Assure': np.random.randint(400000, 1200000),
'Child Advantage Plan': np.random.randint(300000, 1000000),
'Income Assure': np.random.randint(250000, 800000),
'Future Gain': np.random.randint(500000, 1200000)
}[product]
```

- Logic-based conditional generation for attributes

  Ex. Code

```
if premium_term < 10:
premium_term_cat = 'short'
elif premium_term <= 20:
premium_term_cat = 'medium'
else:
premium_term_cat = 'long'
```

Generation Logic Examples:

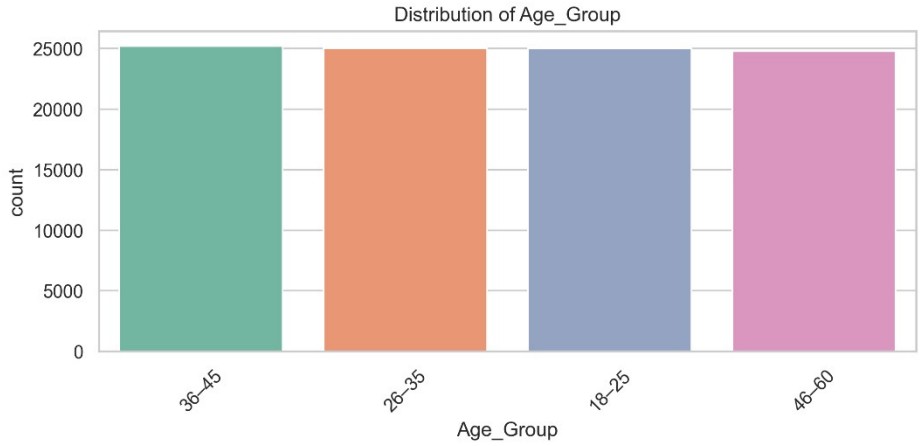
Customer Attribute Logic	Product Allocation
Age 18–25, single, low income	Term plans (Smart Protect Goal)
Age 26–35, middle income, married with children	ULIPs or Child Plans
Age 46–60, nearing retirement	Retirement or guaranteed income plans
Declared income > ₹10L, urban location	High sum assured ULIP or combo products
Policy term logic	`Time_to_Maturity = Benefit_Term - Date_of_Purchase`

1.4 Exploratory Data Analysis (EDA)

- ✅ **Univariate Analysis** (count plots, histograms)
- ✅ **Bivariate Analysis** (age vs product, income vs sum assured)
- ✅ **Multivariate Segmentation** (facet grids, crosstabs)
- ✅ **Risk & Claims Insight** (claim vs region, occupation, etc.)
- ✅ **Advanced Segmentation** (Product + Age + Region + Income heatmaps)

Ex. Code & Output :

```
# count plots for all categorical columns
for col in categorical_columns:
plt.figure(figsize=(8, 4))
sns.countplot(data=df, x=col, order=df[col].value_counts().index,
palette="Set2")
plt.title(f"Distribution of {col}")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



1.5 Key Insights from EDA

Insight Type	Key Finding
📊 **Demographics**	Most customers are aged 36–45, with a nearly even gender split
🏷 **Popular Products**	Smart Protect Goal (Term) and Goal Assure (ULIP) are most common
💵 **Sum Assured Trends**	Higher sum assured seen for income > ₹6L and urban locations
🧠 **Claims Pattern**	Slightly higher claim history in business owners and ULIP policy holders
📈 **Policy Behavior**	Younger customers prefer shorter terms and flexible premiums

Project Work

Section 2: Recommendation Model Development

2.1 Objective :

The objective of this project is to develop a **rule-based insurance recommendation system** capable of suggesting the most suitable **Benefit Term** for a customer based on their **demographic and financial profile**.

Over the course of the internship, **four distinct models** were developed — evolving from manually defined logic to fully data-driven rule extraction using the **Apriori algorithm** in both Python and R environments.

🎯 Core objective of the Project:

- To simulate a real-world recommendation engine for life insurance by identifying patterns in customer preferences and translating them into actionable rules.
- To assist insurance advisors and digital platforms in offering personalized product suggestions based on customer data inputs.
- To enhance understanding of how statistical concepts, such as association rule mining, can be applied in business contexts like policy design, customer targeting, and decision support.

🎯 The Project Goals Were:

- To explore various approaches to rule-based recommendations:
    - Starting with manual rule tables and evolving to Apriori-based rule mining
- To leverage both synthetic and realistic insurance datasets:
    - Including customer attributes like Premium Term, Income, Gender, Occupation, and Marital Status
- To apply the Apriori algorithm for discovering frequent itemsets and association rules that reflect real-world policy preferences
- To implement user-facing interfaces using **Streamlit**, **Tkinter**, and **R Shiny** that allow users to:
    - Input a customer profile
    - Receive a personalized Benefit Term recommendation
    - View rule confidence and explanation for transparency

Ultimately, the goal was to simulate how **data-driven recommendation systems** can support insurance agents, customers, or online platforms in making **personalized, explainable product suggestions** aligned with policyholder needs.

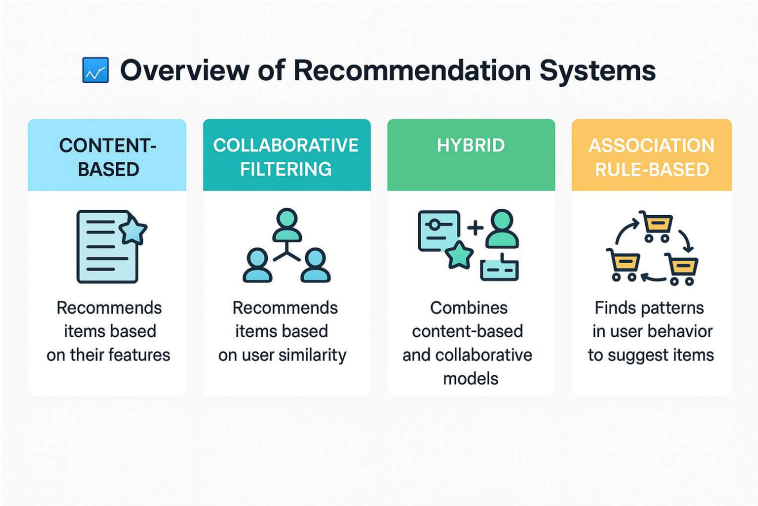
2.2 Model Evolution Overview :

A **recommendation system** is an algorithmic tool that suggests products, actions, or content to users based on patterns in existing data. They are widely used in platforms like Netflix, Amazon, and Spotify.

In this project, we adapt the same concept to the **insurance domain**, where instead of recommending a product or movie, the system suggests a suitable **Benefit Term** (coverage duration) for a life insurance customer.

There are several types of recommendation systems, such as:

- **Content-Based Filtering**: Recommends items similar to what a user has liked in the past.
- **Collaborative Filtering**: Recommends based on preferences of other users with similar behavior.
- **Rule-Based Filtering** (used in this project): Uses logical patterns and conditional rules discovered from data.



In this project, we implement a **rule-based recommendation system** using a technique from data mining known as **association rule mining**. Instead of recommending a product or a movie, the system suggests the most appropriate **Benefit Term** for a life insurance policy based on customer profile attributes.

For example:
If many customers with `income = 6-10L`, `occupation = engineer`, and `premium_term = 10` commonly choose a **Benefit Term** of 20 years,
then the system learns this pattern and can recommend it to a new user with similar characteristics.

Why This Approach Works Well in Insurance

- Insurance decisions are driven by structured demographic and financial data — perfect for rule-based analysis.
- Unlike collaborative filtering, we **don't need historical user feedback** (like movie ratings) — just past choices.
- It creates **transparent, explainable rules** — ideal for regulated industries like insurance.

By using this system, we simulate how an insurance advisor or a digital portal could offer **personalized product suggestions** without needing complex machine learning models.

Why Association Rule Mining (Apriori)?

In this project, the goal is to recommend a suitable **Benefit Term** to customers based on their demographic and financial characteristics. For this purpose, we adopted a technique from the field of **data mining** known as **association rule mining**, specifically the **Apriori algorithm**.

What Is Association Rule Mining?

Association rule mining is used to discover relationships or **co-occurrence patterns** between variables in large datasets. It's widely used in market basket analysis — for example:

"If a customer buys bread and butter, they are likely to buy milk."

In this context, we apply the same principle to insurance customer data:

"If a customer has income of 6–10L, is an engineer, and chooses a premium term of 10 years, they are likely to choose a benefit term of 20 years."

These **if–then rules** are expressed in terms of:

- **Antecedents** (conditions)
- **Consequent** (recommended Benefit Term)
- **Support**, **confidence**, and **lift** (to evaluate strength)

Why Use Apriori in This Project?

Apriori is well-suited for this problem because:

- ✅ **Categorical data**: Our features (Gender, Income, Occupation) are non-numeric and ideal for rule-based logic.
- ✅ **No target dependency**: Unlike regression or classification, Apriori doesn't assume a fixed target variable — we extract rules where `Benefit_Term` naturally appears in the consequent.

- ✅ **Interpretability**: The rules generated are transparent and easy to explain — important for industries like insurance where decisions must be auditable.
- ✅ **Scalability**: Apriori can handle large transaction-style datasets by pruning low-frequency patterns using minimum support thresholds.

Example of a Rule Generated

Rule	Confidence	Lift
If premium_10, income_6-10l, occupation_engineer → benefit_20	0.48	1.5

This rule indicates that **48%** of customers with those characteristics chose a benefit term of **20 years**, making it a strong candidate for recommendation.

In summary,

Apriori provides an efficient, explainable, and data-driven approach to recommending insurance product features based on historical customer behavior.

Models :

As a part of internship we have built the four versions of the recommendation system during the internship, outlining the progression from basic rule-based logic to Apriori-driven recommendation systems in both Python and R environments.

Model Summary Table :

Model Version	Approach	Tech Stack	Dataset Source	Recommendation Logic
Model 1	Manual rule-based	Python + Streamlit	LIC synthetic dataset	Handwritten rules (if-else)
Model 2	Apriori on all combinations	Python + Tkinter	Synthetic dataset(combinations)	Rule mining via Apriori
Model 3	Apriori on real Bajaj data	Python + Streamlit	Synthetic dataset	Rule mining + live GUI
Model 4	Rule-based in R	R + Shiny	Synthetic dataset	Apriori with R integration

2.3 Model 1: Manual Rule-Based System (LIC Dataset)

The first version of the recommendation system was designed as a **manually coded rule-based model** that recommends suitable LIC policies based on user inputs. This version emphasized **logic transparency and simplicity**, serving as a foundational prototype to demonstrate how customer profiles can be mapped to policy types using structured business logic.

➢ Dataset Description

A **synthetic dataset** was created using Python to simulate realistic customer scenarios for four flagship LIC products:

- LIC Jeevan Umang
- LIC Jeevan Lakshya
- LIC Tech Term
- LIC New Endowment Plan

The dataset included over **60,000 records**, combining demographic and financial variables such as:

- Age Group
- Income Group
- Preferred Policy Duration
- Customer Priority (Savings + Protection / Protection only)

➢ Methodology

This model followed a **deterministic logic-based approach**:

📌 **Step 1:Rule Formulation**:

- A set of decision rules was created based on LIC product characteristics, age-income compatibility, and savings/protection priorities.
- These rules were framed into a **decision table**, mapping every possible combination of user traits to a recommended policy.

📌 **Step 2:Python Implementation**:

- The rules were implemented using if-else and elif statements in a function (recommend_policy) within the Streamlit interface.
- User inputs were captured through dropdowns and radio buttons.

📌 **Step 3:Streamlit Interface**:

- An interactive app was built using Streamlit.
- The app prompts the user for:

- Age Group
- Income Bracket
- Policy Term Preference
- Coverage Priority

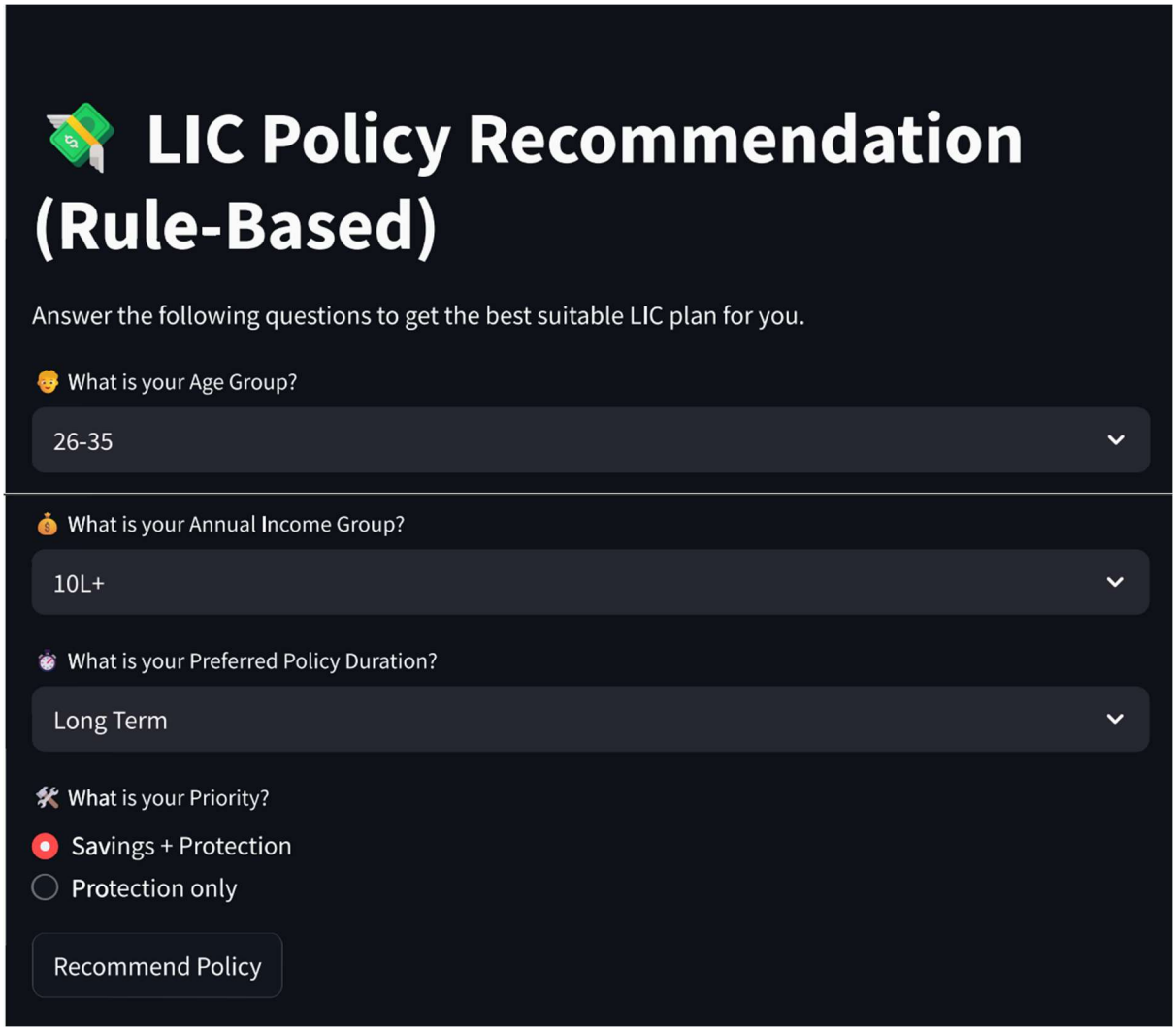
- Based on this input, the app displays the recommended LIC policy along with a short explanation of its features

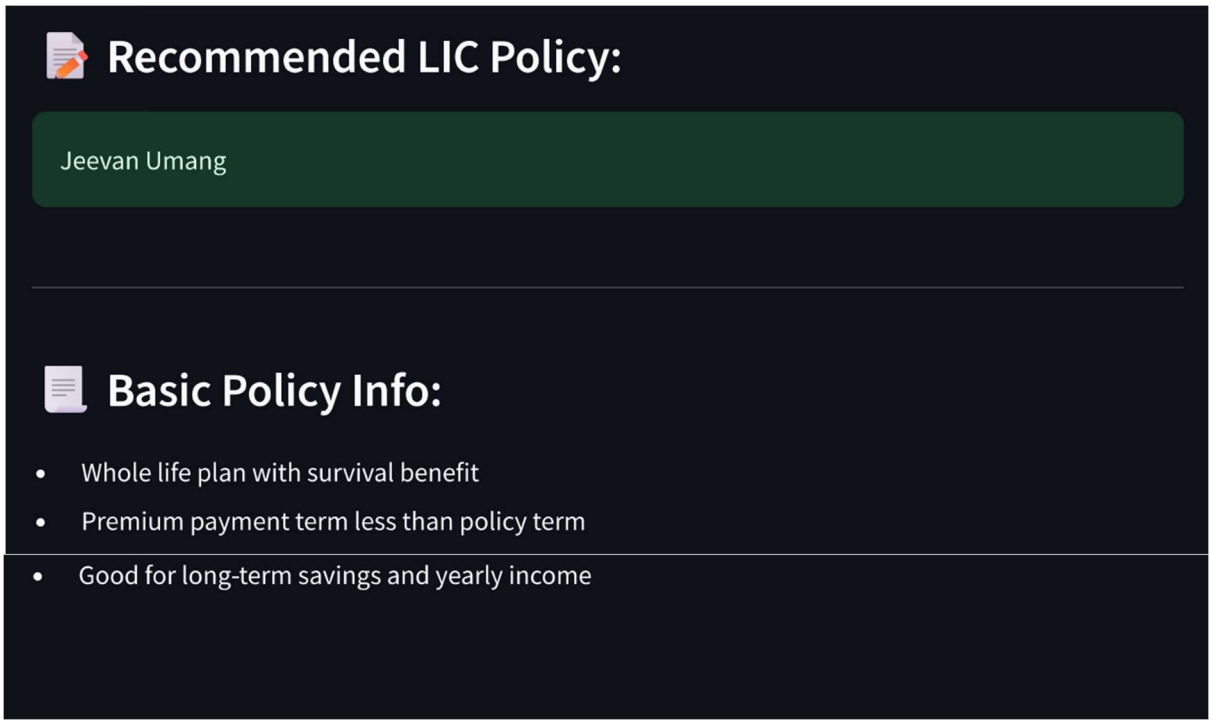
Sample Decision Rules :

Age Group	Income Group	Term	Priority	Recommended Policy
18–25	<3L	Short	Savings + Protection	LIC Jeevan Lakshya
18–25	<3L	Long	Savings + Protection	LIC Jeevan Umang
Any	Any	Short	Protection Only	LIC Tech Term
Any	Any	Long	Protection Only	LIC Jeevan Umang

Full rule table included in the decision logic file.

🖥️ Interface Screenshot :





📊 Results & Observations (Model 1)

- Returned recommendations accurately for basic cases
- All logic was transparent and hardcoded
- No ability to adapt or score confidence
- Performed well for educational or controlled demos
- Ideal as a first prototype but not scalable

✅ Strengths

- Fully explainable logic with human-readable conditions
- No need for complex libraries or algorithms
- Quick development cycle and fast performance
- Served as a solid **baseline model**

⚠️ Limitations

- **Static and hardcoded** — no ability to learn from new data
- Doesn’t scale well to larger feature sets
- No support for dynamic pattern discovery or confidence scoring
- Limited flexibility across real-world product diversification

2.4 Model 2: Apriori-Based Recommender (Tkinter GUI) :

The second model aimed to evolve from a manually coded system to a **data-driven recommendation model** by using the **Apriori algorithm** to automatically learn rules from data. This version was designed to work with a **synthetically generated dataset** comprising all meaningful combinations of demographic and insurance features, providing more flexibility and realism than static rule logic.

Dataset Description

A synthetic dataset was constructed programmatically to cover **all possible combinations** of key customer attributes relevant to life insurance policy selection. The features included:

- Premium Term
- Gender
- Income Group
- Occupation
- Marital Status
- Location
- Children

These combinations allowed the Apriori algorithm to detect **frequent co-occurrences** that lead to particular Benefit Terms. The dataset was converted into transaction format and used to generate frequent itemsets and rules using mlxtend.

Methodology

The second model aimed to transition from static rule definitions to an **automated rule discovery approach** by using the **Apriori algorithm** on a **synthetic dataset** containing all feasible combinations of customer profile features.

This methodology involved two major components:

A. Rule Mining Using Apriori (Backend)

The backend component focused on preparing the data and applying association rule mining to discover useful patterns.

📌 Step 1: Synthetic Data Generation

- A structured dataset was created programmatically by combining all valid values across 7 features:
  Premium_Term, Gender, Income, Occupation, Marital_Status, Location, and Children.

- Each combination was paired with one or more realistic Benefit_Term values.
- This approach allowed coverage of both common and edge-case customer profiles.

📌 **Step 2: Data Preprocessing**

- All categorical values were **standardized** (e.g., lowercase, no spaces).
- Each record was converted into a list of labeled strings for Apriori compatibility, e.g.:

['premium_10', 'gender_male', 'income_6-10l', 'occupation_engineer', 'benefit_20']

📌 **Step 3: Transaction Encoding**

- The dataset was transformed using TransactionEncoder from mlxtend.preprocessing to produce a binary matrix, with True/False values indicating presence of items.

📌 **Step 4: Frequent Itemset Generation**

- The binary-encoded transaction data was passed to the apriori() function.
- A minimum support threshold (e.g., 0.005) was set to filter only meaningful itemsets.
- The output contained combinations of customer attributes that occurred frequently.

📌 **Step 5: Association Rule Generation**

- association_rules() was applied on frequent itemsets to generate **if–then rules**.
- Filtering criteria:
  - **Confidence ≥ 0.4**
  - **Lift > 1.0**
  - Rules must have Benefit_Term in the consequent
- Resulting rules were saved to a CSV (filtered_rules.csv) for easy integration with the GUI.

🖥️ B. Interactive GUI Using Tkinter (Frontend)

To demonstrate the usability of the recommendation model, a lightweight GUI was built using **Tkinter** and styled with **ttkbootstrap**.

📌 **Step 1. User Input Capture**

- Users select their profile values via dropdowns:
  - Premium Term, Income Group, Gender, Occupation, Marital Status, Location, Children
- A submit button triggers the recommendation engine.

📌 **Step 2 :Rule Matching Logic**

- The system reads the user inputs and transforms them into Apriori-style labels.
- It then searches filtered_rules.csv for all rules where:
    - All antecedent items are contained in the user's input
- The rules are sorted by **confidence** or **lift**, based on user preference.

📌 **Step 3:Top Recommendations Display**

- Up to **three matching recommendations** are shown in the UI.
- Each recommendation includes:
    - Recommended Benefit Term
    - Confidence score
    - Lift (strength of correlation)
    - Explanation of the matched rule

📌 **Step 4 :Logging**

- Each recommendation event is recorded in recommendation_log.csv, storing:
    - Timestamp
    - Input features
    - Output recommendation
    - Rule confidence

✅ This methodology created a **self-contained desktop application** that could dynamically respond to any synthetic customer profile, simulate real-time insurance advising, and display interpretable rule-based outcomes.

✅ **Sample Output Rule :**

If {'premium_10', 'income_6-10l', 'occupation_engineer'}

→ Then {'benefit_20'}

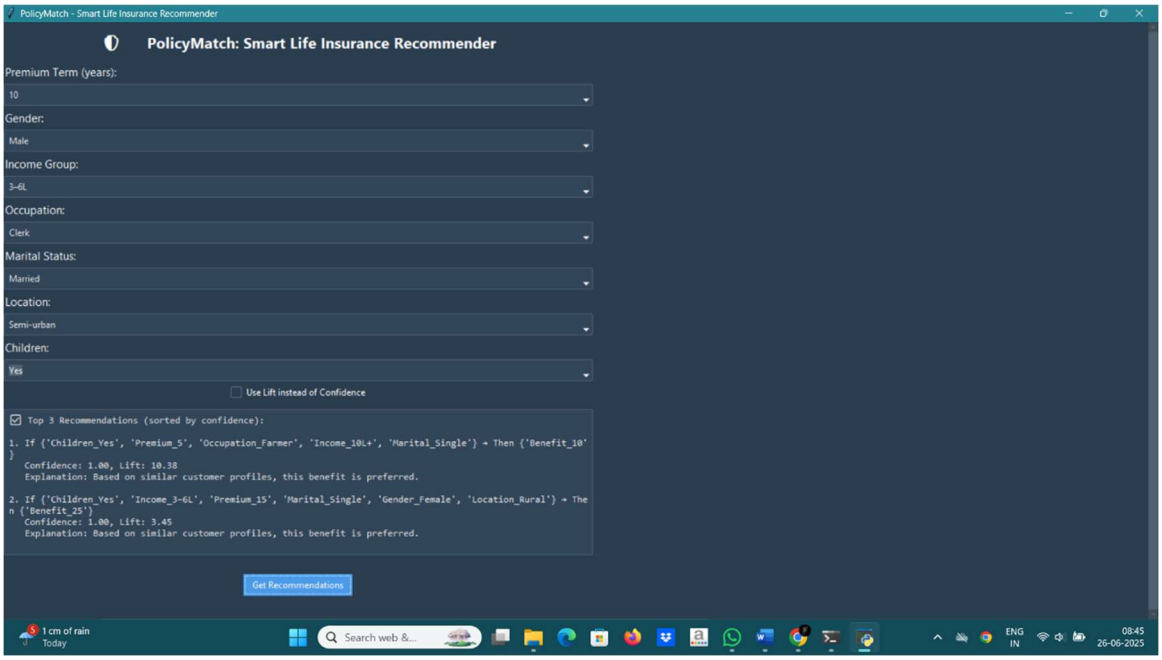
Confidence: 0.48, Lift: 1.52

Explanation: Based on similar customer profiles, this benefit is preferred.

📊 **Results & Observations (Model 2)**

- The system was able to generate and rank **hundreds of association rules** from synthetic data
- Recommendations were dynamically matched and presented with rule explanation
- Lift vs. Confidence toggle allowed deeper understanding of recommendation strength
- Tkinter provided a basic but functional UI for desktop environments

🖥️ Interface Screenshot :



✅ Strengths

- **Dynamic Rule Discovery & Scalable Logic**
  The use of the Apriori algorithm allowed the system to learn patterns from data, rather than relying on manually coded if-else statements & New data combinations or policy terms can be added without rewriting business logic — only retraining the rules is needed.
- **Fully Categorical & Interpretable**
  Each rule is expressed as a human-readable "if–then" format with accompanying confidence and lift scores, enhancing interpretability.
- **Multiple Recommendation Criteria & Synthetic Data Coverage**
  Users can choose to rank benefit term suggestions by either **confidence** or **lift**, giving flexibility in recommendation logic & also By generating all valid combinations, this model ensured that **rare but plausible** customer profiles were also represented and tested.
- **Traceability via Logging**
  All user interactions and recommendations are logged for analysis, useful for audit or user feedback tracking.

⚠️ Limitations

- Rules are **based on artificial combinations**, not real customer behavior
- GUI is **not web-based**; Tkinter limits portability compared to Streamlit
- No handling of edge cases where **no rule matches** exactly
- No continuous learning — rules must be re-mined if data changes

2.5 Model 3: Apriori Based Recommender (Streamlit Interface) :

The insurance industry offers a variety of policies with different premium terms and benefit terms. This project aims to create a personalized **recommendation model** to guide users in choosing a suitable **benefit term.** This is achieved by:

- The model uses the **Apriori algorithm** to find patterns between user demographic features and insurance benefit terms.
- A **Streamlit-based** web interface collects user inputs and displays recommendations accordingly .

Dataset Description

The dataset used in this project is a **synthetically generated life insurance dataset** consisting of 151,200 customer records. It simulates realistic combinations of demographic and financial features to model customer behavior with respect to insurance product terms.This data forms the foundation for applying the **Apriori association rule mining algorithm** to recommend a suitable **Benefit Term** based on customer profile attributes.

Dataset Overview :

- **File Name**: `large_insurance_dataset.csv`
- **Total Records**: 151,200
- **Format**: CSV with 7 input features and 1 target feature
- **Features Used**:

Feature Name	Type	Description
`Premium_Term`	Numeric	Number of years the customer pays premium
`Gender`	Categorical	Male / Female
`Income`	Categorical	Income range: `<3L`, `3-6L`, `6-10L`, `10L+`
`Occupation`	Categorical	e.g., Engineer, Teacher, Doctor, Business Owner
`Marital_Status`	Categorical	Single / Married
`Location`	Categorical	Urban, Semi-urban, Rural
`Children`	Categorical	Yes / No
`Benefit_Term`	Numeric	🔺 Target variable: number of years of life coverage

Each feature was transformed into a **standardized labeled string** (e.g., `income_6-10l`, `occupation_engineer`) to enable effective rule extraction using the Apriori algorithm.

Methodology

This section outlines the overall methodology used to build a **rule-based insurance benefit term recommendation system** using the **Apriori algorithm**.

Below is the step-by-step approach used to implement the model:

📌 Step 1: Data Transformation and Cleaning

Before feeding into the algorithm:

- All categorical values were converted to **lowercase**
- **Extra spaces** were removed using .str.strip()
- Each feature was **prefixed** (e.g., income_6-10l, gender_male, premium_10)
- This ensured formatting consistency and minimized rule duplication

📌 Step 2: (Optional) Exploratory Data Analysis (EDA)

EDA was performed in a separate notebook to:

- Understand distributions (e.g., most popular Premium Terms)
- Identify target patterns (e.g., Benefit Term vs Occupation)
- Validate the logic behind the rules discovered by Apriori

Visuals included histograms, count plots, and heatmaps using **Seaborn** and **Matplotlib**.

📌 Step 3: Boolean Encoding For Apriori

Since Apriori requires **transaction-like input**, so by using TransactionEncoder from mlxtend, each customer record was converted into a **binary-encoded transaction list**, where each feature becomes a boolean item , for example:

['gender_male', 'occupation_engineer', 'income_6-10l', 'benefit_20']

This format allowed the Apriori algorithm to scan for frequent co-occurring attribute sets.

That was done by:

- Renaming column values with consistent labels
- Merging all features into one list per row
- Result: A transaction list of 151,200 rows × 7 features

📌 Step 4: Frequent Itemset Generation (Apriori)

The Apriori algorithm was applied using mlxtend.frequent_patterns.apriori(), the following steps were performed:

- A **minimum support** threshold was defined (e.g., 0.005 or 0.01)
- Frequent itemsets were generated using:

  apriori(df_encoded, min_support=0.005, use_colnames=True)

This step identified all feature combinations that occurred frequently in the data.

📌 Step 5: Rule Extraction and Filtering

Using mlxtend.frequent_patterns.association_rules():

- Rules were extracted from frequent itemsets
- Only rules with **Benefit_Term in the consequent** were retained
- Additional filtering was done using:
  - Confidence ≥ 0.4
  - Lift > 1

Sample rule output:

If `premium_10`, `income_6-10l`, `occupation_engineer` → Then `benefit_20`
Confidence: 0.48

📌 Step 6: Recommendation Logic

When a user inputs data via the **Streamlit interface**, the system:

1. Converts inputs into Apriori-compatible labels (e.g., income_<3l)
2. Matches the input feature set to all rule antecedents
3. Filters for rules where input ⊆ antecedent
4. Sorts matching rules by confidence & recommends the Benefit Term(s) with the highest confidence

Example:

IF ['premium_10', 'income_6-10l', 'occupation_engineer']

THEN ['benefit_20'] Confidence: 0.48

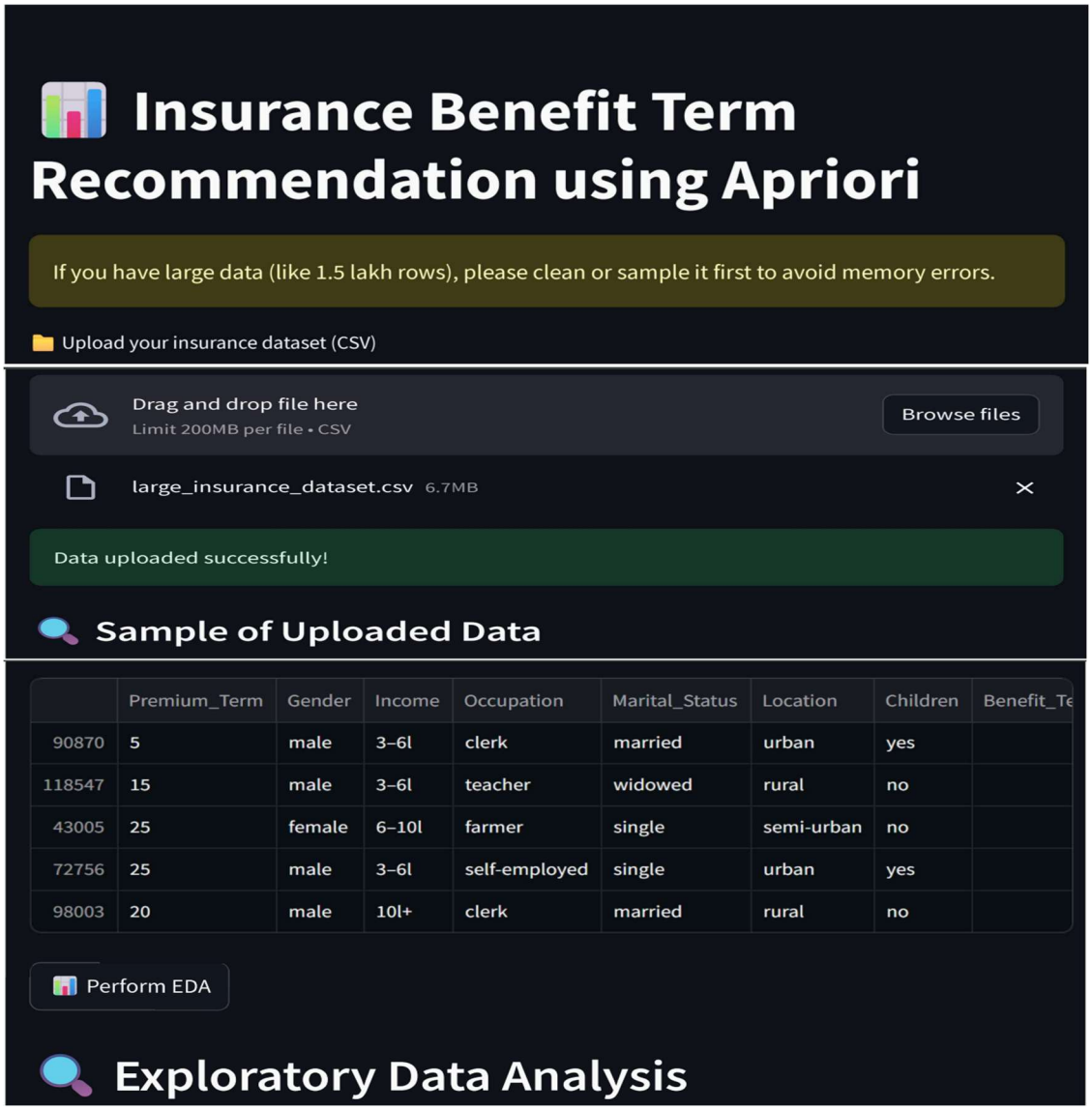
🖥️ Streamlit Interface :

To make the recommendation model interactive and accessible, a web-based interface was built using **Streamlit**, an open-source Python framework for deploying data science applications. The Streamlit GUI serves as the **frontend** of the system, allowing users (insurance agents, analysts, or customers) to enter a set of customer profile attributes and receive a personalized **Benefit Term** recommendation. Following are the **Interface Features:**

- **User Inputs**:
  - Dropdown menus for:
    Premium Term, Gender, Income, Occupation,
    Marital Status, Location, and Children
  - All options are pre-cleaned and standardized for consistency with the model

- **Real-Time Recommendation**:
    - Upon clicking "Submit", the system:
        - Converts inputs to Apriori-compatible format
        - Matches them to previously extracted rules
        - Displays the most confident matching **Benefit Term** recommendation
- **Rule Explanation**:
    - The interface also shows the **confidence level** of the selected rule
    - Users can optionally view the rule antecedents and matched logic

📸 Interface Screenshot :



📊 Results & Observations Model

- The model successfully extracted **over 500 association rules** from the dataset.
- Most common recommendations fell within the **20–30 year Benefit Term** range.
- **Higher-income professionals** (e.g., doctors, engineers) were typically linked to longer benefit terms.
- The Streamlit interface allowed for **real-time recommendations** with **explainable rule matches** & matching rules ranked by **confidence**, providing transparency in suggestions.

✅ Strengths of Model (Apriori + Streamlit)

- Uses a **realistic dataset**, enhancing practical relevance
- Fully **interpretable rule-based logic** (Apriori)
- **Web-based interface** (Streamlit) for easy access and demo
- **Tunable thresholds** (support, confidence, lift)
- Recommendations are **clear, confident, and business-aligned**

⚠️ Limitations of Model

- Still based on **synthetic data**, not live customer records
- May return **no output** for rare input combinations
- Limited to **Benefit Term** prediction only
- **Apriori is resource-heavy** for large datasets

2.6 Model 4: Apriori-Based Recommender (R Shiny)

This fourth version of the recommendation model was developed using **R Shiny**, with the goal of exploring how **association rule mining** could be applied in a different programming ecosystem while maintaining transparency and user interactivity. Here we have :

✅ replicate the Apriori-driven logic of earlier Python models using R

✅ build a fully functional **web interface** using **Shiny**

✅ allow users to upload insurance data, explore trends, and receive **policy term recommendations** based on discovered patterns

✅ demonstrate **cross-platform scalability** and extend the model's accessibility to R-based analytics teams

This model emphasizes both **explainability** and **usability**, enabling non-technical stakeholders (like agents or underwriters) to interact with data-backed policy suggestions.

Dataset Description

The application works with insurance data uploaded by the user in .csv format. The expected dataset contains customer attributes relevant for life insurance decisions, including:

Column Name	Description
`age`	Numerical customer age
`gender`	Male / Female
`declared_income`	Annual income in INR
`premium_term`	Number of years the premium is paid
`policy_term`	Number of years the policy provides coverage
`booking_frequency`	Payment frequency: Monthly, Quarterly, etc.
`location`	Urban, Semi-urban, Rural
`policy_type`	Product category (e.g., ULIP, Term Plan)
`current_status`	Marital status: Married, Unmarried, Widow

Derived Features (inside app):

- `age_group`: Categorized as 18–30, 31–40, etc. using `cut()`
- `income_group`: Binned income groups like 0–3L, 3–6L using `case_when()`

These derived attributes are essential for converting the dataset into **transactional format** suitable for rule mining.

Methodology

The methodology divided into **4 logical stages**, all implemented within a single R Shiny script:

A. Data Upload & Preprocessing

- Users upload an insurance CSV file via the interface.
- The app performs:
  - **Column validation** (ensures required fields exist)
  - **Data transformation** into labeled strings like income=6-10L, gender=male
- New features (age_group, income_group) are derived to allow categorical rule mining.

B. Transaction Creation & Apriori Rule Mining

- The processed dataset is converted to a **transaction object** using the `arules` package:

```
transactions <- as(split(item_list, seq(nrow(filtered_data))), "transactions")
```

- The **Apriori algorithm** is applied with:
  - Minimum support: **0.05**
  - Minimum confidence: **0.6**
- Rules are filtered so that:
  - The **antecedent (LHS)** contains `premium_term`
  - The **consequent (RHS)** includes `policy_term` or related policy factors

C. Recommendation Logic & Rule Ranking

- Based on user input filters (from dropdowns), the app:
  - Extracts all matching rules
  - Sorts them by confidence × lift & Displays the top rules with:
    - Rule explanation
    - Confidence score
    - Lift value

If no rule matches, a fallback mechanism recommends the most frequent policy term associated with the selected premium term.

D. EDA & Summary Tab

- The app includes an **EDA dashboard** with:
  - Histograms for Age, Income, Premium Term, Gender, Location, Booking Frequency, etc.
  - Automatically generated insights like:

"Most users fall into the 31–40 age group"
"Booking frequency is highest for 'Quarterly' mode"

This enhances understanding of the uploaded dataset before triggering recommendations.

🖥️ R Shiny Interface Features

The R Shiny application consists of a well-organized interface that makes the recommendation model highly interactive and user-friendly. The layout is divided into two main functional tabs:

⭕ Input Panel (Sidebar)

Users interact with the system by selecting key attributes via **dropdown menus**:

- **Premium Term** (in years)
- **Age Group** (e.g., 18–30, 31–40)
- **Income Group** (e.g., 0–3L, 6–10L)
- **Gender** (Male, Female)
- **Location** (Urban, Semi-Urban, Rural)
- **Current Marital Status** (Married, Unmarried, Widow)
- **Booking Frequency** (Monthly, Quarterly, etc.)

An **action button** triggers the recommendation system after all selections are made.

📋 Recommendations Tab

This tab displays **real-time policy term suggestions** based on Apriori rule matching:

- 🔹 A short **summary message** stating the premium term and recommendation context
- 🔹 A **table of top matching rules**, ranked by confidence × lift, showing:
  - Full rule explanation (e.g., "premium_term=10, income=6-10L ⇒ policy_term=25")
  - Confidence level (probability of correctness)
  - Lift (rule strength vs. random chance)

If no rules match the input, a **fallback policy term** is shown — based on the most common outcome for the selected premium term.

📊 Data Summary Tab

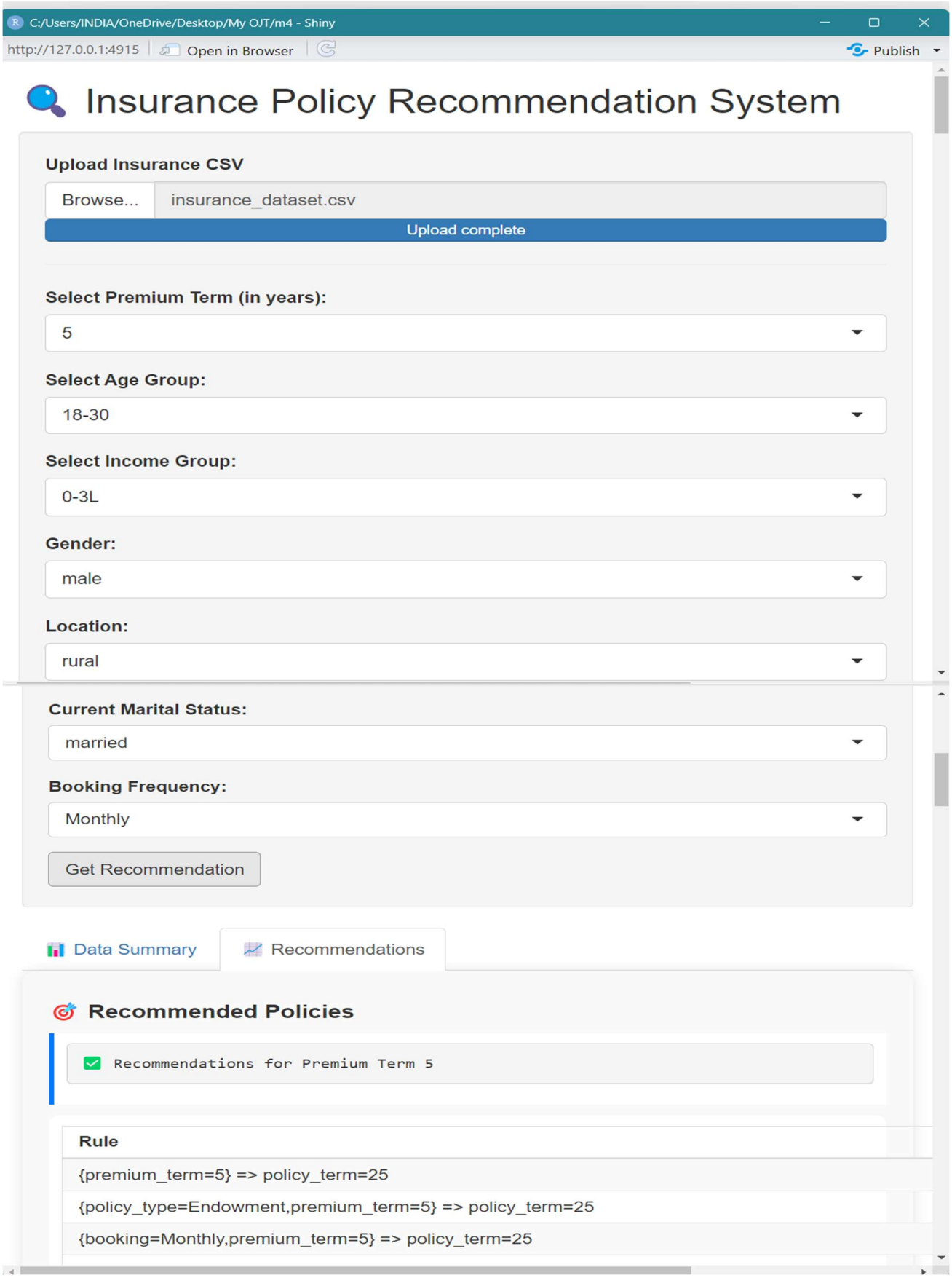
This tab provides **exploratory visualizations and auto-generated insights**:

- Histograms for Age, Declared Premium,Premium Term,Gender,Booking Frequency,Location,Marital Status .
- Text-based conclusions below each chart, such as:

“Most users belong to the 31–40 age group”
“Quarterly booking frequency is most popular”

These visuals help users understand data distribution before viewing rule-based outcomes.

Interface Screenshot :



📊 Results & Observations

- The app accurately returned **top rule-based policy terms** for most input combinations
- Fallbacks ensured no dead-ends for the user
- EDA visualizations helped in validating data trends
- Users could clearly interpret **why** a policy term was suggested (via rule table)

✅ Strengths of Model 4

- Built entirely in **R**, enabling wider accessibility for data teams familiar with the R ecosystem
- **Real-time rule filtering** with fallback support
- Embedded **EDA** visualizations for better insight
- **Confidence + Lift ranking** for interpretable recommendations
- Clean UI with tab-based layout

⚠️ Limitations of Model 4

- Requires CSV uploads — no built-in data simulation
- Fewer options for large-scale web deployment compared to Streamlit
- Static rules — needs re-running Apriori on new uploads
- Slightly more setup required for non-R users

2.8 Comparative Insights Across All Models :

To evaluate the progression and effectiveness of the four recommendation models developed during the internship, the following comparative analysis summarizes key aspects, capabilities, and trade-offs for each approach.

Key Comparative Insights

- **Model 1** served as a transparent baseline and proof of logic, but lacked adaptability or scalability.
- **Model 2** introduced Apriori mining, showing how rules can be learned from data but lacked real-world context.
- **Model 3** offered the best balance of **realistic data**, **explainability**, and **interactive interface**, making it suitable for business use.
- **Model 4** demonstrated that the same Apriori logic can be extended to **R environments**, with bonus features like fallback handling and built-in EDA.

Comparison Table :

Feature	Model 1Manual Rule-Based	Model 2Apriori + Tkinter	Model 3Apriori + Streamlit	Model 4Apriori + R Shiny
Data Source	LIC Synthetic	Synthetic (all combinations)	Uploaded CSV	Uploaded CSV
Algorithm Used	Hardcoded If–Else Logic	Apriori (Python)	Apriori (Python)	Apriori (R)
UI Framework	Streamlit	Tkinter	Streamlit	R Shiny
Rule Discovery	Manual	Automated (Apriori)	Automated (Apriori)	Automated (Apriori)
Target Variable	Policy Name	Benefit Term	Benefit Term	Policy Term
Explainable Logic	✅ Transparent	✅ Rule-based	✅ Rule-based	✅ Rule-based
Cold Start Handling	❌ None	❌ None	❌ None	✅ Fallback Rule Added
User Interaction	Basic	Moderate	Interactive Web UI	Rich UI with Tabs
EDA Support	❌ None	❌ None	✅ Yes	✅ Yes
Best Use Case	Prototype Demonstration	Rule Mining Proof of Concept	Realistic Insurance Application	R-based Deployments & Analysts

2.9 Overall Results & Observations :

After implementing four progressively advanced recommendation models, several key results and patterns were observed across the project:

General Observations

- Across all models, the most commonly recommended **Benefit Terms** or **Policy Terms** fell between **20 to 30 years**, particularly for middle-income and younger age groups.
- **Higher-income customers** and professionals such as **engineers and doctors** were more frequently associated with longer coverage durations.
- **Shorter premium terms** (10–15 years) were often linked to longer benefit durations, indicating a preference for minimized payment periods with extended protection.

Model-Specific Output Trends

- **Model 1 (Manual Rule-Based)**
  Offered fast outputs but lacked flexibility. Recommendations were static and could not adapt to new input combinations.
- **Model 2 (Apriori + Tkinter)**
  Generated over **500+ frequent itemsets**, enabling automated rule-based logic. However, due to being built on fully synthetic combinations, some rules were impractical or redundant.
- **Model 3 (Apriori + Streamlit)**
  Using a structured synthetic dataset modeled on real-world logic, the system returned **hundreds of filtered rules** with high confidence and lift. Streamlit allowed real-time interaction and clear explanation of rule matches.
- **Model 4 (Apriori + R Shiny)**
  Delivered real-time suggestions with fallback logic when no rule matched. The tab-based UI and built-in EDA charts helped validate both the input data and rule strength visually.

Practical Takeaways

- Models using the **Apriori algorithm** significantly outperformed static logic by offering **data-driven, explainable rules**.
- **Lift and confidence** provided an effective way to rank and interpret recommendations.
- Web-based frameworks like **Streamlit** and **R Shiny** offered better accessibility and user interaction than desktop GUIs.
- **Fallback logic**, introduced in Model 4, improved reliability by ensuring recommendations even when rule coverage was low.

2.10 Overall Limitations :

While the project successfully demonstrated various approaches to building a rule-based insurance recommendation system, several overarching limitations were identified across all models:

⚠️ 1. Use of Synthetic Data

All models were developed using synthetic datasets. Although efforts were made to simulate realistic customer profiles and insurance logic, real-world customer behavior, market volatility, and product-specific constraints could not be fully captured.

⚠️ 2. Cold Start Problem

For rare or unusual combinations of customer features, the Apriori-based systems often returned **no matching rule**, resulting in no recommendation (except Model 4, which added a fallback). This limitation reduces reliability for edge-case users.

⚠️ 3. Static Rule Framework

Once rules are generated using Apriori, they remain **static** until the algorithm is re-run on updated data. This limits adaptability in a dynamic product environment where customer behavior or offerings frequently change.

⚠️ 4. Limited Output Scope

All models focused solely on predicting the **Benefit Term** or **Policy Term**. A full-fledged recommender in practice would ideally recommend:

- Policy Type
- Sum Assured
- Riders or Add-ons
- Frequency of Premium Payments

⚠️ 5. Performance & Scalability

For large datasets (especially in Models 2 & 3), Apriori becomes memory-intensive. Without dimensionality reduction or rule pruning, performance slows and memory usage spikes — making real-time usage difficult in high-volume settings.

⚠️ 6. No Learning from Feedback

The recommendation system does not adapt or improve over time based on user actions. It lacks reinforcement mechanisms or active learning pipelines, which would make the model smarter with continued use.

These limitations represent opportunities for future enhancement and refinement of the system in real-world deployment scenarios.

Challenges Faced During Internship & Model Building

The process of building a rule-based insurance recommendation system, especially as part of an industry internship, came with multiple challenges—both technical and conceptual. These challenges spanned across data limitations, algorithm complexity, model implementation, and UI development.

These challenges, while difficult at first, became the **foundation for major learning** throughout the project and helped shape the structure, depth, and technical strength of the final recommendation system.

🔸 1. Unavailability of Real Insurance Data

At the beginning of the internship, one of the biggest obstacles was the **lack of access to real-life insurance data** due to confidentiality and privacy laws. Despite efforts like sending data requests to LIC and SBI Life Insurance (with official letters from the mentor), we were informed that data could not be shared even for academic purposes.
➡️ This led to the need for simulating **synthetic datasets** using Python libraries (faker, random) and business rules derived from official websites and insurance brochures.

🔸 2. Uneven or Imbalanced Data Distributions

During the initial stages of data generation, some variables (e.g., age, number of dependents, income groups) had **flat or overly uniform distributions**.
➡️ This affected the performance of the Apriori algorithm and the meaningfulness of generated rules. To fix this, we modified the random generation logic to reflect **realistic product mix behavior**, such as:

- Younger customers preferring term plans
- Parents selecting child education policies
- Higher-income customers choosing ULIPs

🔸 3. High Memory Usage with Apriori

As we started using **Apriori** with large datasets (e.g., 150,000+ records and 7–9 categorical features), the **memory load increased exponentially** due to the need to generate all possible itemsets.
➡️ Initial model runs in Python crashed due to MemoryError. This led us to:

- Reduce the dataset to smaller samples (e.g., 25,000 rows)
- Filter and encode fewer variables
- Lower the support threshold
- Tune min_support and confidence dynamically

🔸 4. Missing or Sparse Rule Generation

In multiple cases, the Apriori algorithm failed to generate **any rule** for certain input combinations. This is because some user profiles did not meet the support threshold, even though they looked valid logically.
➡️ This resulted in messages like "⚠️ No recommendation found." We solved this by:

- Improving dataset variability
- Lowering min_support
- Implementing **fallback logic** in Model 4 (R Shiny)
- Adding exception handling in the GUI

🔸 5. Ensuring Required Columns in Uploaded Datasets

Since some models allowed **user-uploaded CSVs**, there was a risk that required columns like premium_term, policy_term, or income might be missing.
➡️ This required implementing **column validation logic** in both Streamlit and R Shiny:

- Checking if all expected columns are present
- Showing custom error messages if not
- Preventing rule mining until proper format is ensured

🔸 6. Streamlit Handling of Large Datasets

Streamlit was an intuitive and modern tool, but handling **large datasets and rule computation in real-time** caused noticeable delays or crashes during:

- Data encoding
- Frequent itemset generation
- Filtering matching rules

➡️ We introduced dataset downsizing, code optimization, and selective input filtering to improve speed and reduce lag in the web app.

🔸 7. Learning Curve with Unfamiliar Concepts

Coming from an actuarial and statistical background, I had not worked with **recommendation engines** or **association rule mining** before. Even simple mistakes like spelling "Apriori" incorrectly delayed the early learning phase.
➡️ Through research, practice, and mentorship support, I was able to:

- Understand rule mining logic
- Differentiate between LHS (antecedent) and RHS (consequent)
- Learn how support, confidence, and lift impact recommendations

Experience & Learnings

During my one-month On-the-Job Training (OJT) at Bajaj Allianz Life Insurance, I had the opportunity to work on a complete end-to-end insurance recommendation system. The project enabled me to bridge the gap between academic learning and real-world data application, combining theoretical concepts from statistics and actuarial science with business intelligence tools, Python/R coding, and insurance domain knowledge.

1. Technical Learnings

- Gained hands-on experience in **data generation** by creating realistic synthetic datasets using Python. This included simulating multiple LIC and life insurance product features such as age group, income slab, premium term, policy type, etc.
- Implemented **rule-based logic** using if-else statements and manual decision tables in early prototype models.
- Learned to apply the **Apriori algorithm** using both Python (mlxtend) and R (arules) for extracting association rules from large-scale insurance data.
- Developed interactive, user-friendly GUIs using both **Streamlit** (for web apps) and **R Shiny**, enabling real-time recommendation delivery.
- Tackled **memory optimization challenges** due to high-dimensional data and learned how to handle them using:
    - Sampling techniques (e.g., 25,000 records from 1.5L rows)
    - Threshold tuning (support and confidence), dataset filtering and encoding

2. Conceptual Understanding

- Deepened my understanding of the **life insurance domain**, including product mechanics like:
    - Term insurance vs. Endowment plans
    - Premium vs. benefit terms
    - Riders and frequency options
- Learned how **customer demographics** influence insurance preferences.
- Understood the practical significance of **association rule mining**, including how support, confidence, and lift help uncover **hidden decision patterns** in customer data.

3. Professional & Analytical Growth

- Improved my skills in **data interpretation**, pattern identification, and translation of rules into actionable business insights.
- Strengthened my **report writing** and **technical documentation** skills by explaining complex systems in a clear, structured way.
- Gained confidence in working with ambiguity—especially when facing challenges like no rule matches, missing data, or algorithm errors.
- Collaborated with fellow interns from different academic backgrounds, enhancing my communication and leadership abilities.

🔹 4. Overall Internship Experience

This internship provided a **rich learning opportunity** that allowed me to:

- Apply classroom knowledge in statistics, actuarial science, and analytics in a real business context.
- Understand how recommendation systems are implemented in the insurance industry.
- Combine business domain understanding with technical tools like Python, R, and Shiny to deliver functional solutions.

🎯 **This experience has not only enhanced my technical foundation but also prepared me to handle real-world data challenges in insurance analytics, machine learning, or actuarial practice.**

Conclusion

This internship project at **Bajaj Allianz Life Insurance** provided an enriching opportunity to apply academic knowledge of statistics and data analysis in a real-world business setting. The core objective of the project—to develop a rule-based recommendation engine for life insurance products—was successfully achieved through a structured, iterative approach.

Throughout the course of the internship, I explored multiple modeling strategies, progressing from manual decision-rule systems to fully automated recommendation engines using the **Apriori algorithm**. These were implemented using both **Python** (Streamlit, Tkinter) and **R** (Shiny), offering a wide spectrum of deployment formats and user interfaces.

Despite challenges like the **unavailability of real insurance data**, high memory consumption, and the initial learning curve associated with **association rule mining**, the internship delivered meaningful outcomes:

- A realistic synthetic insurance dataset was created using domain-based logic.
- Multiple models were tested and compared to understand performance trade-offs.
- Functional GUIs were built to simulate real-time customer recommendations based on demographic and financial profiles.

The project not only deepened my technical understanding of recommendation systems, EDA, and rule mining, but also improved my grasp of insurance domain logic and user experience design.

This OJT experience was a key step forward in my academic and professional journey—helping me grow into a more confident, problem-solving data analyst who can bridge the gap between **statistical theory** and **business insight**.

References

This project was guided by a blend of academic knowledge, practical tools, and domain-based research. The following references were instrumental in building, validating, and presenting the recommendation system models:

- **A. Python Libraries & Tools**

  - **numpy,pandas**: Numerical operations & data manipulation, cleaning – https://pandas.pydata.org & https://numpy.org
  - **mlxtend**: Frequent itemset and Apriori implementation – https://rasbt.github.io/mlxtend/
  - **matplotlib** & **seaborn**: Data visualization libraries – https://matplotlib.org, https://seaborn.pydata.org
  - **streamlit**: Web application interface for Python – https://streamlit.io

- **B. R Libraries**

  - **shiny**: Web-based reactive application framework – https://shiny.rstudio.com
  - **arules**: Association rule mining and Apriori algorithm in R – https://cran.r-project.org/web/packages/arules

- **C. Domain Sources , Academic & Technical Concepts**

  - **Bajaj Allianz Life Insurance**: Product features, brochures, and policy information – https://www.bajajallianzlife.com
  - **LIC India**: For synthetic LIC dataset modeling – https://www.licindia.in
  - **IRDAI** Annual Reports and statistics – For claim settlement ratio benchmarks
  - **Association Rule Mining – Apriori Algorithm**
    Article: GeeksforGeeks – Association Rules
    Overview of itemset generation, support, confidence, and lift.
  - **IJISRT Research Paper: "Association Rule Based Recommender System for Life Insurance"**
    [Ref 1] – This paper presents a foundation on building recommendation systems using Apriori in the insurance domain.
    Source: International Journal of Innovative Science and Research Technology, March 2023.
  - **"Recommending Insurance Products Using User Sentiments"** – Amity University
    [Ref 2] – Combines sentiment analysis and customer profile data to improve insurance product recommendations using ML models like Logistic Regression and Random Forest.
    Keywords: Sentiment Analysis, Recommender Systems, Random Forest, Customer Feedback.

- **D. AI-Powered Learning Assistance**

  - **ChatGPT (OpenAI) :** Assisted with research synthesis, code troubleshooting, interpreting EDA outputs, and structuring the report.

Deliverables

All files, models, code, visuals, and documentation developed during the internship have been structured and submitted in a unified repository. You may access the complete submission in either of the following formats:

- 🗂️ **Local Archive**: OJT.zip (shared offline or via email/drive)
- 🌐 **GitHub Repository (Ctrl+click to follow lonk) :** https://github.com/RajshriDPatil/OJT_Bajaj_Allianz_Life

📁 Folder Structure & Repository Overview

Folder Name	Description
1. Data & EDA	Contains the generated synthetic insurance datasets, data preprocessing, and exploratory data analysis notebooks (`.ipynb`)
2. Model 1	Manual rule-based model using if-else logic (Python + LIC dataset + Streamlit interface)
3. Model 2	Apriori-based recommendation using Tkinter GUI (`tkinter_policy_recommender.py`)
4. Model 4	Apriori-based recommendation using Streamlit (`apriori_update.py`) and filtered dataset
5. Model 4	R Shiny app (`app.R`) with histogram visualizations and fallback logic
report/	Final internship report: `My OJT Report.docx`
references/	Academic research papers and supporting notes (`ref 1.pdf`, `ref 2.pdf`)

📂 File Formats Included

- **.csv** – Datasets
- **.ipynb**, **.py**, **.R** – Notebooks and model scripts
- **.docx** – Final report document
- **.pdf** – Research references
- **.png** – GUI screenshots and visuals (if added)

📎 Submission Summary

All deliverables are included as:

- 📁 A structured zipped folder OJT.zip
- 🖥️ Optional GitHub repository with the same structure
- 📄 Accompanied by a README.txt file (included in both ZIP and GitHub), documenting:
  - Project objective
  - Model structure
  - How to run each model
  - Software used and credits