Project Report on

# CUSTOMER CHURN PREDICTION

**MINI PROJECT I**

22ITC07

in

B.E, IV-SEM, INFORMATION TECHNOLOGY

Submitted by

**DEVANABOYINA CHANDANA, 160123737012**

**PODICHETTY SAMIKSHA, 160123737021**

UNDER THE GUIDANCE OF:

**Mr. V. Santhosh**

Assistant Professor, Department of Information Technology

DEPARTMENT OF INFORMATION TECHNOLOGY

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY(A)

(Affiliated to Osmania University; Accredited by NBA, NAAC, ISO)

Kokapet(V), Gandipet(M), Hyderabad

2024-2025

# CERTIFICATE

This is to certify that the project work entitled "Customer Churn Prediction" submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY,** in partial fulfilment of the requirements for the award of the completion of Mini Project-II (22ITC15) IV Semester of B.E in Information Technology, during the academic year 2024-2025, is a record of original work done by **Devanaboyina Chandana (160123737012), Podichetty Samiksha (160123737021)** during the period of study in Department of IT, CBIT, HYDERABAD, under our supervision and guidance.

**Project Guide**
**Mr.V Santhosh**
Assistant Professor, Dept. of IT,
CBIT, Hyderabad.

**Head of the Department**
**Dr.M. Venu Gopalachari**
Professor, Dept. of IT,
CBIT, Hyderabad.

# TABLE OF CONTENTS

# LIST OF FIGURES AND SCREENSHOTS

# ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who supported us throughout the course of this mini project.First and foremost, we extend our heartfelt thanks to our **Project Guide, Dr. Santhosh V**, Assistant Professor, Department of Information Technology, CBIT, Hyderabad, for his valuable guidance, consistent encouragement, and continuous support throughout the development of this project.We are also immensely grateful to **Dr. M. Venu Gopalachari**, Professor and **Head of the Department of Information Technology**, CBIT, Hyderabad, for providing us with the opportunity and the resources to carry out this project successfully. We would also like to acknowledge the support of our faculty members, lab assistants, and peers who provided timely feedback and motivation during different phases of this project. Finally, we thank our institution, **Chaitanya Bharathi Institute of Technology**, Hyderabad, for fostering an environment of learning and innovation.

# ABSTRACT

The Customer Churn Prediction System leverages machine learning techniques to identify customers at risk of leaving a business. Using a Random Forest Classifier, the system analyzes key customer attributes such as age, annual income, spending score, and purchase history to predict churn probability. It offers actionable business recommendations based on overall data trends, helping companies proactively address retention challenges. An interactive Gradio-based user interface allows users to upload customer data or provide input manually, receiving real-time churn risk predictions and strategy suggestions. Visualizations like histograms and box plots provide additional insights into customer behavior patterns. This system empowers businesses to make informed decisions, reduce customer attrition, and enhance profitability. By combining predictive analytics with an intuitive design, it serves as a practical and accessible solution for modern churn management.

# INTRODUCTION

Customer churn, the phenomenon where clients or customers discontinue their engagement with a company's products or services, represents a persistent and costly challenge across industries. As markets become increasingly competitive and customer expectations continue to rise, retaining existing customers has become more important—and more cost-effective—than acquiring new ones. Research consistently shows that customer retention has a direct and profound impact on long-term profitability, brand loyalty, and overall business sustainability. In this context, the ability to predict customer churn in advance allows businesses to act proactively rather than reactively, giving them a strategic advantage.

This project introduces a robust and intelligent Customer Churn Prediction System that leverages machine learning to address this issue with precision and efficiency. By harnessing historical customer data, the system aims to identify individuals who are at risk of leaving, enabling companies to take timely and targeted measures to retain them. The predictive model is built using a Random Forest Classifier, which is well-suited for classification tasks due to its ensemble learning structure and ability to handle complex, non-linear relationships in data. The model processes key customer attributes such as age, annual income, spending score, and purchase history to assess churn likelihood with high accuracy.

To enhance usability and accessibility, the system features an intuitive frontend developed with Gradio. This interface allows users to enter individual customer information or upload bulk data through CSV files, generating real-time churn predictions along with practical business recommendations. Beyond predictive functionality, the system also serves as an analytical tool—providing visual insights and patterns that can inform retention strategies and optimize marketing campaigns. With its focus on usability, interpretability, and impact, this project offers a scalable solution that empowers businesses to make data-driven decisions, reduce churn rates, and foster long-term customer relationships.

## 1.1 PROBLEM STATEMENT/OBJECTIVES

- To develop a machine learning model (Random Forest Classifier) that predicts customer churn accurately.
- To analyze customer attributes such as age, annual income, spending score, and purchase history for churn prediction.
- To provide real-time churn risk assessments using an interactive Gradio user interface.
- To generate actionable business strategy recommendations based on overall customer behavior trends.
- To visualize customer data through dynamic charts and graphs for better interpretation of churn patterns.
- To create an easy-to-use system accessible even to non-technical users for decision-making support.
- To help businesses improve customer retention and profitability through data-driven insights.

## 1.2 ORGANIZATION OF THE REPORT

The report is organized as follows:

- **Chapter 1: Introduction**
  Overview of the project, its significance in the business world, and the motivation behind developing a machine learning-based churn prediction system.
- **Chapter 2: Literature Survey**
  Analysis of existing churn prediction techniques, limitations of traditional monitoring systems, and the need for proactive churn prediction solutions.
- **Chapter 3: Methodology**
  Detailed explanation of the system architecture, components involved, data preprocessing steps, model training approach, and customer interaction workflow.
- **Chapter 4: Implementation**
  Description of the technologies used (Python, scikit-learn, pandas, seaborn, matplotlib, Gradio) and implementation of various modules like churn prediction, data visualization, and business strategy generation.
- **Chapter 5: Testing and Results**
  Explanation of the testing process, evaluation metrics used, visualizations created, and insights derived from model predictions and analysis.
- **Chapter 6: Conclusion and Future Scope**
  Summary of the project's outcomes, key findings, identified limitations,

and potential enhancements for expanding the system's capabilities in the future.

- **References**
  List of all libraries, documentation, and other resources consulted during the research, development, and testing phases of the project.

# 2. LITERATURE SURVEY

Customer churn prediction has been a critical area of research in the fields of business analytics and machine learning. Traditional churn monitoring systems are largely reactive, identifying customer loss only after it occurs. These systems often lack personalized risk scoring and advanced predictive capabilities, limiting their effectiveness in proactive customer retention strategies.Several machine learning techniques such as Logistic Regression, Decision Trees, and Support Vector Machines have been used in earlier studies for churn prediction. However, Random Forest Classifiers have emerged as one of the most effective models due to their ability to handle large datasets, manage missing values, and prevent overfitting through ensemble learning.

Existing churn analysis tools primarily offer basic dashboards with historical insights but do not provide real-time, customer-specific risk assessments or actionable business recommendations. This gap highlights the need for a system that combines predictive modeling with business strategy generation and user-friendly visualization.Our proposed system addresses these limitations by implementing a Random Forest-based model that not only predicts churn but also provides data-driven strategic suggestions. It integrates an interactive Gradio user interface to ensure accessibility even for non-technical users, offering a comprehensive and proactive solution to customer retention challenges. Customer churn prediction has become an essential focus for businesses aiming to sustain growth and profitability. Traditional churn detection systems primarily rely on historical analysis and reactive reporting, often identifying churn after the customer has already left. These systems typically lack predictive capabilities and offer minimal personalization, making targeted retention efforts challenging. Recent advancements in machine learning have introduced models such as Logistic Regression, Decision Trees, and Support Vector Machines for churn prediction. However, Random Forest Classifiers have shown superior performance due to their robustness, accuracy, and ability to handle noisy and incomplete data.

# 3. METHODOLOGY

## 3.1 ARCHITECTURE

**Type:** Website

**Components:**

1. **Frontend (Client-side):**
   • Built using Gradio (Python-based interactive web UI)
   • Runs on a web browser
   • Accepts user inputs (CSV upload or manual sliders) and displays prediction results dynamically
2. **Backend (Server-side):**
   • Built using Python with scikit-learn and pandas
   • Handles data preprocessing, churn prediction, business strategy generation, and visualization
3. **Data Storage:**
   • Customer data temporarily processed in memory (pandas DataFrame)
   • No permanent database integration (for simplicity and portability)

## 3.2 ASSUMPTIONS

• Users interact with the application through a modern web browser.
• Uploaded CSV files are properly formatted with required columns (Age, Annual Income, Spending Score, Purchase History, Churn).
• Predictions are based on current customer attributes only, without historical behavior trends.
• All machine learning operations (training and prediction) are done locally during runtime.
• No sensitive customer authentication or login system is implemented (open public use assumed).
• Real-time data visualizations are generated on demand without requiring page refresh.

## 3.3 REQUIREMENTS

### 3.3.1 Functional Requirements

• Upload customer dataset via CSV file
• Manually input customer details using sliders
• Predict churn risk (High/Low) and show churn probability percentage
• Provide business strategy recommendations based on dataset trends
• Display dynamic visualizations (histograms, box plots, churn distribution)
• Show data summary statistics

### 3.3.2 Non-Functional Requirements

• Intuitive and simple user interface
• Responsive layout accessible via any modern browser
• Lightweight design suitable for cloud-based and local deployment
• Minimal hardware requirements (works with 4GB RAM systems)

## 3.4 TERMS USED

• **Churn:** When a customer stops doing business with the company.
• **Churn Risk:** The predicted likelihood that a customer will leave.
• **Spending Score:** A metric indicating a customer's purchasing behavior and loyalty.
• **Purchase History:** Past purchase frequency used to predict future behavior.
• **Random Forest Classifier:** A machine learning algorithm that uses multiple decision trees to improve prediction accuracy.

# 4. IMPLEMENTATION

The Customer Churn Prediction System is implemented using Python with libraries like pandas, numpy, scikit-learn, seaborn, matplotlib, and Gradio. The backend handles data processing, machine learning, and visualization, while the frontend uses Gradio to offer an intuitive web-based interface for user interaction.Initially, the customer dataset is uploaded through a CSV file or provided via slider inputs. The system preprocesses the data by converting dates, handling missing values, and preparing features for model training. The Random Forest Classifier is used to train the model on customer attributes like Age, Annual Income, Spending Score, and Purchase History, with Churn as the target variable.Alongside predictions, the system also generates business strategy recommendations based on overall customer behavior trends, such as launching retention campaigns or adjusting pricing strategies.Dynamic data visualizations, including histograms, box plots, and churn distribution charts, are generated to help users better understand customer patterns.. This implementation combines machine learning, business analytics, and web development into a single, easy-to-use system that supports proactive customer retention strategies.For real-time churn prediction, the Gradio interface enables two methods of input: CSV file upload or manual entry through sliders.

1. **Importing libraries**

```
import gradio as gr
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, r2_score
import datetime
from dateutil.relativedelta import relativedelta
```

## 2. Interface Designing

```
CSS = """
:root {
    --primary: #1a4b8c;
    --secondary: #3a7bd5;
    --accent: #FF7F50;
    --background: #e6f0fa;
    --card: #FFFFFF;
    --text: #333333;
    --font-heading: 'Montserrat', sans-serif;
}
@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@600;700&display=swap');
body, html {
    background-color: var(--background) !important;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
    height: 100%;
}
.dashboard-header {
    background: linear-gradient(135deg, var(--primary), var(--secondary));
    color: white !important;
    padding: 2rem;
    margin-bottom: 1.5rem;
    text-align: center;
    box-shadow: 0 4px 12px rgba(0,0,0,0.1);
    font-family: var(--font-heading);
    border-bottom: 4px solid var(--accent);
}
.dashboard-header h1 {
    font-size: 2.5rem;
    margin: 0;
    letter-spacing: 1px;
    text-shadow: 1px 1px 3px rgba(0,0,0,0.2);
}
```

```
.dashboard-header h1 {
    font-size: 2.5rem;
    margin: 0;
    letter-spacing: 1px;
    text-shadow: 1px 1px 3px rgba(0,0,0,0.2);
}
.dashboard-header p {
    font-size: 1.2rem;
    margin: 0.5rem 0 0;
    opacity: 0.9;
}
.card {
    background: var(--card);
    border-radius: 10px;
    padding: 1.5rem;
    margin-bottom: 1.5rem;
    box-shadow: 0 4px 12px rgba(0,0,0,0.08);
    border: none;
    transition: transform 0.2s, box-shadow 0.2s;
}
.card:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 16px rgba(0,0,0,0.12);
}
.alert-success {
    color: #28A745;
    background-color: #E8F5E9;
    border-left: 4px solid #28A745;
    padding: 1rem;
    border-radius: 4px;
}
.alert-error {
    color: #DC3545;
    background-color: #F8E8E8;
    border-left: 4px solid #DC3545;
```

## 3. Data Processing

```
def preprocess_data(file_obj):
    global df, models, feature_importance

    try:
        if file_obj is None:
            return "<div class='alert-error'>❌ Please upload a file first</div>", None

        # Reset previous data and models
        df = None
        models = {}
        feature_importance = {}

        df = pd.read_csv(file_obj.name)

        # Check required columns (removed Date from required columns)
        required_cols = ['CustomerID', 'Age', 'Annual_Income', 'Spending_Score', 'Purchase_History']
        if not all(col in df.columns for col in required_cols):
            missing = [col for col in required_cols if col not in df.columns]
            return f"<div class='alert-error'>❌ Missing columns: {', '.join(missing)}</div>", None

        # Add Date column if not present (using current date)
        if 'Date' not in df.columns:
            df['Date'] = datetime.datetime.now().strftime('%Y-%m-%d')

        # Convert currency to ₹ (Indian Rupees)
        df['Annual_Income_INR'] = df['Annual_Income'] * 75

        # Feature engineering
        df['Date'] = pd.to_datetime(df['Date'])
        df['Days_Since_First_Purchase'] = (df['Date'] - df['Date'].min()).dt.days
        df['Purchase_Frequency'] = df['Purchase_History'] / df['Days_Since_First_Purchase'].replace(0, 1)
        df['Customer_Lifetime_Value'] = (df['Annual_Income_INR'] * 0.1) + (df['Purchase_History'] * 500)
```

```
        # Generate summary stats
        stats = {
            "Total Customers": f"{len(df):,}",
            "Average Age": f"{df['Age'].mean():.1f} years",
            "Average Income (₹)": f"₹{df['Annual_Income_INR'].mean():,.0f}",
            "Avg. Spending Score": f"{df['Spending_Score'].mean():.1f}/100",
            "Total Purchases": f"{df['Purchase_History'].sum():,}",
            "Data Coverage": f"{df['Date'].min().strftime('%b %Y')} to {df['Date'].max().strftime('%b %Y')}",
            "Avg. Customer Value": f"₹{df['Customer_Lifetime_Value'].mean():,.0f}"
        }

        stats_html = "<div class='card'><h4 class='section-title'>📊 Dataset Summary</h4><ul style='columns: 2;'>"
        for k, v in stats.items():
            stats_html += f"<li style='margin-bottom: 0.5rem;'><strong>{k}:</strong> {v}</li>"
        stats_html += "</ul></div>"

        return "<div class='alert-success'>✅ Data processed successfully! ML can now analyze patterns.</div>", stats_html
    except Exception as e:
        return f"<div class='alert-error'>❌ Error: {str(e)}</div>", None
```

## 4. Model Training

```
def train_models():
    global df, models, feature_importance

    if df is None:
        return "<div class='alert-error'>❌ No data available. Please upload and process data first.</div>", None, None

    try:
        # Prepare features with ML explanations
        features = ['Age', 'Annual_Income', 'Spending_Score', 'Purchase_History',
                    'Days_Since_First_Purchase', 'Purchase_Frequency']

        X = df[features]
        y_churn = (df['Spending_Score'] < 40).astype(int)  # Churn = low spending score
        y_sales = df['Purchase_History']
        y_clv = df['Customer_Lifetime_Value']

        # Train-test split
        X_train, X_test, y_train, y_test = train_test_split(X, y_churn, test_size=0.3, random_state=42)

        # Train models with ML explanations
        churn_model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='balanced')
        sales_model = RandomForestRegressor(n_estimators=100, random_state=42)
        clv_model = RandomForestRegressor(n_estimators=100, random_state=42)

        churn_model.fit(X_train, y_train)
        sales_model.fit(X_train, y_sales[X_train.index])
        clv_model.fit(X_train, y_clv[X_train.index])

        # Store models and metrics
        models = {
            'churn': churn_model,
            'sales': sales_model,
            'clv': clv_model,
            'metrics': {
```

```
    # Model cards with ML explanations
    model_cards = f"""
    <div class='card'>
        <h4 class='section-title'>🤖 Model Performance</h4>
        <div style="display: grid; grid-template-columns: 1fr 1fr 1fr; gap: 1rem;">
            <div>
                <h5>Churn Model</h5>
                <p><i>ML predicts customers likely to stop purchasing based on spending patterns</i></p>
                <p><strong>Accuracy:</strong> {models['metrics']['churn_accuracy']:.2f}</p>
                <p><strong>AUC:</strong> {models['metrics']['churn_auc']:.2f}</p>
            </div>
            <div>
                <h5>Sales Model</h5>
                <p><i>ML forecasts future purchase volumes using customer behavior patterns</i></p>
                <p><strong>R²:</strong> {models['metrics']['sales_r2']:.2f}</p>
            </div>
            <div>
                <h5>CLV Model</h5>
                <p><i>ML estimates long-term customer value from income and purchase history</i></p>
                <p><strong>R²:</strong> {models['metrics']['clv_r2']:.2f}</p>
            </div>
        </div>
    </div>
    """

    # Generate feature importance plot with ML explanation
    feature_fig = plot_feature_importance()

    return "<div class='alert-success'>✅ Models trained successfully!</div>", model_cards, feature_fig
except Exception as e:
    return f"<div class='alert-error'>❌ Training error: {str(e)}</div>", None, None
```

```
try:
    features = ['Age', 'Income', 'Spending', 'Purchases', 'Days', 'Frequency']

    fig = go.Figure()
    fig.add_trace(go.Bar(
        x=features,
        y=feature_importance['churn'],
        name='Churn Importance',
        marker_color='#1a4b8c',
        hovertemplate='<b>%{x}</b><br>Importance: %{y:.2f}<extra></extra>'
    ))
    fig.add_trace(go.Bar(
        x=features,
        y=feature_importance['sales'],
        name='Sales Importance',
        marker_color='#3a7bd5',
        hovertemplate='<b>%{x}</b><br>Importance: %{y:.2f}<extra></extra>'
    ))
    fig.add_trace(go.Bar(
        x=features,
        y=feature_importance['clv'],
        name='CLV Importance',
        marker_color='#5ca0ff',
        hovertemplate='<b>%{x}</b><br>Importance: %{y:.2f}<extra></extra>'
    ))
    fig.update_layout(
        title='<b>Feature Importance</b><br><i>ML quantifies how much each factor influences predictions</i>',
        barmode='group',
        plot_bgcolor='white',
        paper_bgcolor='white',
        height=400,
        legend=dict(orientation="h", yanchor="bottom", y=1.02, xanchor="right", x=1)
    )
    return fig
except Exception as e:
```

## 5.Analytics

```
def generate_sales_analytics():
    global df
    if df is None:
        return "<div class='alert-error'>Please upload and process data
first</div>"

    try:
        # Ensure Date column exists
        if 'Date' not in df.columns:
            df['Date'] = datetime.datetime.now()

        # Sales statistics
        monthly_sales =
df.set_index('Date')['Purchase_History'].resample('M').sum()
        total_sales = monthly_sales.sum()
        avg_monthly = monthly_sales.mean()
        growth_rate = (monthly_sales.iloc[-1] - monthly_sales.iloc[0])
/ monthly_sales.iloc[0] * 100 if len(monthly_sales) > 1 else 0

        stats = {
            "Total Sales": f"{total_sales:,.0f}",
            "Average Monthly Sales": f"{avg_monthly:,.0f}",
            "Growth Rate": f"{growth_rate:.1f}%",
            "Best Month": f"{monthly_sales.idxmax().strftime('%b %Y')}
({monthly_sales.max():,.0f})" if len(monthly_sales) > 0 else "N/A",
            "Worst Month": f"{monthly_sales.idxmin().strftime('%b %Y')}
({monthly_sales.min():,.0f})" if len(monthly_sales) > 0 else "N/A"
        }

        stats_html = "<div class='card'><h4 class='section-title'>📊
Sales Statistics</h4><ul style='columns: 2;'>"
        for k, v in stats.items():
            stats_html += f"<li style='margin-bottom:
0.5rem;'><strong>{k}:</strong> {v}</li>"
        stats_html += "</ul></div>"

        # Monthly trend
        monthly_df = monthly_sales.reset_index()
        monthly_df['Rolling_Avg'] =
monthly_df['Purchase_History'].rolling(3, min_periods=1).mean()

        trend_fig = px.line(
            monthly_df,
            x='Date',
            y=['Purchase_History', 'Rolling_Avg'],
            title='<b>Monthly Sales Trend</b>',
            color_discrete_sequence=['#1a4b8c', '#3a7bd5'],
```

```
            height=350
        )
        trend_fig.update_layout(
            plot_bgcolor='white',
            paper_bgcolor='white',
            legend_title="Metric"
        )

        analytics_html = f"""
        <div class='card'>
            <h4 class='section-title'>📈 Sales Analytics</h4>
            {stats_html}
            <div style="margin-top: 1rem;">
                {trend_fig.to_html(full_html=False,
include_plotlyjs='cdn')}
            </div>
        </div>
        """

        return analytics_html
    except Exception as e:
        return f"<div class='alert-error'>Sales analytics error:


{str(e)}</div>"
```

## 6. Planning

```
def generate_production_plan():
    global df
    if df is None:
        return "<div class='alert-error'>Please upload and process data first</div>"

    try:
        # Ensure Date column exists
        if 'Date' not in df.columns:
            df['Date'] = datetime.datetime.now()

        monthly_sales = df.set_index('Date')['Purchase_History'].resample('M').sum()

        # ML-enhanced seasonal factors
        seasonal_factor = {
            1: 1.3, 2: 1.1, 3: 1.0, 4: 0.9, 5: 0.95, 6: 0.85,
            7: 0.8, 8: 0.9, 9: 1.0, 10: 1.1, 11: 1.4, 12: 1.5
        }

        # Generate 12-month plan
        current_date = datetime.datetime.now()
        months = [current_date + relativedelta(months=i) for i in range(12)]
        plan = []

        for month in months:
            month_name = month.strftime('%b %Y')
            factor = seasonal_factor.get(month.month, 1.0)
            predicted_sales = monthly_sales.mean() * factor if len(monthly_sales) > 0 else 0

            if month.month in [11, 12, 1]:
                plan.append({
                    "month": month_name,
                    "action": "Increase +30%",
                    "reason": "Peak season (ML predicts ↑ sales)",
```

```
                plan.append({
                    "month": month_name,
                    "action": "Reduce -20%",
                    "reason": "Slow season (ML predicts ↓ sales)",
                    "prediction": f"{predicted_sales:,.0f} (↓{100-factor*100:.0f}%)"
                })
            else:
                plan.append({
                    "month": month_name,
                    "action": "Standard",
                    "reason": "Typical sales period",
                    "prediction": f"{predicted_sales:,.0f}"
                })

        # Create sales trend visualization
        sales_fig = px.line(
            monthly_sales.reset_index(),
            x='Date',
            y='Purchase_History',
            title='<b>Historical Sales</b><br><i>ML detects trends for inventory optimization</i>',
            color_discrete_sequence=['#1a4b8c'],
            height=300
        )
        sales_fig.update_layout(
            plot_bgcolor='white',
            paper_bgcolor='white'
        )

        # Create plan table
        plan_html = "<table style='width: 100%; border-collapse: collapse;'>"
        plan_html += """
        <tr style='background-color: #1a4b8c; color: white;'>
            <th style='padding: 10px; text-align: left;'>Month</th>
            <th style='padding: 10px; text-align: left;'>Action</th>
            <th style='padding: 10px; text-align: left;'>ML Insight</th>
        </tr>
        """
        for i, item in enumerate(plan):
            bg_color = '#f0f8ff' if i % 2 == 0 else 'white'
            icon = "📈" if "Increase" in item["action"] else "📉" if "Reduce" in item["action"] else "📊"
            plan_html += f"""
            <tr style='background-color: {bg_color};'>
                <td style='padding: 10px; border-bottom: 1px solid #ddd;'>{icon} {item['month']}</td>
                <td style='padding: 10px; border-bottom: 1px solid #ddd;'><strong>{item['action']}</strong></td>
                <td style='padding: 10px; border-bottom: 1px solid #ddd;'>{item['reason']}</td>
                <td style='padding: 10px; border-bottom: 1px solid #ddd; text-align: right;'>{item['prediction']}</td>
            </tr>
            """
        plan_html += "</table>"

        return f"""
        <div class='card'>
            <h4 class='section-title'>📅 12-Month Production Plan</h4>
            <div style="display: grid; grid-template-columns: 1fr 1fr; gap: 1rem; margin-bottom: 1rem;">
                <div>
                    <h5>ML-Generated Recommendations</h5>
                    {plan_html}
                </div>
                <div>
                    {sales_fig.to_html(full_html=False, include_plotlyjs='cdn')}
                </div>
            </div>
        </div>
        """
    except Exception as e:
        return f"<div class='alert-error'>Production plan error: {str(e)}</div>"
```

## 7. Gradio

```
with gr.Blocks(css=CSS, theme=gr.themes.Default()) as app:
    with gr.Column():
        # Header
        gr.Markdown("""
        <div class='dashboard-header'>
            <h1>Business and Churn Analysis</h1>
            <p>AI-powered customer insights for strategic decisions</p>
        </div>
        """)

        with gr.Tabs():
            # Data & Modeling
            with gr.Tab("🗁 Data & Modeling"):
                with gr.Row():
                    with gr.Column(scale=2):
                        gr.Markdown("### 1. Upload and Prepare Data")
                        file_input = gr.File(label="Upload Customer
Data (CSV)", file_types=[".csv"])
                        with gr.Row():
                            preprocess_btn = gr.Button("Process Data",
variant="primary")
                        preprocess_status = gr.HTML()
                        data_summary = gr.HTML()

                    with gr.Column(scale=3):
                        gr.Markdown("### 2. Train ML Models")
                        with gr.Row():
                            train_btn = gr.Button("Train Models",
variant="primary")
                        train_status = gr.HTML()
                        model_cards = gr.HTML()

                        gr.Markdown("### 3. Model Insights")
                        feature_plot = gr.Plot()

            # Analytics
            with gr.Tab("🗹 Analytics"):
                sales_analytics = gr.HTML()

            # Predictions
            with gr.Tab("🧠 Predictions"):
                with gr.Row():
                    with gr.Column(scale=1):
                        gr.Markdown("### Customer Profile")
                        age = gr.Slider(18, 100, value=35, label="Age",
step=1)
```

```
                        income = gr.Slider(0, 5000000, value=1500000,
label="Annual Income (₹)", step=10000)
                        spending_score = gr.Slider(0, 100, value=65,
label="Spending Score (0-100)", step=1)
                        purchase_history = gr.Slider(0, 1000,
value=120, label="Past Purchases", step=1)
                        predict_btn = gr.Button("Analyze Customer",
variant="primary")

                    with gr.Column(scale=2):
                        prediction_output = gr.HTML()
                        with gr.Row():
                            churn_gauge = gr.Plot()
                            sales_gauge = gr.Plot()
                        with gr.Row():
                            clv_gauge = gr.Plot()

            # Planning
            with gr.Tab("📅 Planning"):
                production_plan = gr.HTML()

    # Event handlers
    preprocess_btn.click(
        preprocess_data,
        inputs=[file_input],
        outputs=[preprocess_status, data_summary]
    )

    train_btn.click(
        train_models,
        outputs=[train_status, model_cards, feature_plot]
    )

    predict_btn.click(
        predict_customer,
        inputs=[age, income, spending_score, purchase_history],
        outputs=[prediction_output, churn_gauge, sales_gauge,
clv_gauge]
    )

    def load_analytics():
        return generate_sales_analytics()

    def load_plan():
        return generate_production_plan()

    app.load(
        load_analytics,
        outputs=[sales_analytics]
```

```
    )

    app.load(
        load_plan,
        outputs=[production_plan]
    )

app.launch()
```

## 8. Predictions

```
def predict_customer(age, income, spending_score, purchase_history):
    global models, df
    if not models:
        return "❌ Please train models first", None, None, None, None

    try:
        income_usd = income / 75

        customer_data = pd.DataFrame([[
            age, income_usd, spending_score, purchase_history, 365, purchase_history/365
        ]], columns=['Age', 'Annual_Income', 'Spending_Score', 'Purchase_History',
                     'Days_Since_First_Purchase', 'Purchase_Frequency'])

        # Predict using ML models
        churn_prob = models['churn'].predict_proba(customer_data)[0][1]
        sales_pred = models['sales'].predict(customer_data)[0]
        clv_pred = models['clv'].predict(customer_data)[0]

        # Create consumer profile
        profile_html = f"""
        <div class='consumer-profile'>
            <h4>🎯 Target Consumer Profile</h4>
            <div style="display: grid; grid-template-columns: 1fr 1fr; gap: 1rem;">
                <div>
                    <p><strong>Age:</strong> {age} years</p>
                    <p><strong>Income:</strong> ₹{income:,.0f}/year</p>
                </div>
                <div>
                    <p><strong>Spending Score:</strong> {spending_score}/100</p>
                    <p><strong>Purchase History:</strong> {purchase_history} items</p>
                </div>
            </div>
            <p style="margin-top: 1rem;"><i>ML analyzes these attributes to predict customer behavior and value</i></p>
        </div>
        """
```

```
    # Create gauges
    churn_gauge = go.Figure(go.Indicator(
        mode="gauge+number",
        value=churn_prob*100,
        domain={'x': [0, 1], 'y': [0, 1]},
        title={'text': "<b>Churn Risk %</b><br><i>ML predicts likelihood to stop purchasing</i>", 'font': {'size': 12}},
        gauge={
            'axis': {'range': [None, 100]},
            'steps': [
                {'range': [0, 30], 'color': "lightgreen"},
                {'range': [30, 70], 'color': "orange"},
                {'range': [70, 100], 'color': "red"}],
            'threshold': {
                'line': {'color': "black", 'width': 4},
                'thickness': 0.75,
                'value': churn_prob*100}
        }
    ))
    churn_gauge.update_layout(height=250, margin=dict(t=50, b=10))

    sales_gauge = go.Figure(go.Indicator(
        mode="number",
        value=sales_pred,
        domain={'x': [0, 1], 'y': [0, 1]},
        title={'text': "<b>Predicted Purchases</b><br><i>ML forecasts future buying behavior</i>", 'font': {'size': 12}},
        number={'font': {'color': "#3a7bd5"}}
    ))
    sales_gauge.update_layout(height=250, margin=dict(t=50, b=10))

    clv_gauge = go.Figure(go.Indicator(
        mode="number",
        value=clv_pred,
        domain={'x': [0, 1], 'y': [0, 1]},
        title={'text': "<b>Predicted CLV (₹)</b><br><i>ML estimates customer lifetime value</i>", 'font': {'size': 12}},
        number={'prefix': "₹", 'valueformat': ",.0f", 'font': {'color': "#1a4b8c"}}
```

```
def generate_recommendation(churn_prob, sales_pred, clv_pred):
    global df
    avg_sales = df['Purchase_History'].mean() if df is not None else 0
    avg_clv = df['Customer_Lifetime_Value'].mean() if df is not None else 0

    # ML-driven business recommendations
    if churn_prob < 0.3:
        churn_status = "🟢 Low Risk"
        action1 = "Maintain engagement"
        detail1 = "ML suggests standard retention strategies"
    elif churn_prob < 0.7:
        churn_status = "🟡 Medium Risk"
        action1 = "Loyalty benefits"
        detail1 = "ML recommends targeted incentives"
    else:
        churn_status = "🔴 High Risk"
        action1 = "Immediate action"
        detail1 = "ML flags for urgent retention efforts"

    if sales_pred > avg_sales * 1.5:
        sales_status = "📈 High Potential"
        action2 = "Upsell premium"
        detail2 = "ML identifies high-value opportunities"
    elif sales_pred > avg_sales:
        sales_status = "📊 Steady"
        action2 = "Cross-sell"
        detail2 = "ML suggests complementary products"
    else:
        sales_status = "📉 Low Activity"
        action2 = "Re-engage"
        detail2 = "ML recommends win-back campaigns"
```

```
return f"""
<div style="display: grid; grid-template-columns: 1fr 1fr 1fr; gap: 1rem;">
    <div>
        <h5>Churn Analysis</h5>
        <p><strong>{churn_status}</strong></p>
        <p>Probability: {churn_prob*100:.1f}%</p>
        <p>Action: {action1}</p>
        <p><small>{detail1}</small></p>
    </div>
    <div>
        <h5>Sales Potential</h5>
        <p><strong>{sales_status}</strong></p>
        <p>Predicted: {sales_pred:.1f}</p>
        <p>Action: {action2}</p>
        <p><small>{detail2}</small></p>
    </div>
    <div>
        <h5>Customer Value</h5>
        <p><strong>{clv_status}</strong></p>
        <p>₹{clv_pred:,.0f}</p>
        <p>Action: {action3}</p>
        <p><small>{detail3}</small></p>
    </div>
</div>
"""
```

Those were the code snippets of the project highlighting all the technologies and process we used.

# TESTING AND RESULTS

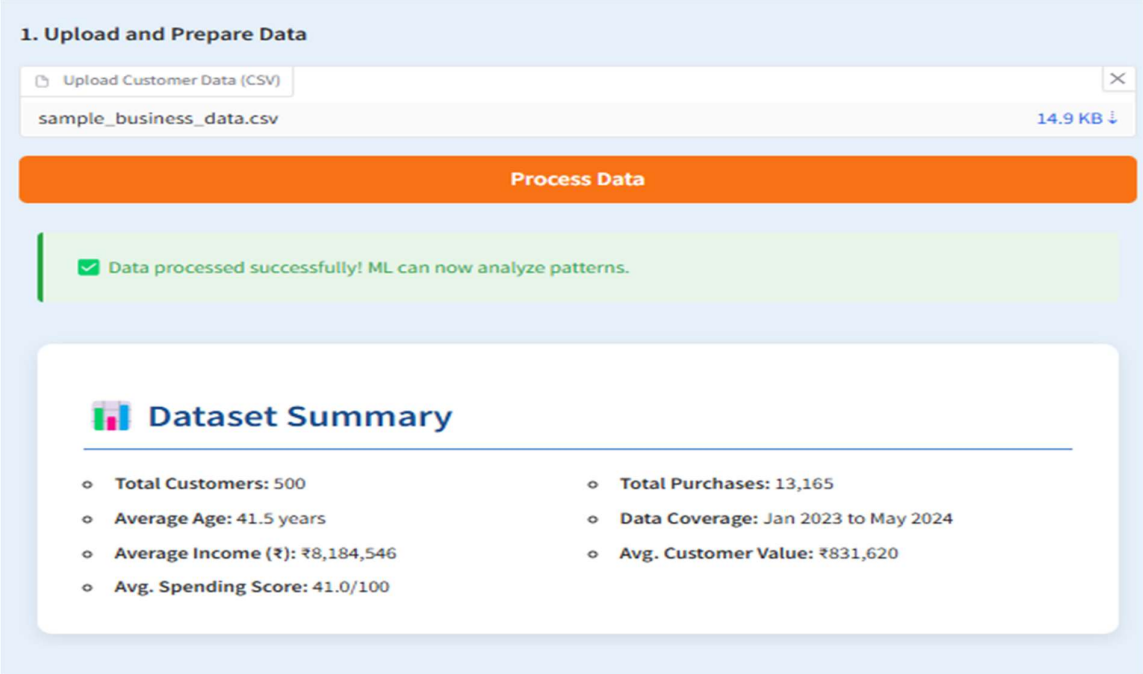


Fig 1.1 Uploading data

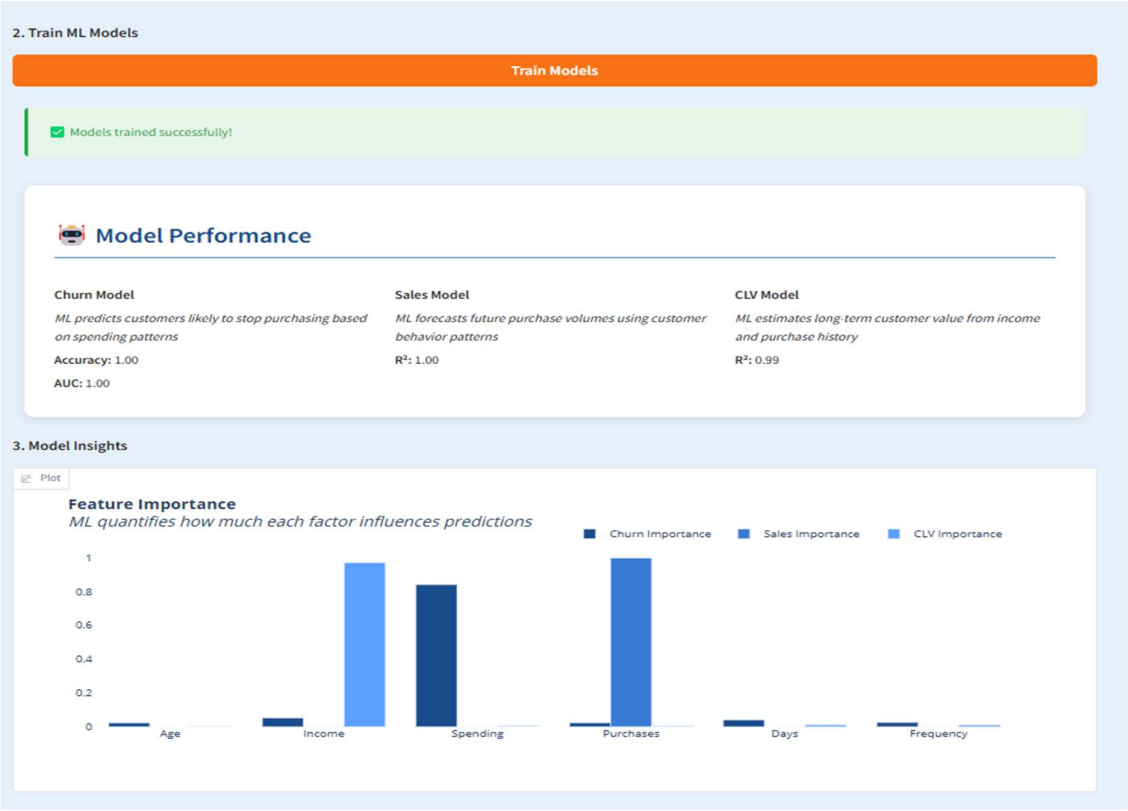


Fig 1.2 Training ML Model

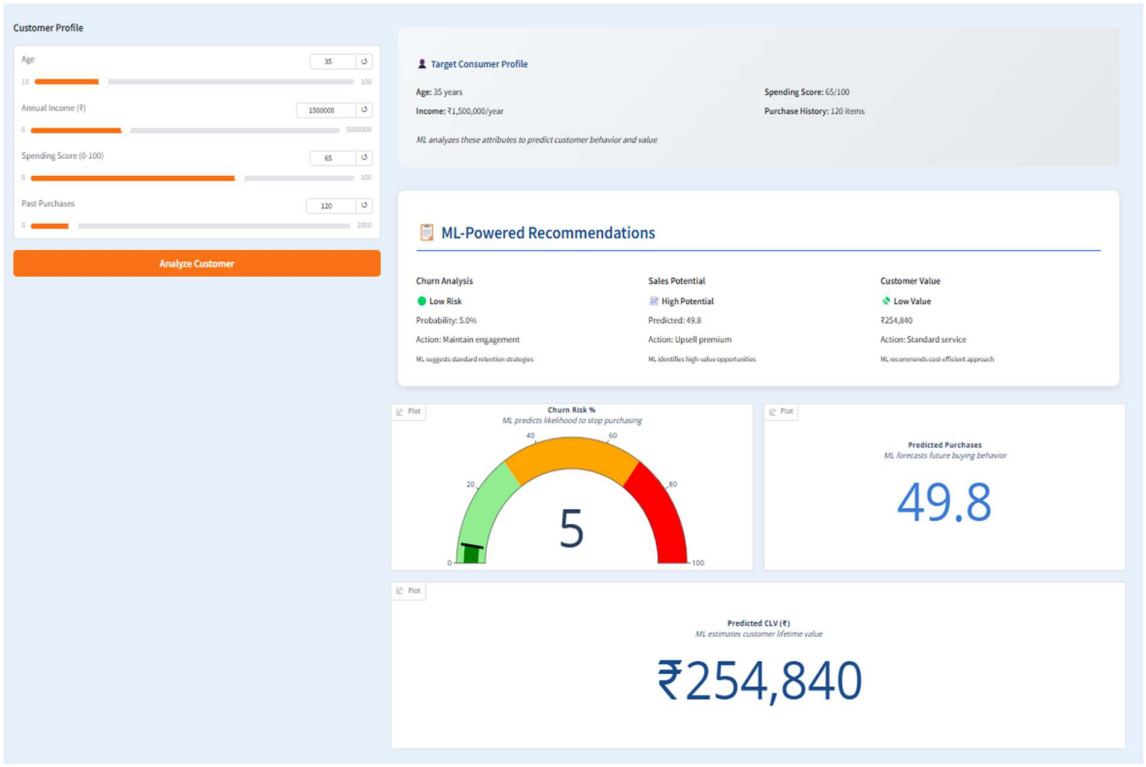


Fig 1.3 Recommendations page

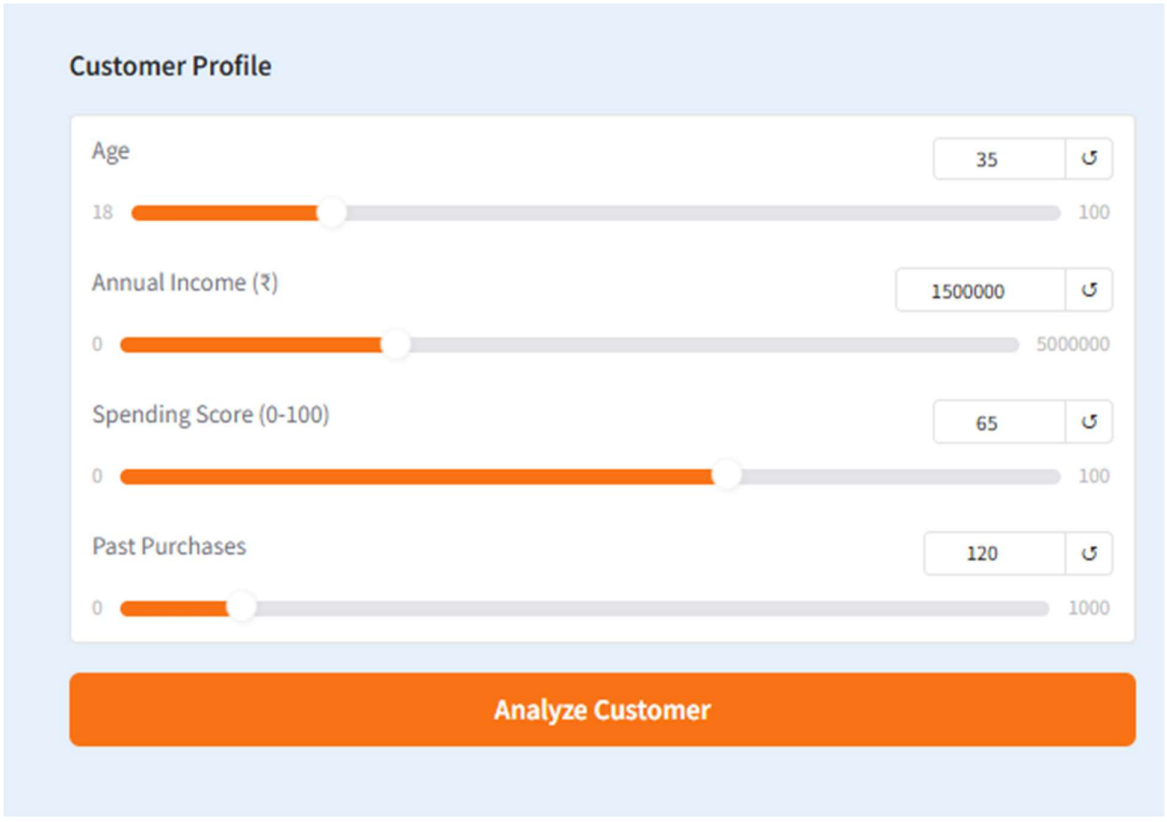


Fig 1.4 Parameters adjuster

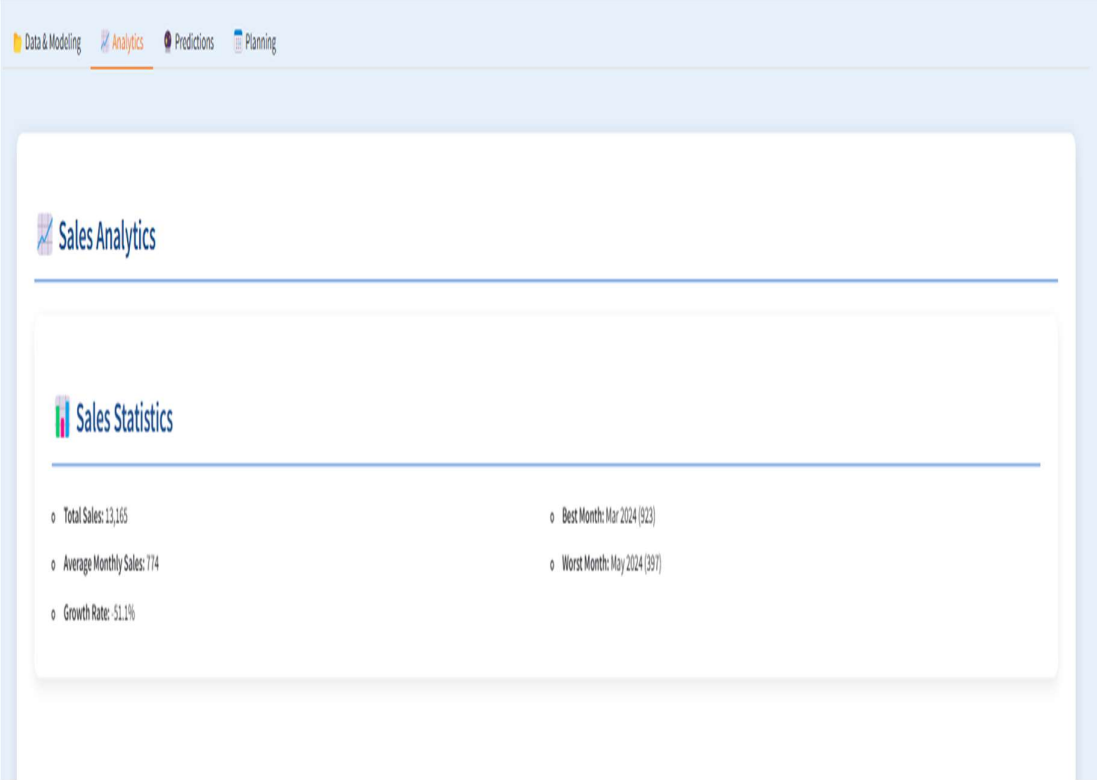


Fig 1.5 Sales Analysis

# CONCLUSION AND FUTURE SCOPE

## 6.1 CONCLUSION

The Customer Churn Prediction System presents a comprehensive and intelligent approach to addressing one of the most pressing challenges faced by modern businesses—retaining valuable customers. Through the integration of machine learning algorithms, particularly the Random Forest Classifier, the system leverages historical data to predict churn with a high degree of accuracy. This enables businesses to identify at-risk customers and implement timely interventions, significantly reducing customer attrition rates. The model considers essential customer attributes such as age, income, spending habits, and purchase history, ensuring predictions are grounded in relevant behavioral patterns. The system is further strengthened by the use of integrated data visualizations, which provide intuitive and actionable insights into customer behavior, while embedded strategy recommendations help guide decision-making processes. With a Gradio-powered frontend, the tool is designed to be user-friendly and accessible, catering to both technical and non-technical users alike. This ensures a wider adoption across various departments, from marketing to customer service. Beyond its immediate capabilities, the system promotes long-term profitability by lowering customer acquisition costs and enhancing retention strategies. Its modular architecture allows for scalability and future enhancements, such as real-time data processing, deeper personalization through AI-driven recommendations, and integration with CRM platforms. In essence, this system is not only a powerful predictive tool but also a strategic asset that helps businesses build stronger customer relationships, improve operational efficiency, and gain a competitive edge in today’s dynamic market landscape.

## 6.2 LIMITATIONS

• Basic Security Measures: The system lacks advanced security features like user authentication, encryption, or access control, posing a risk for sensitive data exposure.
• Limited Feature Set: Predictions are based on a few basic attributes; complex behavioral and transactional patterns are not yet incorporated.
• No Real-Time Data Integration: The system relies on static file uploads or manual entries, without support for live database connectivity.
• Lack of Personalized Strategy Suggestions: Business recommendations are based on overall trends rather than tailored to individual customer profiles.
• Single User Mode: There is no role differentiation (e.g., admin, analyst) for managing multiple user activities or customizing access levels.

## 6.3 FUTURE SCOPE

1. **Incorporate Advanced Features:**
    - Integrate additional customer attributes such as contract type, tenure duration, service usage patterns, and payment behavior.
    - Employ feature engineering to derive new insights, such as customer lifetime value or engagement scores, enhancing model accuracy.
2. **Enhance Security:**
    - Implement multi-factor authentication and role-based access control to secure user access.
    - Ensure end-to-end encryption of all data transmissions and secure storage using modern encryption standards (e.g., AES-256).
3. **Real-Time Data Integration:**
    - Establish APIs to pull real-time data from CRM systems or customer service databases for immediate churn predictions.
    - Enable data streaming with platforms like Kafka or Firebase to allow dynamic updates and alerting mechanisms.
4. **Personalized Recommendations:**
    - Use clustering or segmentation techniques to identify user personas and generate targeted retention strategies.
    - Develop automated intervention tools (like sending customized offers or support follow-ups) based on predicted churn risk levels.
5. **Expand Platform:**
    - Build a comprehensive web-based dashboard with interactive visualizations, user analytics, and churn heatmaps.
    - Introduce administrative tools for user creation, audit logging, and detailed reporting across different business verticals.
6. **Model Optimization and Explainability:**
    - Continuously improve the model with A/B testing, hyperparameter tuning, and ensemble methods.

# BIBLIOGRAPHY


- **Scikit-learn Documentation**
  Used for building the Random Forest Classifier and other machine learning functionalities.
  https://scikit-learn.org/stable/documentation.html
- **Gradio Documentation**
  Referred for creating the interactive web-based user interface for churn prediction.
  https://gradio.app/docs/
- **Pandas Documentation**
  Used for data manipulation, preprocessing, and CSV handling operations.
  https://pandas.pydata.org/docs/
- **Seaborn and Matplotlib Documentation**
  Used for creating dynamic data visualizations like histograms, box plots, and bar charts.
  https://seaborn.pydata.org/
  https://matplotlib.org/stable/contents.html
- **Python Data Science Handbook – Jake VanderPlas**
  Referenced for machine learning techniques, data analysis methods, and visualization examples.
- **Kaggle: Customer Churn Datasets**
  Used as a reference for understanding customer churn features and typical dataset formats.
  https://www.kaggle.com/