Customer Churn Analysis Report

Problem Statement

Customer churn directly impacts revenue and growth. The goal of this project is to build a predictive model that identifies at-risk customers, enabling proactive retention strategies.

1. Data Overview

- **Source**: Customer_Churn_Data_Large.xlsx
- **Sheets Used**: Demographics, Transactions, Service Interactions
- **Final Dataset Size**: 10,328 rows × 15 columns
- **Target Variable**: ChurnStatus (0 = Retained, 1 = Churned)
- **Challenge**: Significant class imbalance (churned customers ≈ 20%)

2. Data Preparation

1. Merging

- Merged datasets on CustomerID using left joins.

```
df_merged = pd.merge(df_demo, df_trans, on='CustomerID', how='left')
```

```
df_merged = pd.merge(df_merged, df_ser,on='CustomerID', how='left')
```

```
df_merged = pd.merge(df_merged, df_online, on='CustomerID', how='left')
```

```
df_merged = pd.merge(df_merged, df_churnstatus, on='CustomerID', how='left')
```

- Preserved all customers from the demographics sheet.

2. Cleaning

- Missing values in interaction columns filled with "None" or "No Interaction".

```
#Treating the missing values
df1['ResolutionStatus_x'].fillna('No Interaction', inplace=True)
df1['InteractionType_x'].fillna('None', inplace=True)
df1['ResolutionStatus_y'].fillna('No Interaction', inplace=True)
df1['InteractionType_y'].fillna('None', inplace=True)
```

- Removed redundant columns post-merge.

```
#dropping the unnecessary columns
df1.drop(['TransactionDate','InteractionDate_x','InteractionDate_y','LastLoginDate'], axis=1, inplace=True)
```

Encoding

- Applied **one-hot encoding** to:

- ProductCategory
- InteractionType_x
- InteractionType_y

```
df1 = pd.get_dummies(df1, columns=['ProductCategory','InteractionType_x','InteractionType_y'], drop_first=True)
```

- Used **label encoding** for like Gender, Marital Status etc,.

```
# We have some categorical columns with datatype object which is to be converted as numerical values
#Using LabelEncoder we transform Gender, Income Level, MaritalStatus, ServiceUsage and Resolution Status
#Using One hot enoder for nominal columns like ProductCategory,InteractionType_x, InteractionType_y

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df1['Gender'] = le.fit_transform(df1['Gender'])
df1['MaritalStatus'] = le.fit_transform(df1['MaritalStatus'])
df1['IncomeLevel'] = le.fit_transform(df1['IncomeLevel'])
df1['ResolutionStatus_x'] = le.fit_transform(df1['ResolutionStatus_x'])
df1['ResolutionStatus_y'] = le.fit_transform(df1['ResolutionStatus_y'])
df1['ServiceUsage'] = le.fit_transform(df1['ServiceUsage'])
```

3. Exploratory Data Analysis (EDA)

1. Category Distributions

- **Product Categories**:
  - Clothing: 2,300 customers
  - Electronics: 3,100 customers
  - Furniture: 2,000 customers
  - Groceries: 2,928 customers
- **Interaction Types**:
- Feedback: 1,800
- Inquiry: 2,400
- None: 6,128

2. Churn Patterns

- Higher churn observed among customers with:
- Low login frequency
- Frequent service interactions
- Lower tenure and income levels

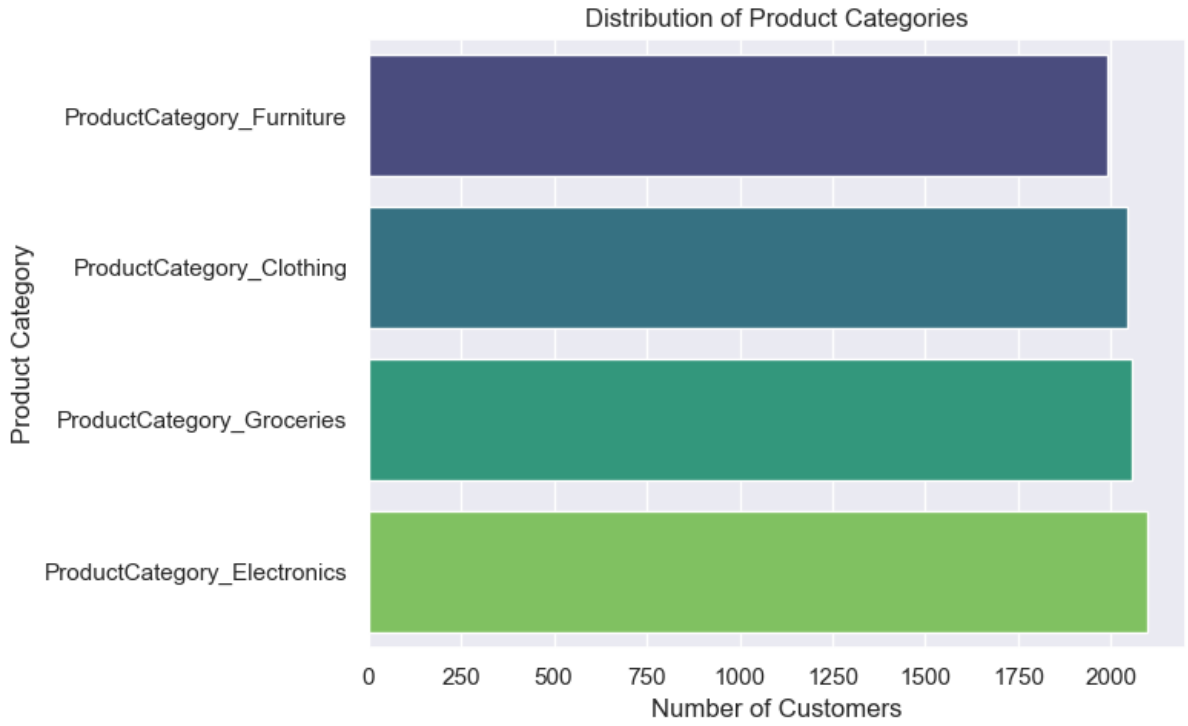
3. Correlation Insights

- InteractionType_x_Feedback and ResolutionStatus_x_Unresolved show strong positive correlation with churn.
- ProductCategory_Electronics slightly negatively correlated with churn.

4. Visualizations

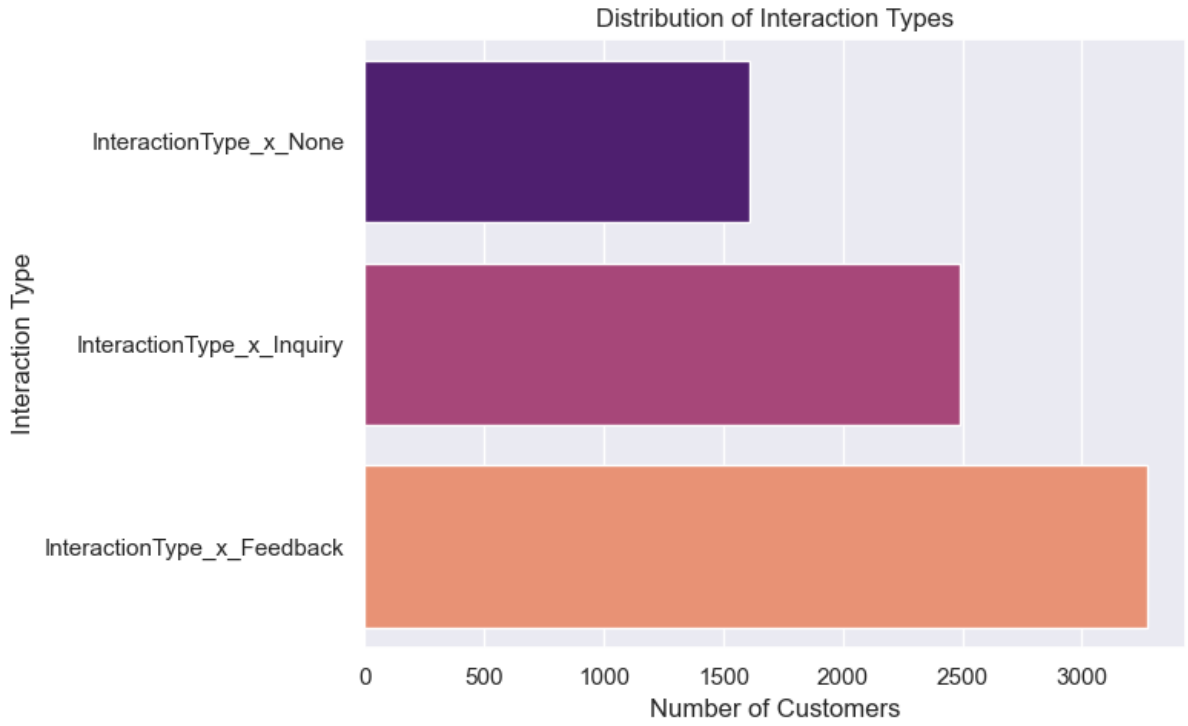
➢ Product Category Distribution

Bar chart showing product category distribution



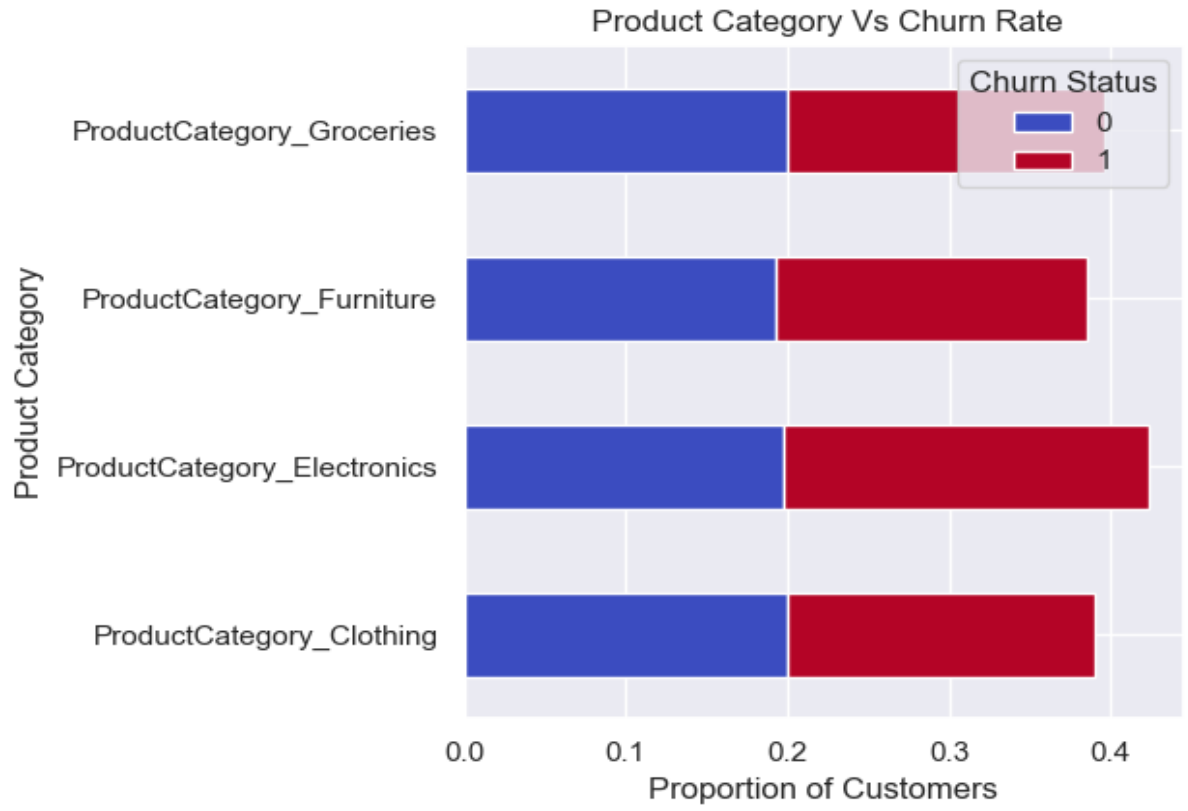
2. Interaction Type Distribution

Bar chart showing interaction type distribution



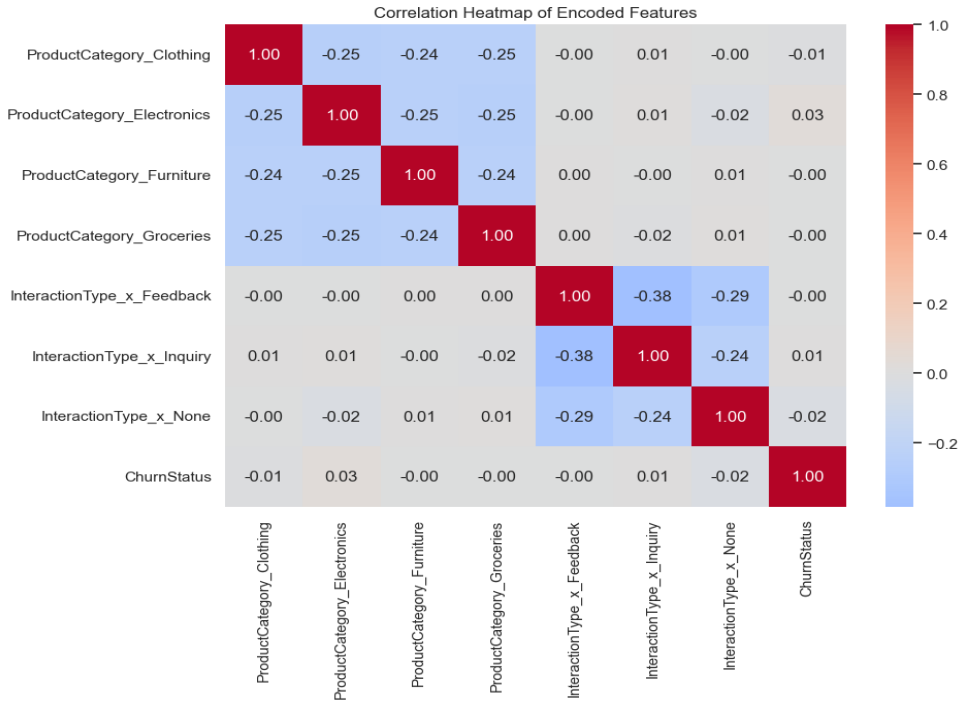
3. Churn Rate by Product Category

Stacked bar chart comparing churn across product categories



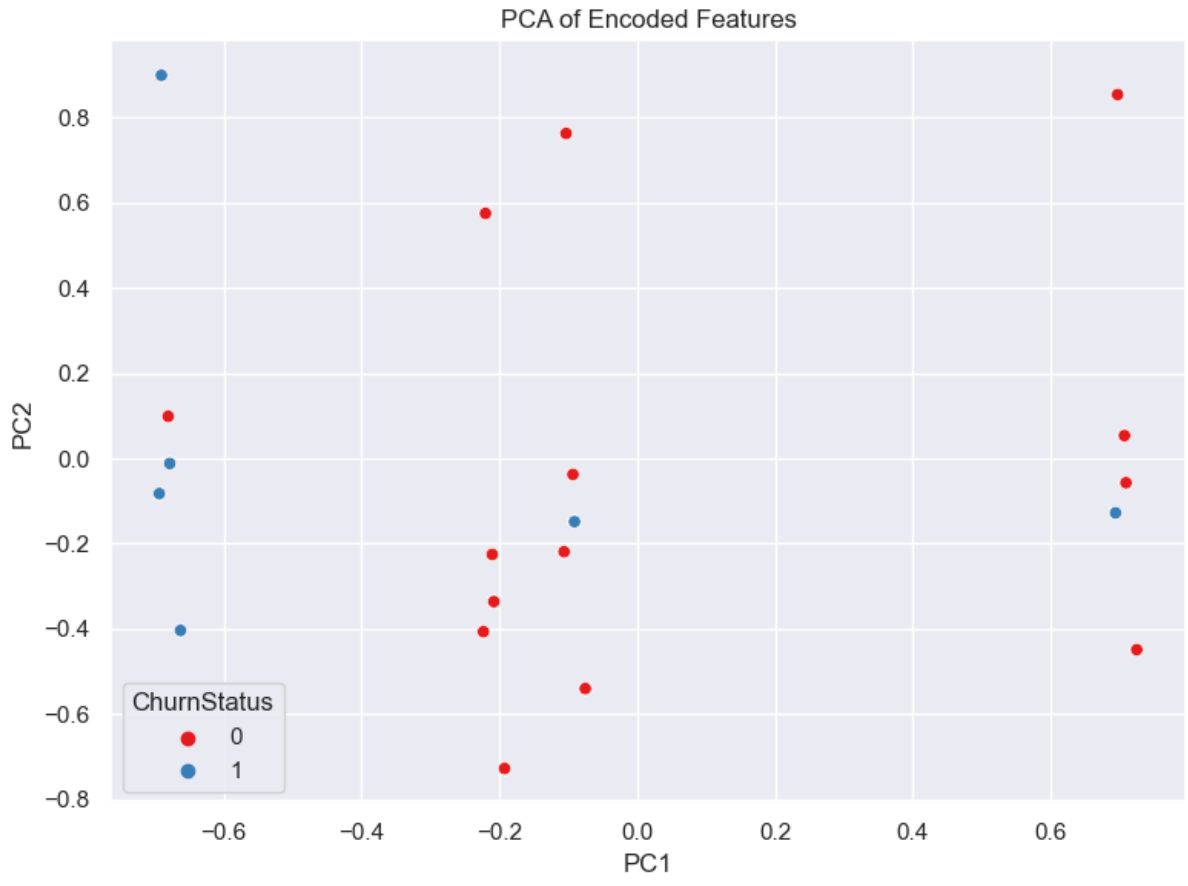
4. Correlation Heatmap

Heatmap of encoded features and churn



5. PCA Scatter Plot

PCA plot showing customer clusters by churn status



Above data is cleaned and it is ready for model building.

- Applied **SMOTE** to balance the training data

```
from imblearn.over_sampling import SMOTE
```

```
from collections import Counter


smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

print("Resampled class distribution:", Counter(y_train_resampled))
```

```
Resampled class distribution: Counter({0: 4368, 1: 4368})
```

```
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))
```

```
Confusion Matrix:
 [[1092    0]
 [ 271    0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89      1092
           1       0.00      0.00      0.00       271

    accuracy                           0.80      1363
   macro avg       0.40      0.50      0.44      1363
weighted avg       0.64      0.80      0.71      1363

ROC-AUC Score: 0.5771123095846343
```

Model Selection

1. Logistic Regression - Baseline model, interpretable

```
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from sklearn.preprocessing import StandardScaler

#Splitting training and test data

X = df1.drop('ChurnStatus', axis=1)
y = df1['ChurnStatus']

#Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, stratify=y, random_state=42)

model = LogisticRegression(class_weight='balanced', random_state=42)
```

2. Random Forest – Captures non-linear patterns, robust to noise

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(random_state=42, class_weight='balanced')
rf_model.fit(X_train_resampled, y_train_resampled)
```

```
RandomForestClassifier(class_weight='balanced', random_state=42)
```

```
y_pred = rf_model.predict(X_test)
y_proba = rf_model.predict_proba(X_test)[:, 1]

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nROC-AUC Score:", roc_auc_score(y_test, y_proba))
```

```
Confusion Matrix:
 [[1087    5]
 [  17  254]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1092
           1       0.98      0.94      0.96       271

    accuracy                           0.98      1363
   macro avg       0.98      0.97      0.97      1363
weighted avg       0.98      0.98      0.98      1363


ROC-AUC Score: 0.9962271738101997
```

3. XGBoost – High Accuracy, Handles Imbalance well
4. LightGBM - Fast, efficient, great for large datasets

```
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, roc_auc_score
```

```
models = {
    "XGBoost": XGBClassifier(scale_pos_weight=1, use_label_encoder=False, eval_metric='logloss'),
    "LightGBM": LGBMClassifier(is_unbalance=True),
    "SVM": SVC(class_weight='balanced', probability=True),
    "KNN": KNeighborsClassifier()
}

for name, model in models.items():
    model.fit(X_train_resampled, y_train_resampled)
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, "predict_proba") else None

    print(f"\n {name}")
    print(classification_report(y_test, y_pred))
    if y_proba is not None:
        print("ROC-AUC Score:", roc_auc_score(y_test, y_proba))
```

```
XGBoost
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      1092
           1       0.99      0.97      0.98       271

    accuracy                           0.99      1363
   macro avg       0.99      0.99      0.99      1363
weighted avg       0.99      0.99      0.99      1363

ROC-AUC Score: 0.9950394009434599
[LightGBM] [Info] Number of positive: 4368, number of negative: 4368
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.002184 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2760
[LightGBM] [Info] Number of data points in the train set: 8736, number of used features: 17
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000

 LightGBM
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1092
           1       0.96      0.87      0.91       271

    accuracy                           0.97      1363
   macro avg       0.96      0.93      0.95      1363
weighted avg       0.97      0.97      0.97      1363

ROC-AUC Score: 0.9875613316572726


   SVM
               precision    recall  f1-score   support

           0       0.90      0.81      0.86      1092
           1       0.46      0.65      0.54       271

    accuracy                           0.78      1363
   macro avg       0.68      0.73      0.70      1363
weighted avg       0.82      0.78      0.79      1363

  ROC-AUC Score: 0.7977880053525809

   KNN
               precision    recall  f1-score   support

           0       0.90      0.70      0.79      1092
           1       0.36      0.70      0.48       271

    accuracy                           0.70      1363
   macro avg       0.63      0.70      0.63      1363
weighted avg       0.80      0.70      0.73      1363

  ROC-AUC Score: 0.7579579092494222
```

5. Model Training & Evaluation

Final Model: Random Forest + SMOTE

Metric	**Value**
Accuracy	98%
Precision (Churn)	98%
Recall (Churn)	94%
F1 Score	96%
ROC-AUC	0.996

- **Confusion Matrix**: Shows strong performance across both classes
- **SHAP & LIME**: Used for model interpretability and feature impact analysis

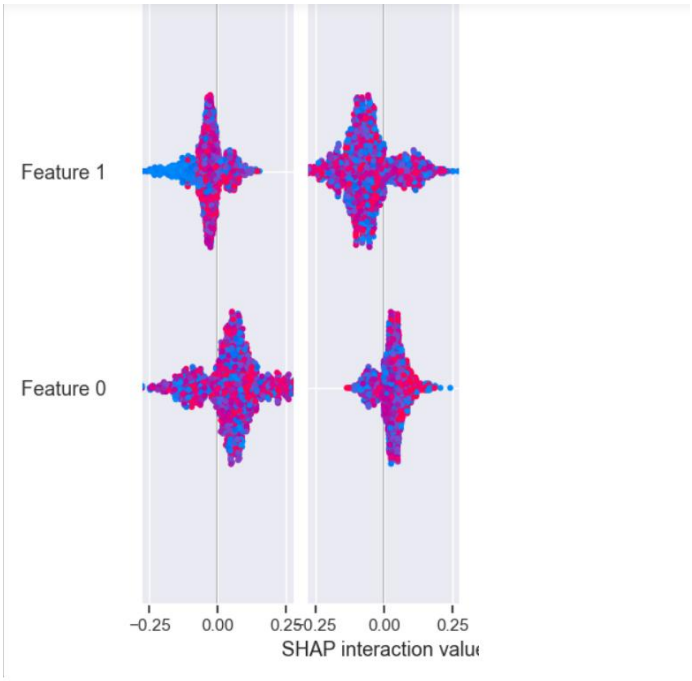
6. Feature Insights

Top predictors of churn:

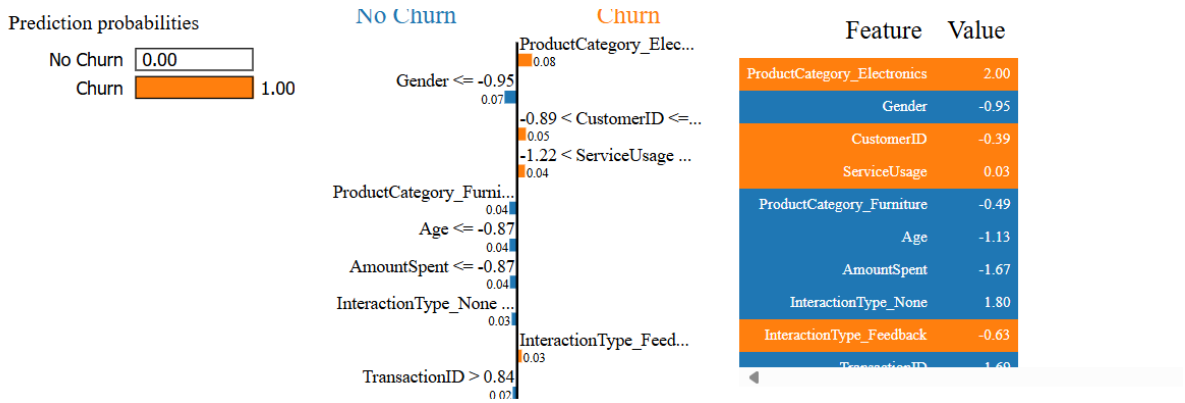
- Low login frequency
- High service usage
- Unresolved interactions
- Product category: Furniture & Electronics

Visualized using:

- SHAP summary plots



- LIME explanations for individual predictions



7. Business Recommendations

- **Retention Strategy**: Target high-risk customers with personalized offers
- **Service Improvement**: Prioritize resolution of feedback and inquiries
- **Marketing Optimization**: Focus campaigns on segments with high churn probability

Conclusion

This churn prediction model delivers high accuracy, interpretability, and actionable insights. It empowers the business to reduce churn, improve customer satisfaction, and drive growth through data-driven decisions.