# Project Summary – Insurance Churn Prediction

## 🔹 Objective

To build a machine learning model that predicts customer churn for an insurance company and segment customers based on churn risk to support better retention planning.

## 🔹 Tools Used

* Python (pandas, scikit-learn, SHAP)
* Google Colab
* Excel (initial data review)

## 🔹 Key Steps

1. Cleaned and explored the dataset (handled missing values, dropped unused columns)
2. Built a logistic regression model with class balancing to predict churn
3. Evaluated model performance using accuracy, recall, F1-score, and confusion matrix
4. Explained predictions using SHAP (individual waterfall plots and global bar charts)
5. Segmented customers into churn-risk categories based on model probabilities:
   * 🟢 Loyal (probability < 0.30)
   * 🟡 At Risk (0.30 ≤ probability < 0.70)
   * 🔴 Likely Churn (probability ≥ 0.70)

## 🔹 Results

* Model Accuracy: 60.86%
* Churn Recall: 65% (successfully identified most churners)
* Top churn drivers:
  + Days of tenure
  + Annual premium
  + Length of residence
* Final segmented file: churn\_risk\_segments.csv

## 🔹 Deliverables

* insurance\_churn\_clean.csv → Cleaned input data
* churn\_model.ipynb → Full Python notebook
* churn\_risk\_segments.csv → Final predictions with churn probabilities and risk segments
* churn\_summary.docx → Project summary report (this document)

## 🔹 Conclusion

This project successfully predicted customer churn and segmented customers by churn risk. SHAP-based explainability made the model transparent and provided actionable insights to support retention strategies.

# 📘 Detailed Project Walkthrough

## 1. Data Understanding & Preparation

The dataset included demographic, financial, and behavioral information about insurance policyholders.

* Demographics: age, marital status, city, state
* Financials: annual premium (curr\_ann\_amt), income, home market value
* Behavioral: days of tenure, good\_credit, length\_of\_residence, has\_children

Steps:

* Dropped unnecessary columns (e.g., IDs, date columns)
* Created an “Age Group” column
* Verified datatypes and handled missing values
* Saved cleaned data as insurance\_churn\_clean.csv

## 2. Exploratory Data Analysis (EDA)

* Used Excel and Python for churn rate analysis by age group
* Created PivotTables to show churn counts and percentages
* Found that customers aged 45–59 had the highest churn (~46%)
* Identified class imbalance: majority of customers were retained

## 3. Feature Engineering & Encoding

* One-hot encoded categorical features: state, city, marital\_status, Age Group
* Kept numerical features: days\_tenure, curr\_ann\_amt, income, etc.
* Confirmed target variable (Churn) was binary
* Split data into training and testing sets (80/20)

## 4. Model Building – Logistic Regression

* Used Logistic Regression with class\_weight='balanced' to handle imbalance
* Trained using scikit-learn
* Performance:
  + Accuracy: 60.86%
  + Churn Recall: 65%
* Confusion matrix showed improved detection of actual churners vs. baseline

## 5. Explainability with SHAP

* Used SHAP to explain predictions at both individual and global levels
* Generated waterfall plots to show how features influenced specific predictions
* Global bar plots showed top contributing factors to churn

Key Drivers of Churn:

* Shorter days\_tenure
* Lower length\_of\_residence
* Annual premium (curr\_ann\_amt)
* Age Group (especially 45–59)

## 6. Customer Segmentation Based on Churn Probability

* Used model.predict\_proba() to calculate churn probabilities
* Classified test customers into:
  + Loyal (0.00 – 0.29)
  + At Risk (0.30 – 0.69)
  + Likely Churn (0.70 – 1.00)
* Stored results in churn\_risk\_segments.csv

This file is useful for:

* Visualizing churn risk in dashboards
* Targeting retention campaigns
* Reporting to business stakeholders

## 7. Final Recommendations

* Focus retention campaigns on At Risk and Likely Churn customers
* Prioritize outreach to mid-tenure customers with moderate policies
* Use churn probability as a dynamic metric in CRM systems
* In future versions, experiment with tree-based models like Random Forest or XGBoost for potential performance gain

Top of Form

Bottom of Form