CREDIT RISK,

SENTIMENT ANALYSIS

&

N.P.S. (NET PROMOTER SCORE)

Tricia

Turning data into meaningful stories with actionable insights

Data Analyst | Python • Power BI • SAS • SQL • Sentiment Analysis • NPS

AGENDA

1. Data Cleansing & Quality Check
2. Correlation
3. Voluntary Attrition & Sentiment Analysis
4. Involuntary Attrition & NPS
5. ML Model
6. Summary

Full List of Data Cleansing / Quality Check

1. Initial Inspection

- df.head() / df.sample() — View data sample
- df.shape — Rows & columns
- df.info() — Check column types and nulls
- df.describe() — Summary statistics
- df.columns — List of column names

2. Structural Integrity

- Check for **duplicate rows**
- df = df.drop_duplicates()
- Check for **duplicate column names**
- df.columns.duplicated().any()

3. Missing Data

- Identify missing values
- df.isnull().sum()
- Decide how to handle:
  - **Drop rows**: df.dropna()
  - **Fill values**: df.fillna(method='ffill') or use mean/median
  - **Flag them**: create an indicator column

4. Data Types

- Ensure correct data types
- df.dtypes
- Convert as needed
- df['date'] = pd.to_datetime(df['date'])
- df['category'] = df['category'].astype('category')

5. Outliers and Invalid Values

- Visualize with boxplots or histograms
- Use z-score or IQR method
- Check for impossible values (e.g. negative ages, invalid ZIP codes)

6. Consistency Checks

- Standardize text:
- df['column'] = df['column'].str.lower().str.strip()
- Ensure consistent units (e.g. dollars vs cents, kg vs lb)
- Align formats (e.g. phone numbers, SSNs, ICD codes)

7. Unique Identifiers

- Confirm that key columns (e.g., ID, claim number) are unique:
- df['ID'].is_unique
- Check for duplicates:
- df[df.duplicated(subset='ID')]

8. Invalid Categories or Values

- Use .unique() to inspect:
- df['status'].unique()
- Cross-check with expected values (e.g. ['active', 'inactive'])

9. Date and Time Consistency

- Convert to datetime: pd.to_datetime()
- Sort and check sequence/order
- Ensure no future dates for things like birth or service dates

10. Numerical Checks

- Check for negatives where not allowed (e.g. income, age)
- Validate ranges
- Handle infinite or NaN values
- df = df.replace([np.inf, -np.inf], np.nan)

11. Data Normalization (Optional, ML-focused)

- Scaling numeric features (MinMax or StandardScaler)
- Encoding categorical variables (Label or OneHot)

12. Column Renaming and Documentation

- Clean messy column names
- df.columns = df.columns.str.strip().str.lower().str.replace(' ', '_')
- Document your data dictionary

Data Cleansing / Quality Check in Progress (1)

```
df.shape  #Rows & columns
```

```
(9015, 40)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9015 entries, 0 to 9014
Data columns (total 40 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   CLIENTNUM                 9015 non-null   object
 1   Attrition_Flag            9015 non-null   object
 2   Customer_Age              9015 non-null   int64
 3   Gender                    9015 non-null   object
 4   Dependent_count           9015 non-null   int64
 5   Education_Level           7641 non-null   object
 6   Marital_Status            8348 non-null   object
 7   Income_Category           9015 non-null   object
 8   Card_Category             9015 non-null   object
 9   Months_on_book            9015 non-null   int64
 10  Total_Relationship_Count  9015 non-null   int64
 11  Months_Inactive_12_mon    9015 non-null   int64
 12  Contacts_Count_12_mon     9015 non-null   int64
 13  Credit_Limit              9015 non-null   float64
 14  Total_Revolving_Bal       9015 non-null   int64
 15  Avg_Open_To_Buy           9015 non-null   float64
 16  Total_Amt_Chng_Q4_Q1      9015 non-null   float64
 17  Total_Trans_Amt           9015 non-null   int64
 18  Total_Trans_Ct            9015 non-null   int64
 19  Total_Ct_Chng_Q4_Q1       9015 non-null   float64
 20  Avg_Utilization_Ratio     9015 non-null   float64
 21  NB_Attrition_Prob_Yes     9015 non-null   float64
 22  NB_Attrition_Prob_No      9015 non-null   float64
 23  Churn_Type                9015 non-null   object
 24  annual_income             9015 non-null   int64
 25  monthly_income            9015 non-null   float64
 26  DTI_ratio                 9015 non-null   float64
 27  DTI_Risk_Level            9015 non-null   category
 28  Credit_Score              9015 non-null   float64
 29  Loan_Default_Flag         9015 non-null   int64
 30  Number_of_Defaults        9015 non-null   int64
 31  Last_Default_Date         9015 non-null   object
 32  Default_Amount            9015 non-null   float64
 33  Bankruptcy_History        9015 non-null   object
 34  Bankruptcy_Flag           9015 non-null   int64
 35  Adjusted_Credit_Score     9015 non-null   float64
 36  Moodys_Rating             9015 non-null   category
 37  Payment_History_Flag      9015 non-null   object
 38  Payment_Risk              9015 non-null   object
 39  estimated_annual_revenue  9015 non-null   float64
dtypes: category(2), float64(13), int64(13), object(12)
memory usage: 2.6+ MB
```

Data Cleansing / Quality Check in Progress (2)

```
df.describe().T
```

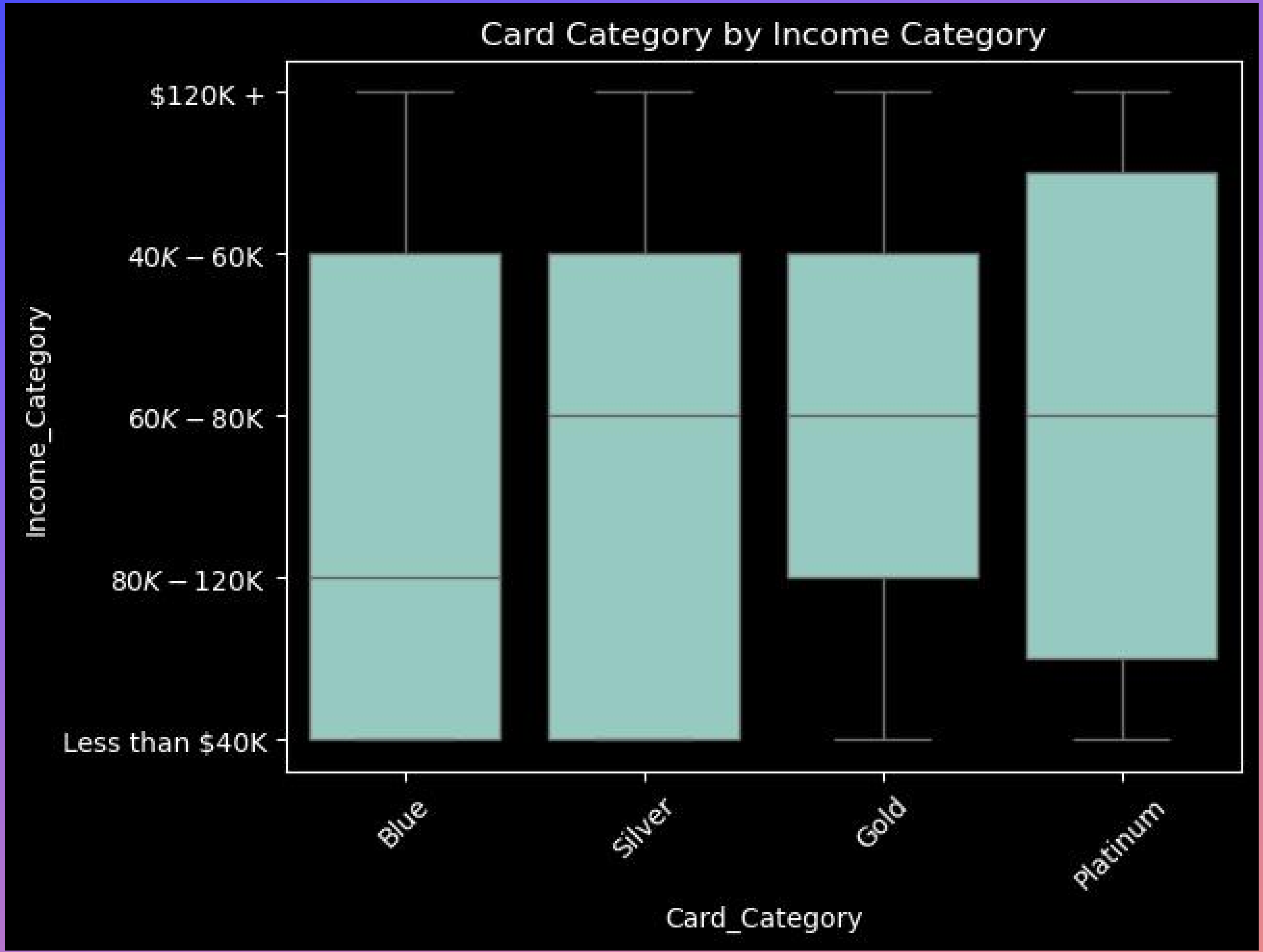
	count	mean	std	min	25%	50%	75%	max
Customer_Age	10128.0	46.326521	8.016616	26.000000	41.000000	46.000000	52.000000	73.000000
Dependent_count	10128.0	2.346465	1.299112	0.000000	1.000000	2.000000	3.000000	5.000000
Months_on_book	10128.0	35.928910	7.986181	13.000000	31.000000	36.000000	40.000000	56.000000
Total_Relationship_Count	10128.0	3.812599	1.554332	1.000000	3.000000	4.000000	5.000000	6.000000
Months_Inactive_12_mon	10128.0	2.342121	1.015120	0.000000	2.000000	2.000000	3.000000	12.000000
Contacts_Count_12_mon	10128.0	2.455075	1.106440	0.000000	2.000000	2.000000	3.000000	6.000000
Credit_Limit	10128.0	8634.557178	9092.103865	1438.300000	2555.000000	4549.000000	11070.250000	35000.000000
Total_Revolving_Bal	10128.0	1162.947769	815.058178	0.000000	360.000000	1276.500000	1784.000000	2517.000000
Avg_Open_To_Buy	10128.0	7471.577814	9093.547561	3.000000	1324.750000	3474.500000	9861.750000	34516.000000
Total_Amt_Chng_Q4_Q1	10127.0	0.759941	0.219207	0.000000	0.631000	0.736000	0.859000	3.397000
Total_Trans_Amt	10127.0	4404.086304	3397.129254	510.000000	2155.500000	3899.000000	4741.000000	18484.000000
Total_Trans_Ct	10127.0	64.858695	23.472570	10.000000	45.000000	67.000000	81.000000	139.000000
Total_Ct_Chng_Q4_Q1	10127.0	0.712222	0.238086	0.000000	0.582000	0.702000	0.818000	3.714000
Avg_Utilization_Ratio	10127.0	0.274894	0.275691	0.000000	0.023000	0.176000	0.503000	0.999000
NB_Attrition_Prob_Yes	10127.0	0.159997	0.365301	0.000008	0.000099	0.000181	0.000337	0.999580
NB_Attrition_Prob_No	10127.0	0.840003	0.365301	0.000420	0.999660	0.999820	0.999900	0.999990
annual_income	9016.0	62535.457964	39300.348399	20001.000000	32856.750000	50407.000000	80095.750000	199975.000000
monthly_income	9016.0	5211.288164	3275.029033	1666.750000	2738.062500	4200.583333	6674.645833	16664.583333
DTI_ratio	9016.0	30.768135	29.151431	0.000000	6.177866	24.654816	46.680241	149.985103
Credit_Score	10128.0	640.672788	114.324010	450.000000	544.000000	635.000000	737.000000	850.000000
Loan_Default_Flag	10128.0	0.222650	0.416046	0.000000	0.000000	0.000000	0.000000	1.000000
Number_of_Defaults	10128.0	0.677626	1.432613	0.000000	0.000000	0.000000	0.000000	5.000000
Default_Amount	10128.0	2275.305247	5008.536585	0.000000	0.000000	0.000000	0.000000	19999.460000
Bankruptcy_Flag	10128.0	0.100316	0.300436	0.000000	0.000000	0.000000	0.000000	1.000000
Adjusted_Credit_Score	10128.0	625.625395	126.250679	301.000000	525.000000	622.000000	730.000000	850.000000
estimated_annual_revenue	10128.0	282.589554	163.011636	50.000000	122.000000	305.300000	406.800000	553.400000

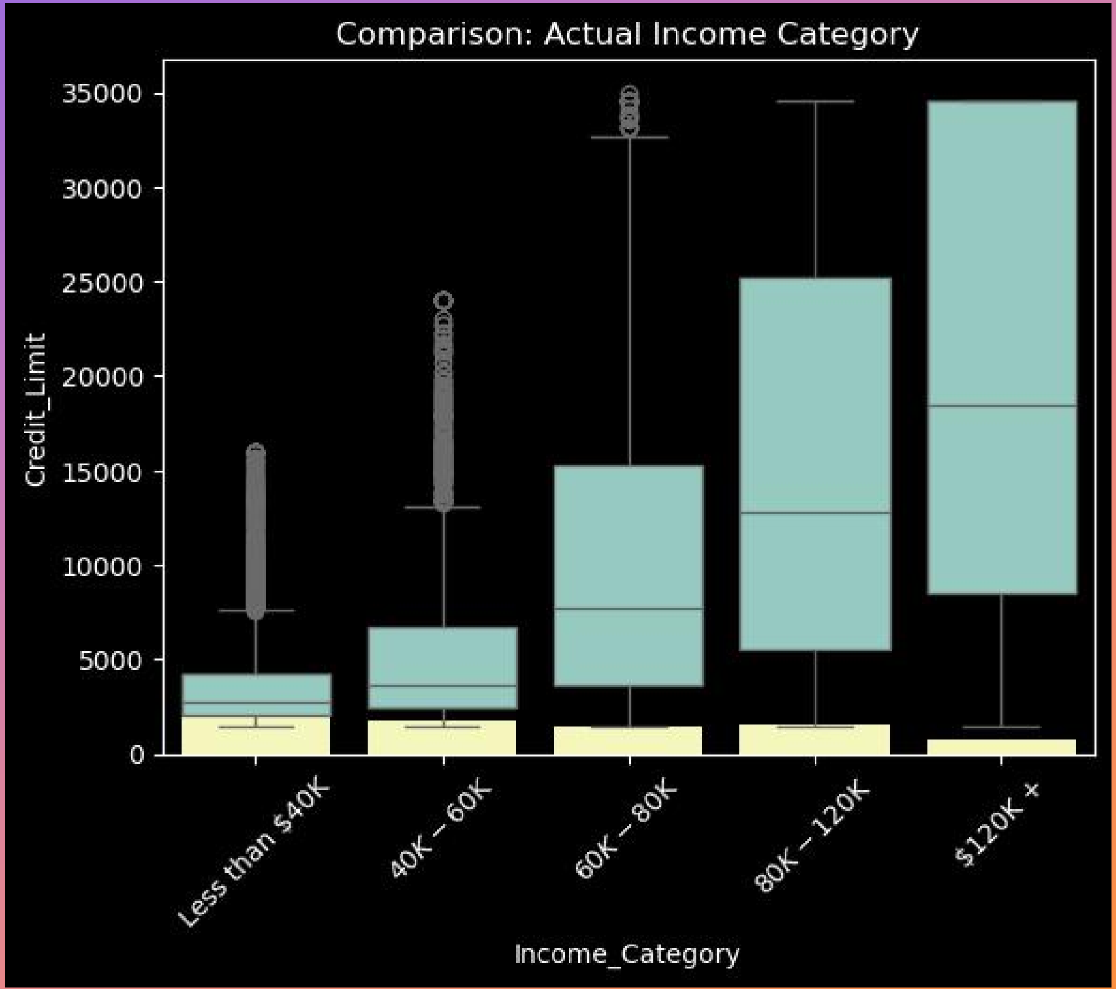
Hidden Missing Value

```python
# Identify categorical columns
cat_cols = df.select_dtypes(include='category').columns

# Display unique values
for col in cat_cols:
    print(f"\nUnique values in '{col}':")
    print(df[col].unique())
```

```
Unique values in 'Attrition_Flag':
['Attrited Customer', 'Existing Customer']
Categories (2, object): ['Attrited Customer', 'Existing Customer']

Unique values in 'Education_Level':
['Undergraduate', 'Graduate', 'Unknown', 'Doctorate', 'Uneducated', 'High School', 'Post-Graduate', 'College']
Categories (8, object): ['College', 'Doctorate', 'Graduate', 'High School', 'Post-Graduate', 'Undergraduate', 'Uneducated', 'Unknown']

Unique values in 'Marital_Status':
['Single', 'Divorced', 'Married', 'Unknown']
Categories (4, object): ['Divorced', 'Married', 'Single', 'Unknown']

Unique values in 'Income_Category':
['$60K - $80K', '$80K - $120K', '$120K +', 'Unknown', '$40K - $60K', 'Less than $40K']
Categories (6, object): ['$120K +', '$40K - $60K', '$60K - $80K', '$80K - $120K', 'Less than $40K', 'Unknown']

Unique values in 'Card_Category':
['Gold', 'Blue', 'Silver', 'Platinum']
Categories (4, object): ['Blue', 'Gold', 'Platinum', 'Silver']

Unique values in 'Attrition_Type':
['Involuntary', 'Existing Customer', 'Voluntary']
Categories (3, object): ['Existing Customer', 'Involuntary', 'Voluntary']
```

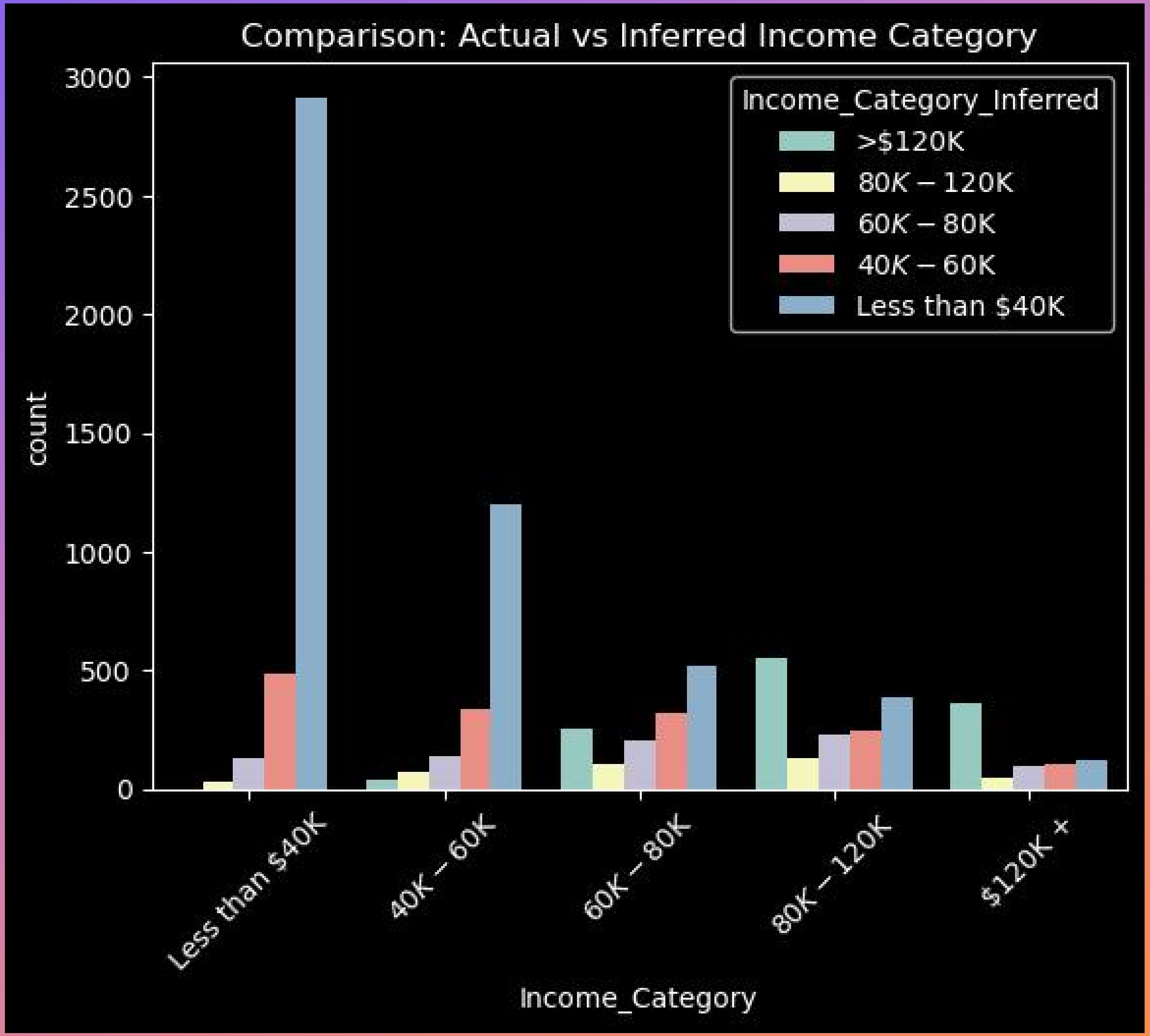
How to fill up missing Income_Category?





Missing Income_Categoary data is filled up according to Credit_Limit

```python
# Create Income Buckets Based on Credit Limit
def infer_income_category(row):
    if row['Credit_Limit'] >= 19000:
        return '>$120K'
    elif row['Credit_Limit'] >= 15000:
        return '$80K - $120K'
    elif row['Credit_Limit'] >= 10000:
        return '$60K - $80K'
    elif row['Credit_Limit'] >= 5462:
        return '$40K - $60K'
    else:
        return 'Less than $40K'
```



Data Cleansing / Quality Check in Progress (3)

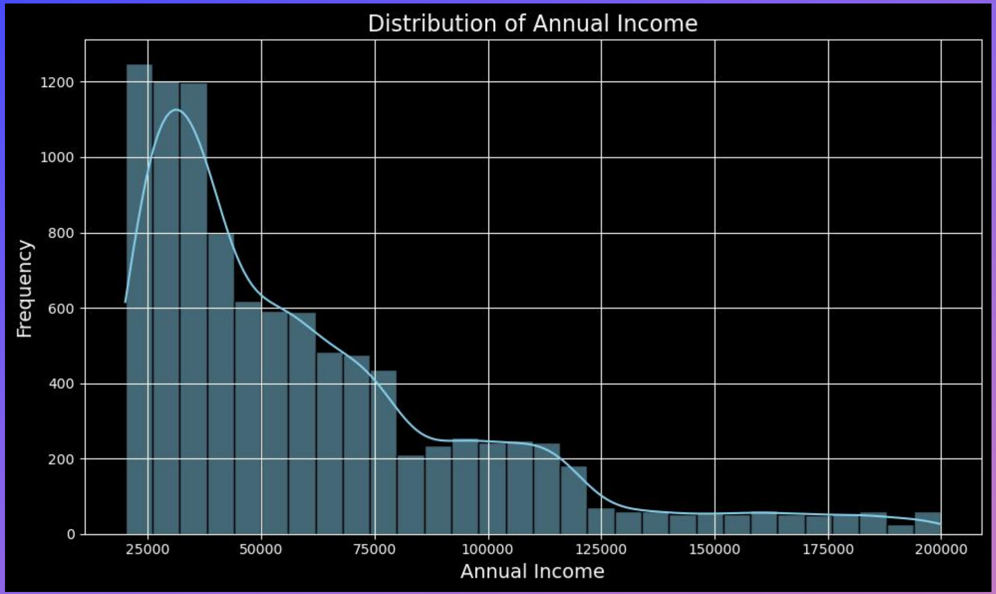
Normality Check for **Numerical** Variables

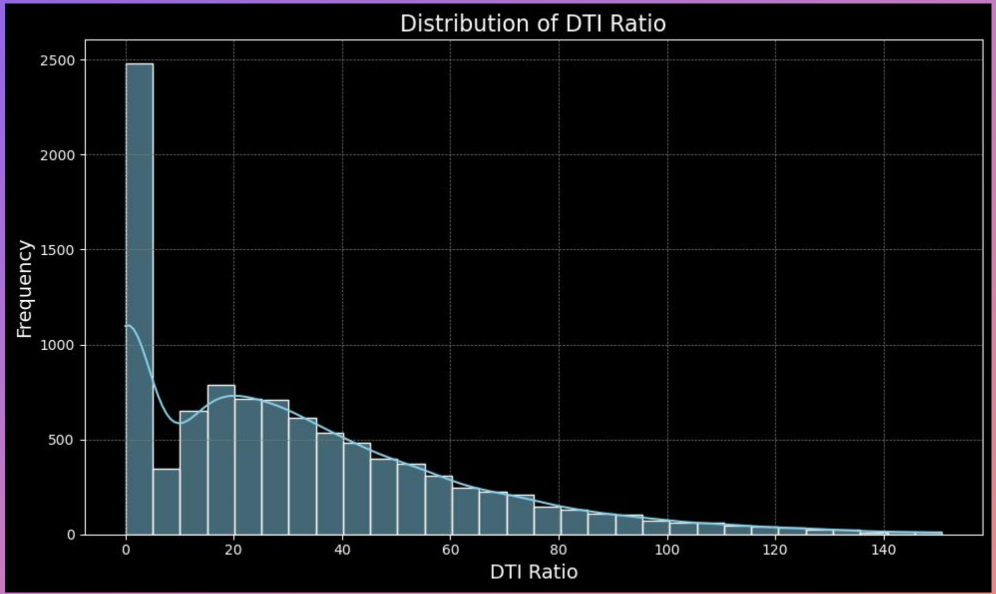
```
import pandas as pd
from scipy.stats import normaltest

# Select numeric columns from your DataFrame
numeric_cols = df.select_dtypes(include=['number']).columns

# List to store results
normality_results = []

# Run the normality test on each column
for col in numeric_cols:
    data = df[col].dropna()
    if len(data) >= 8:  # D'Agostino
        stat, p =
        nor                        else 'No'
        normality_results.append({
            'Column': col,
            'K2_Statistic': round(stat, 4),
            'p_value': round(p, 4),
            'Normally_Distributed': normal,
            'Sample_Size': len(data)
        })
    else:
        normality_results.append({
            'Column': col,
            'K2_Statistic': None,
            'p_value': None,
            'Normally_Distributed': 'N/A (too few samples)',
```

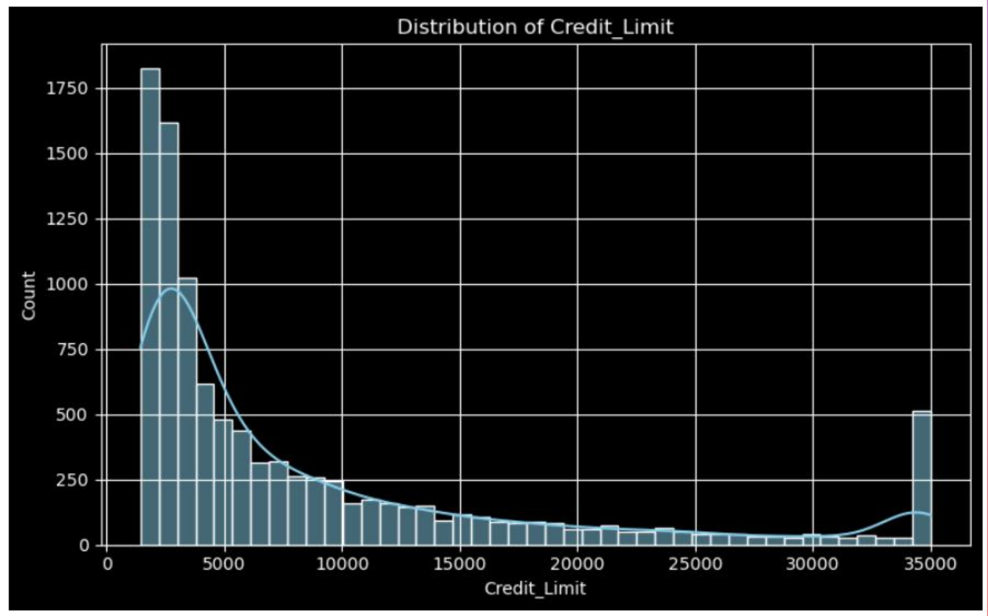
All, except one variable, are not normally distributed.

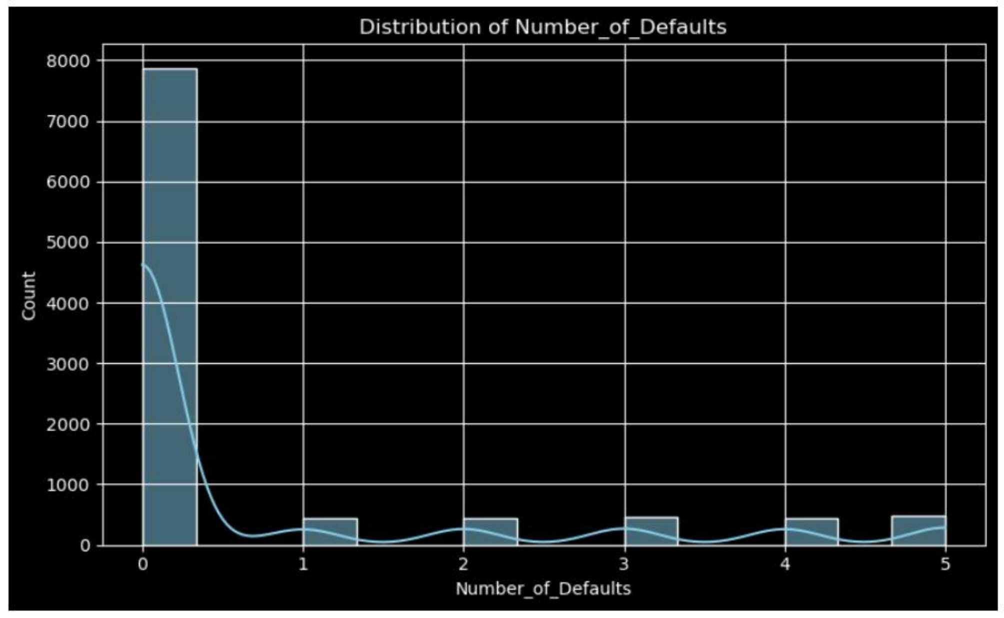
	Column	K2_Statistic	p_value	Normally_Distributed
0	Customer_Age	49.2823	0.000	No
1	Dependent_count	413.9960	0.000	No
2	Months_on_book	49.7030	0.000	No
3	Total_Relationship_Count	2159.6239	0.000	No
4	Months_Inactive_12_mon	1055.1569	0.000	No
5	Contacts_Count_12_mon	0.2716	0.873	Yes
6	Credit_Limit	2568.7141	0.000	No
7	Total_Revolving_Bal	4577.3041	0.000	No
8	Avg_Open_To_Buy	2561.9669	0.000	No
	al_Amt_Chng_Q4_Q1	3771.6388	0.000	No
	Total_Trans_Amt	3437.5569	0.000	No
11	Total_Trans_Ct	130.6147	0.000	No
12	Total_Ct_Chng_Q4_Q1	4624.0271	0.000	No
13	Avg_Utilization_Ratio	1632.7250	0.000	No
14	NB_Attrition_Prob_Yes	2732.4566	0.000	No
15	NB_Attrition_Prob_No	2732.4566	0.000	No
16	annual_income	1983.2656	0.000	No
17	monthly_income	1983.2656	0.000	No
18	DTI_ratio	1434.7890	0.000	No
19	Credit_Score	1018.9534	0.000	No
20	Loan_Default_Flag	1606.5908	0.000	No
21	Number_of_Defaults	3081.6887	0.000	No
22	Default_Amount	3526.4534	0.000	No
23	Bankruptcy_Flag	4991.7803	0.000	No
24	Adjusted_Credit_Score	399.5646	0.000	No
25	estimated_annual_revenue	4577.3041	0.000	No

Normality Check – Bar Chart









DEBT-TO-INCOME

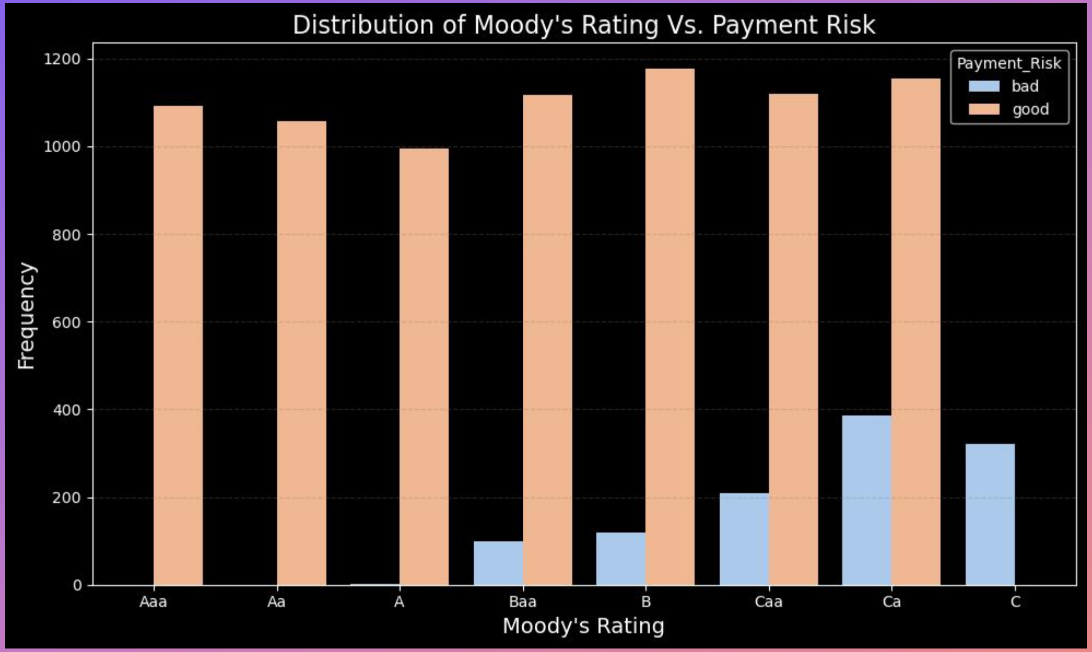
DTI RATIO=

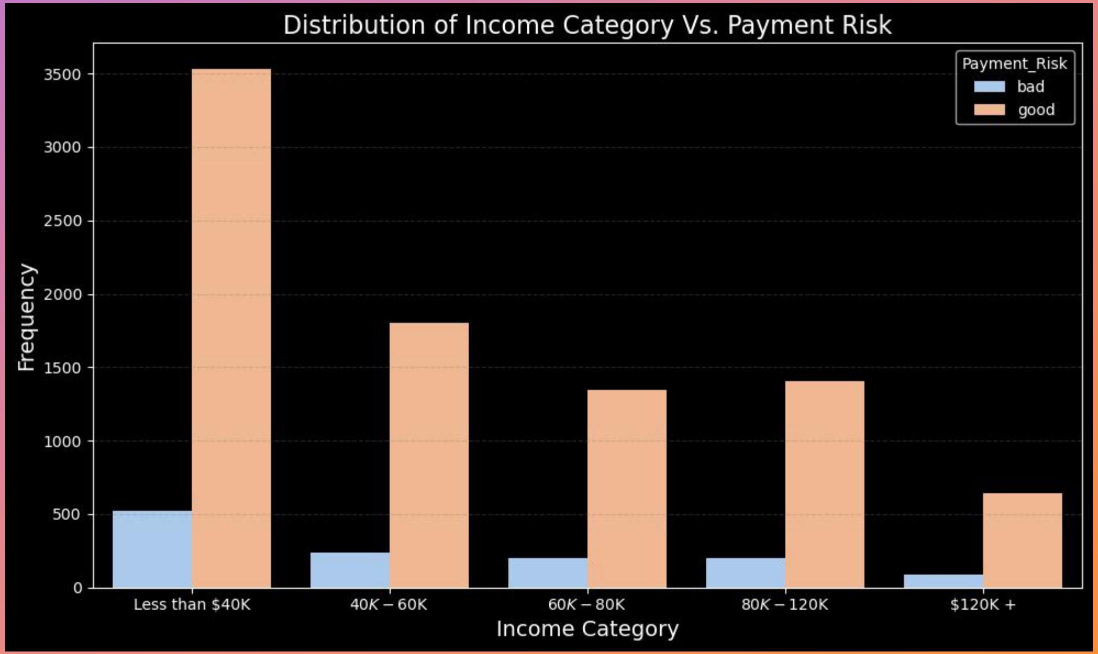
(TOTAL MONTHLY DEBT PAYMENTS / GROSS MONTHLY INCOME)×100

DTI Range	Interpretation
0%–20%	Excellent
21%–36%	Good
37%–49%	Risky
50%+	High Risk

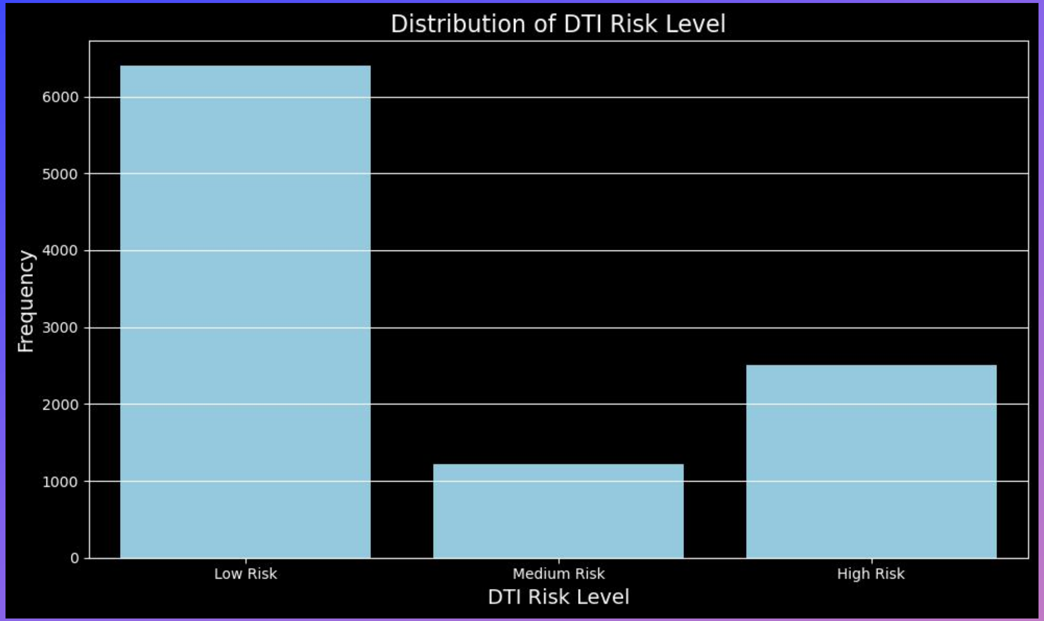
Categorical Data Insights (1)

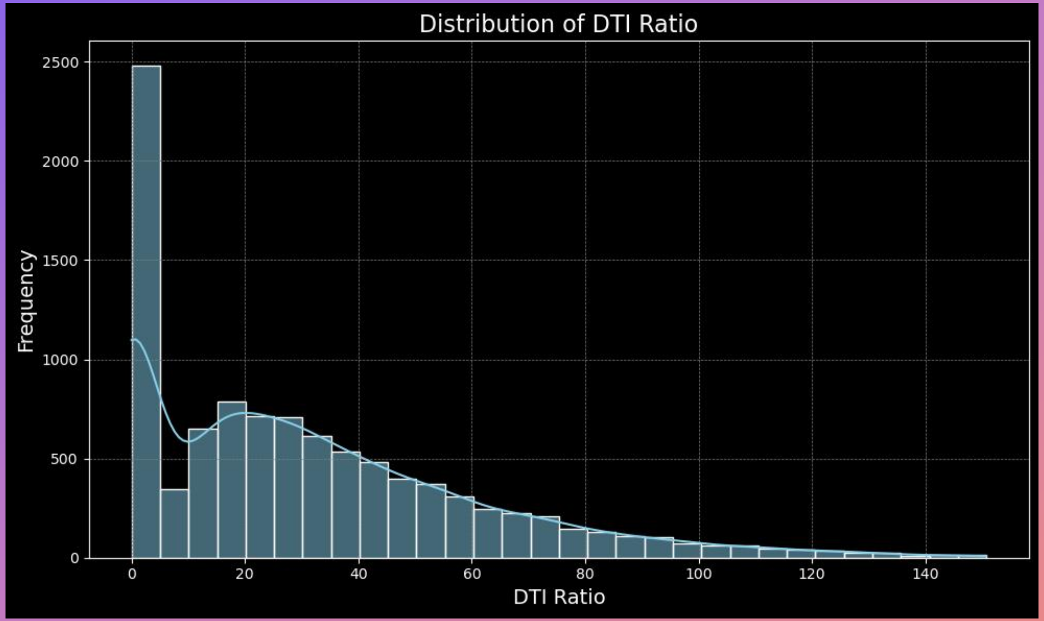
```python
cat_cols = ['Income_Category_Complete', 'DTI_Risk_Level','Moodys_Rating','Payment_History_Flag', 'Bankruptcy_Flag',]
# Set black background style
plt.style.use('dark_background')

for col in cat_cols:
    plt.figure(figsize=(10, 5))
    sns.countplot(x=col, hue='Payment_Risk', data=df)
    plt.title(f'{col} vs Payment Risk Level')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

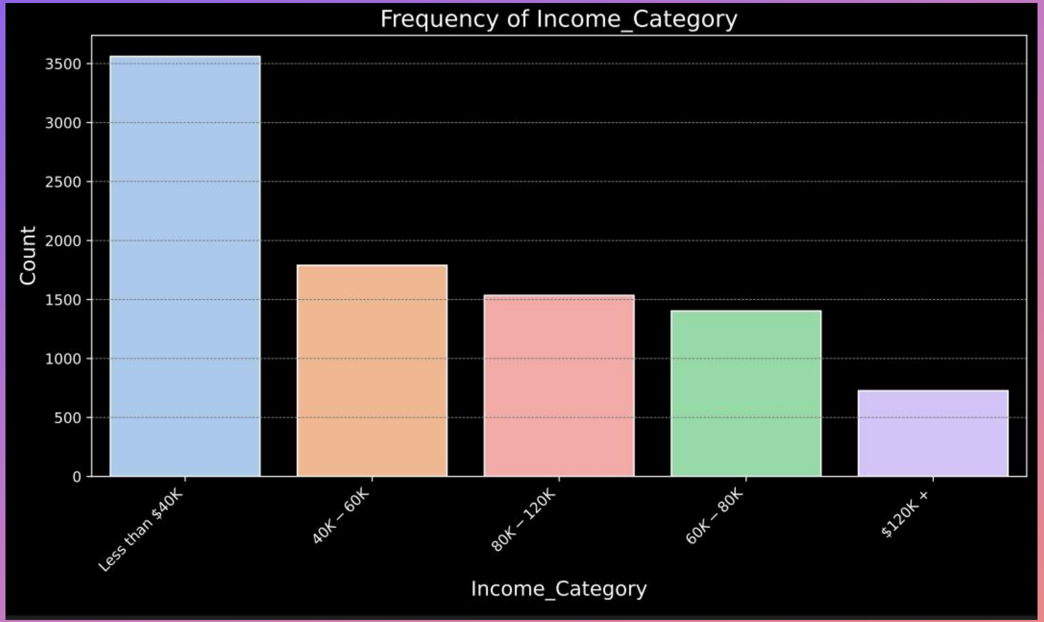


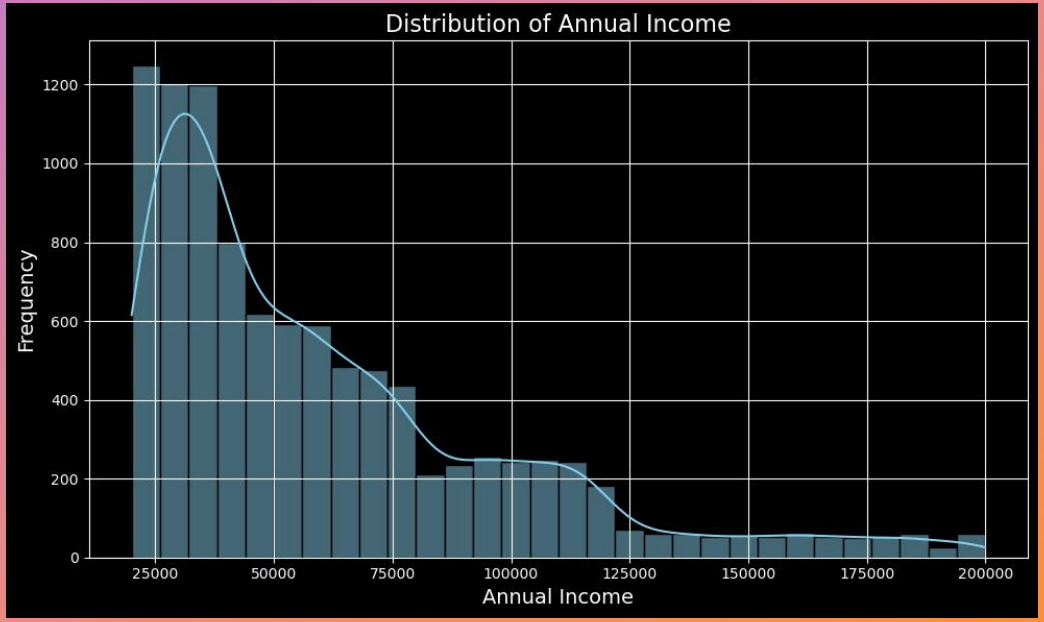


Categorical Data Insights (2)

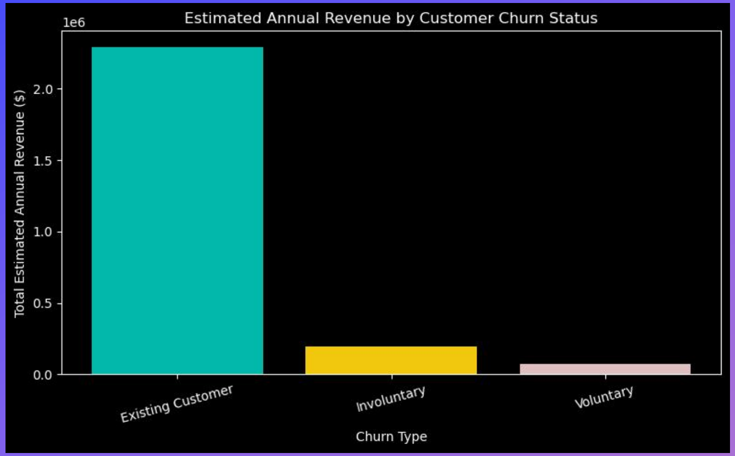


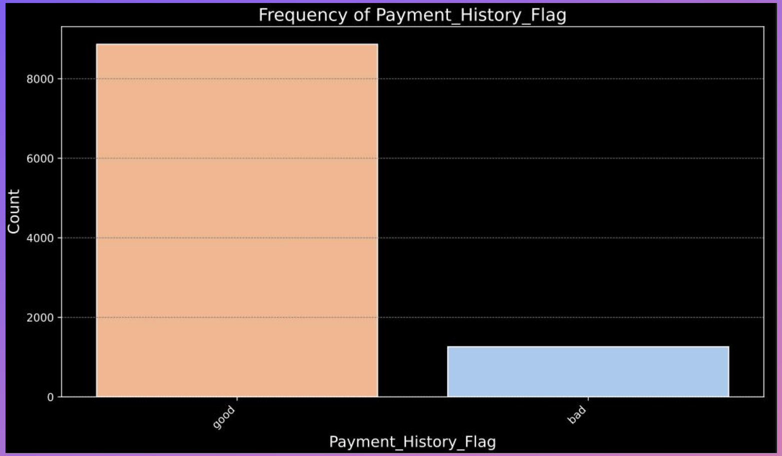


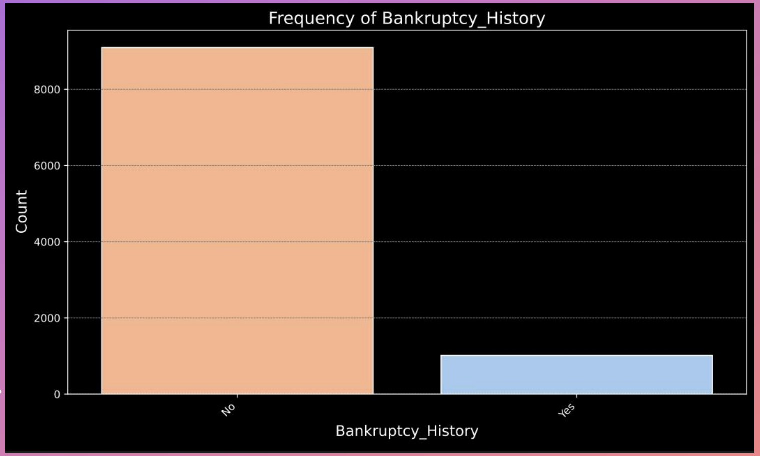


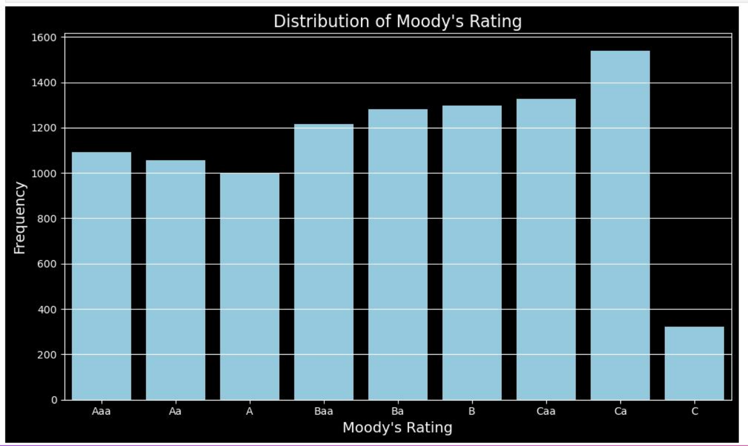


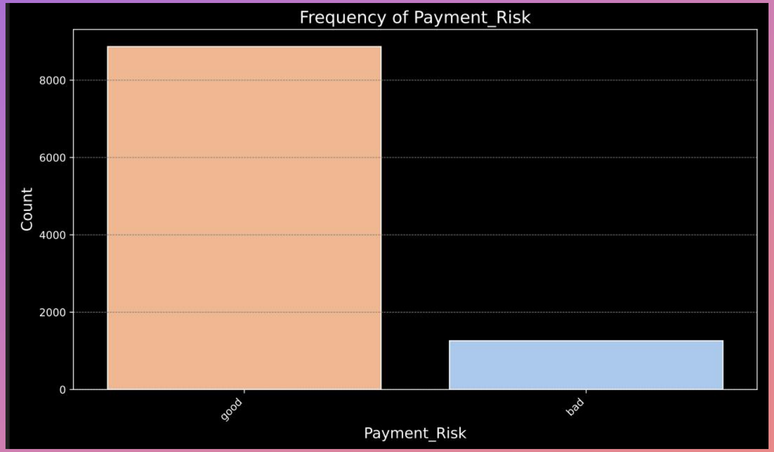
Categorical Data Insights (3)

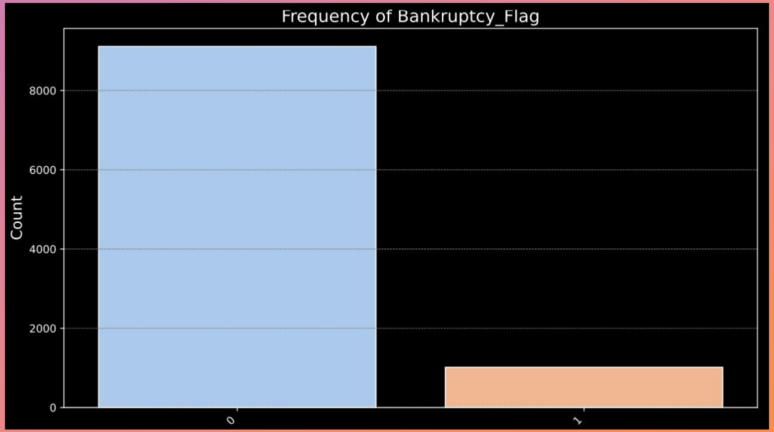




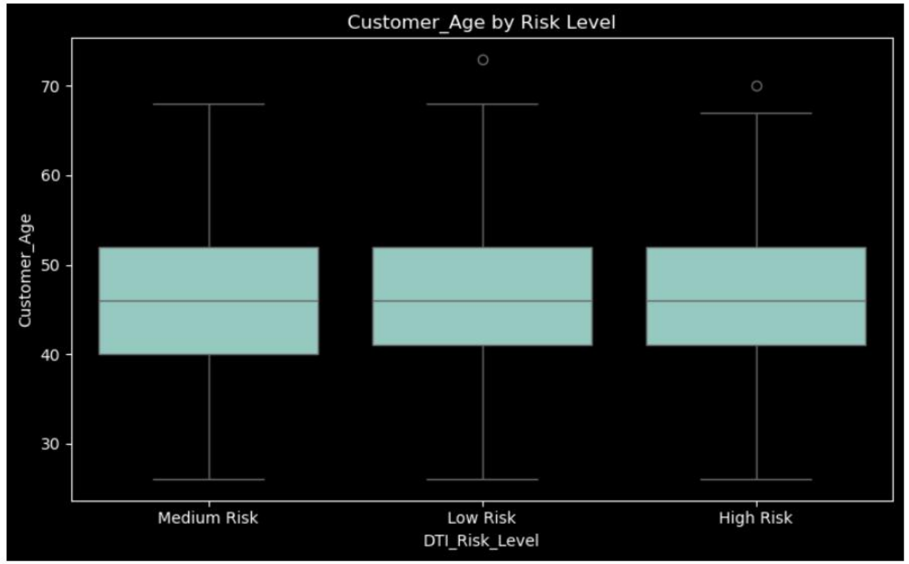


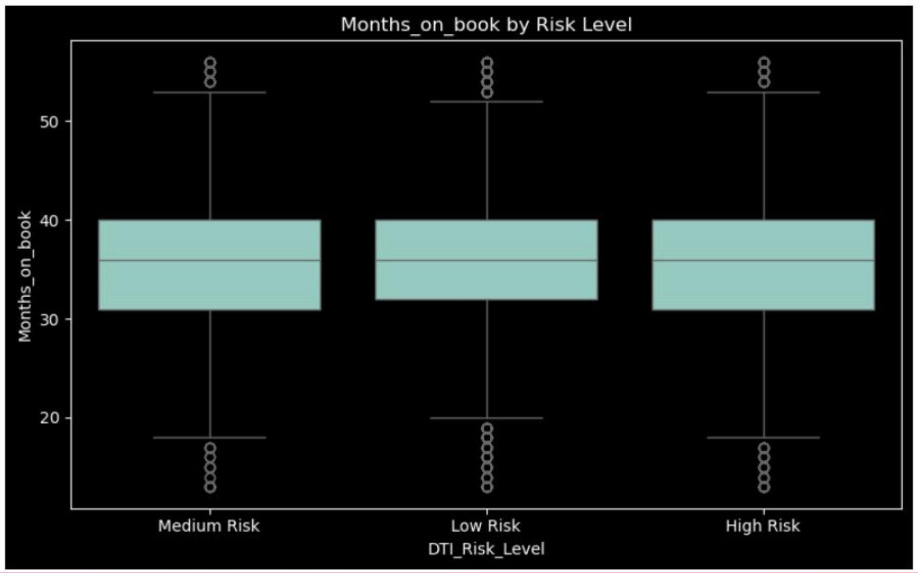


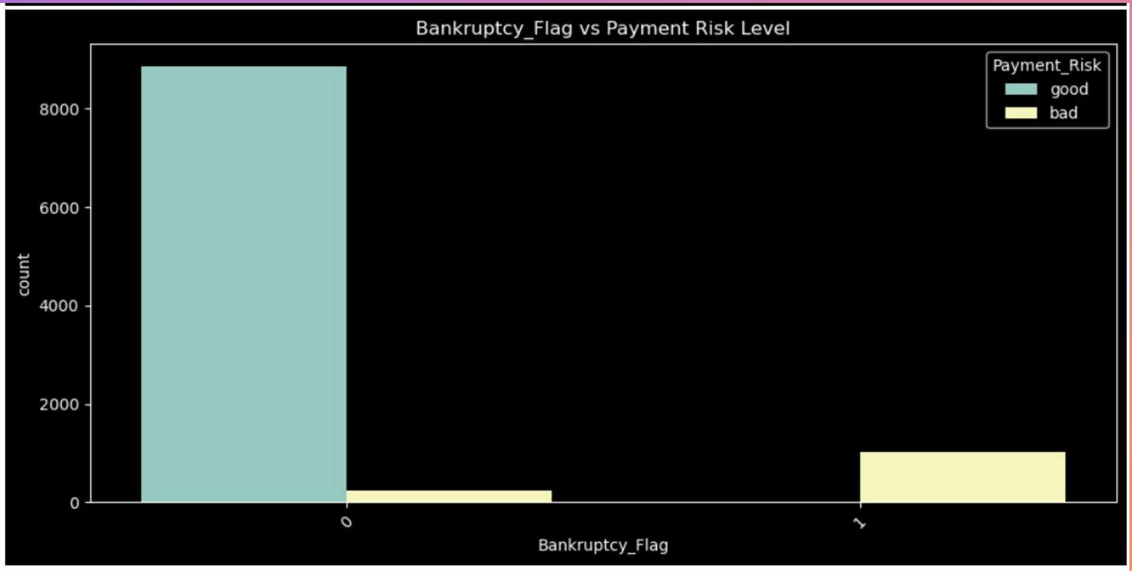




Categorical Data Insights (4)

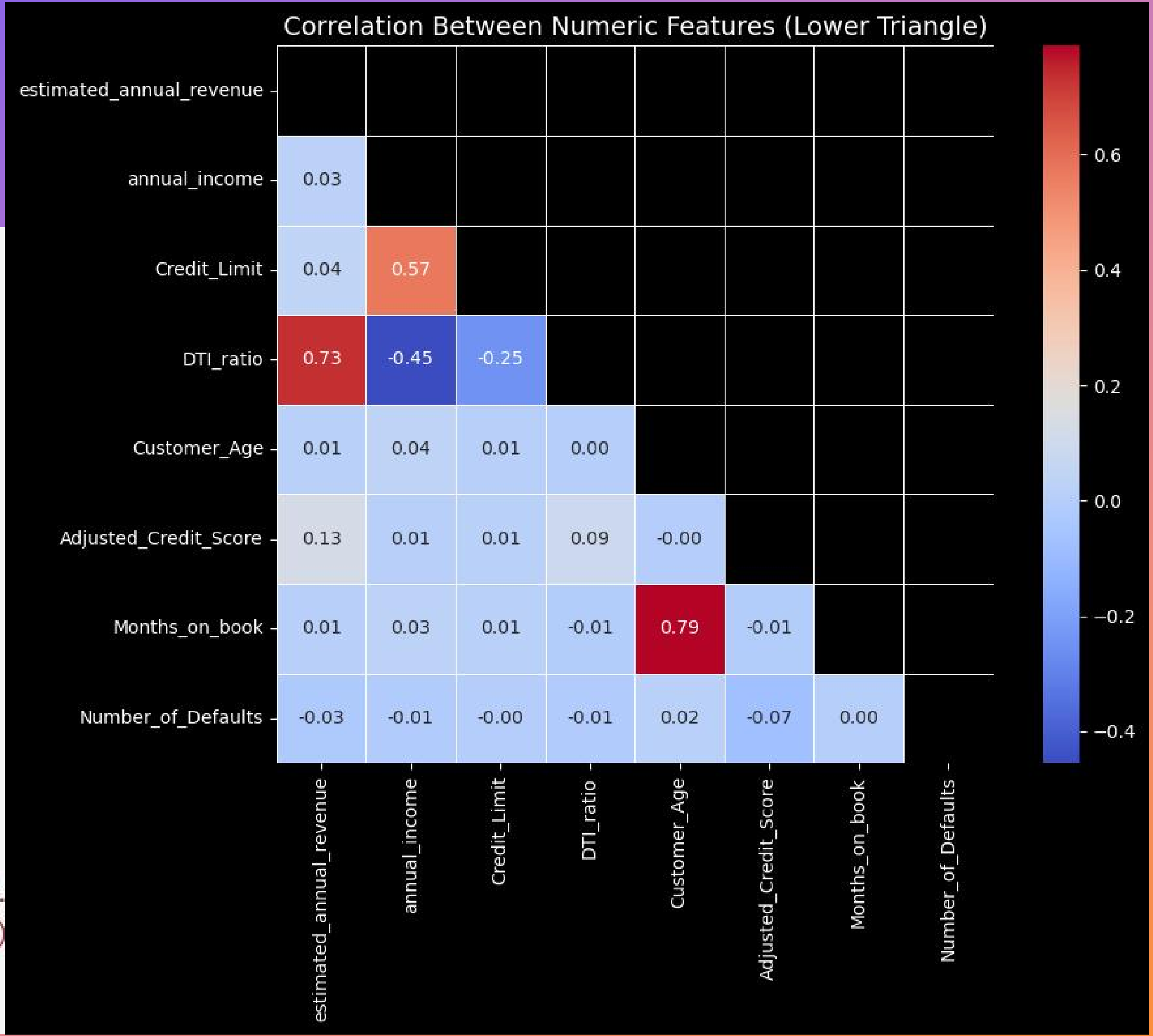






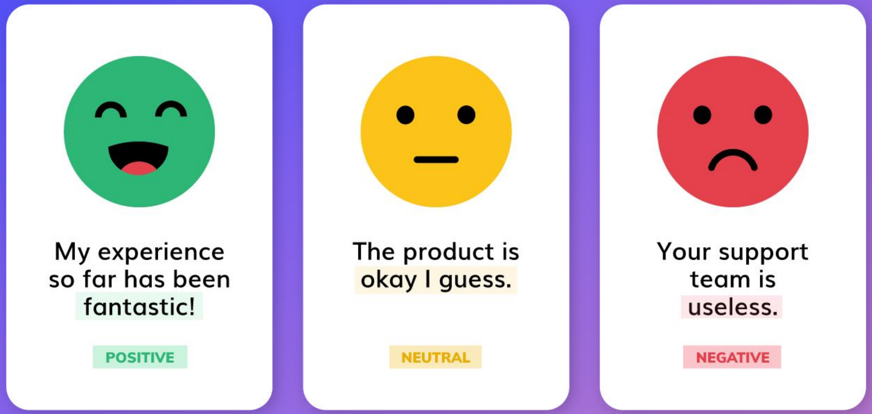
Correlation Heatmap

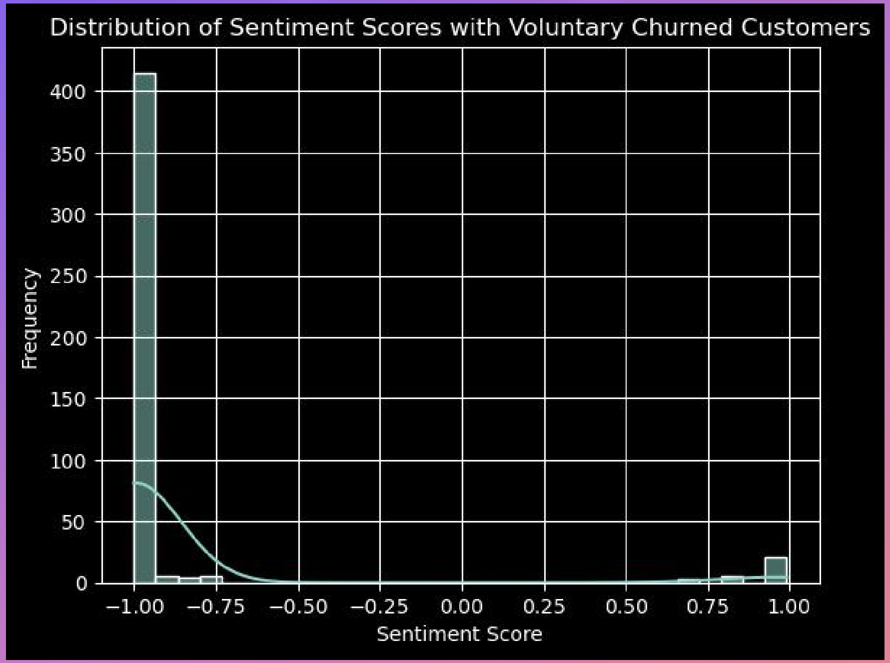
```
# 5. Correlation Heatmap
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Calculate the correlation matrix
corr = df[num_cols].corr()

# Create a mask for the upper triangle
mask = np.triu(np.ones_like(corr, dtype=bool))

# Set the plot style
plt.style.use('dark_background')  # Optional: dark theme

# Plot the heatmap with the mask
plt.figure(figsize=(10, 8))
sns.heatmap(corr, mask=mask, annot=True, cmap='coolwarm', fmt=".
plt.title('Correlation Between Numeric Features (Lower Triangle)
plt.tight_layout()
plt.show()
```

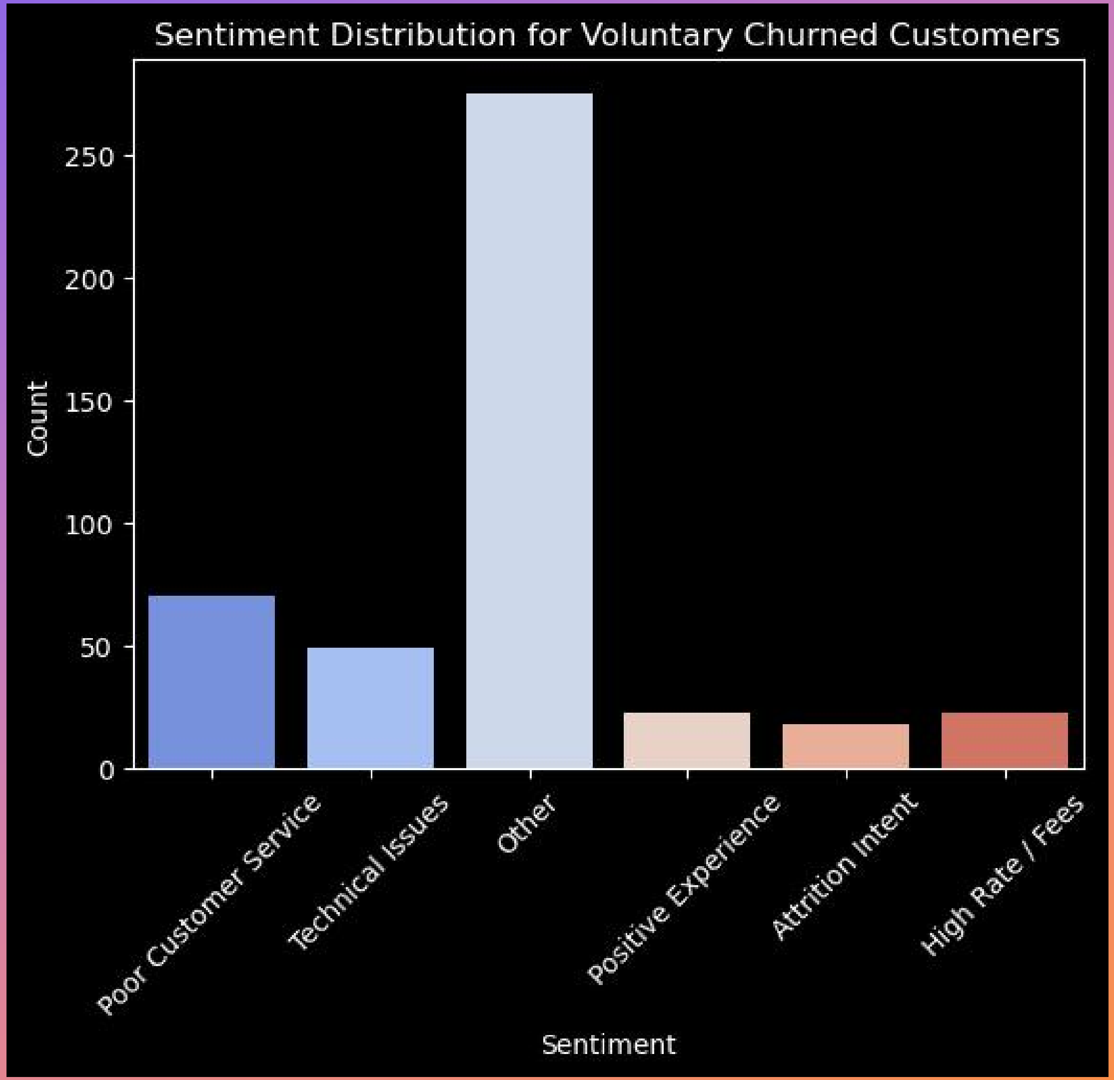


Voluntary Churn

Sentiment Analysis – know why to help us grow





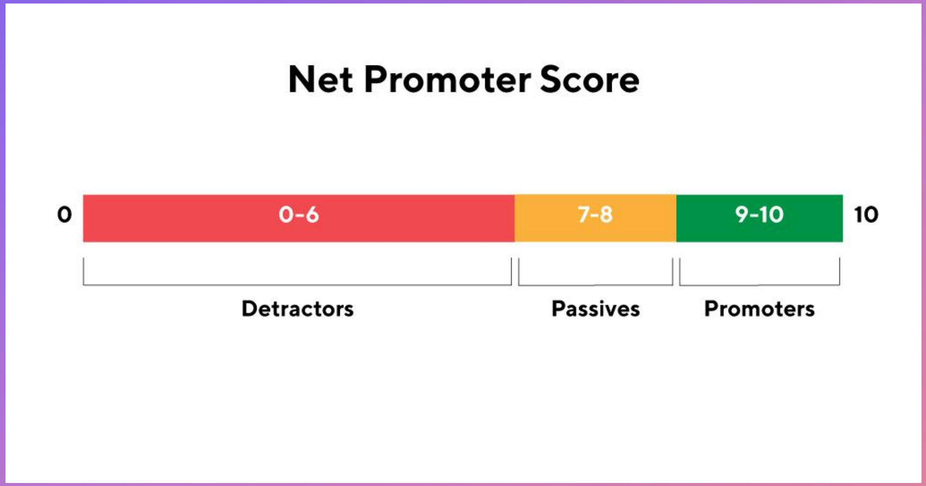


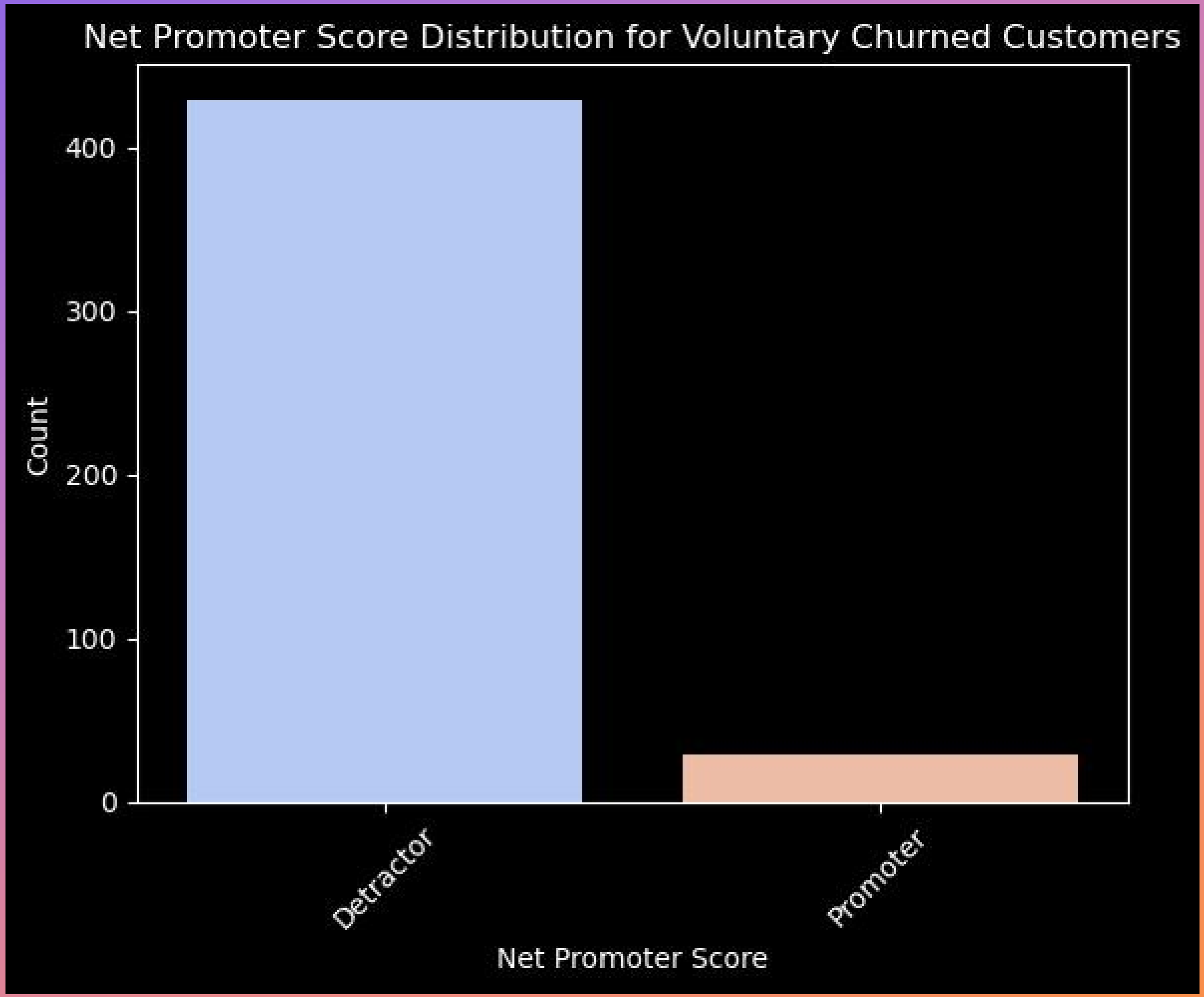
Voluntary Churn

NPS Analysis

(Net Promoter Score)

- Help us know our future



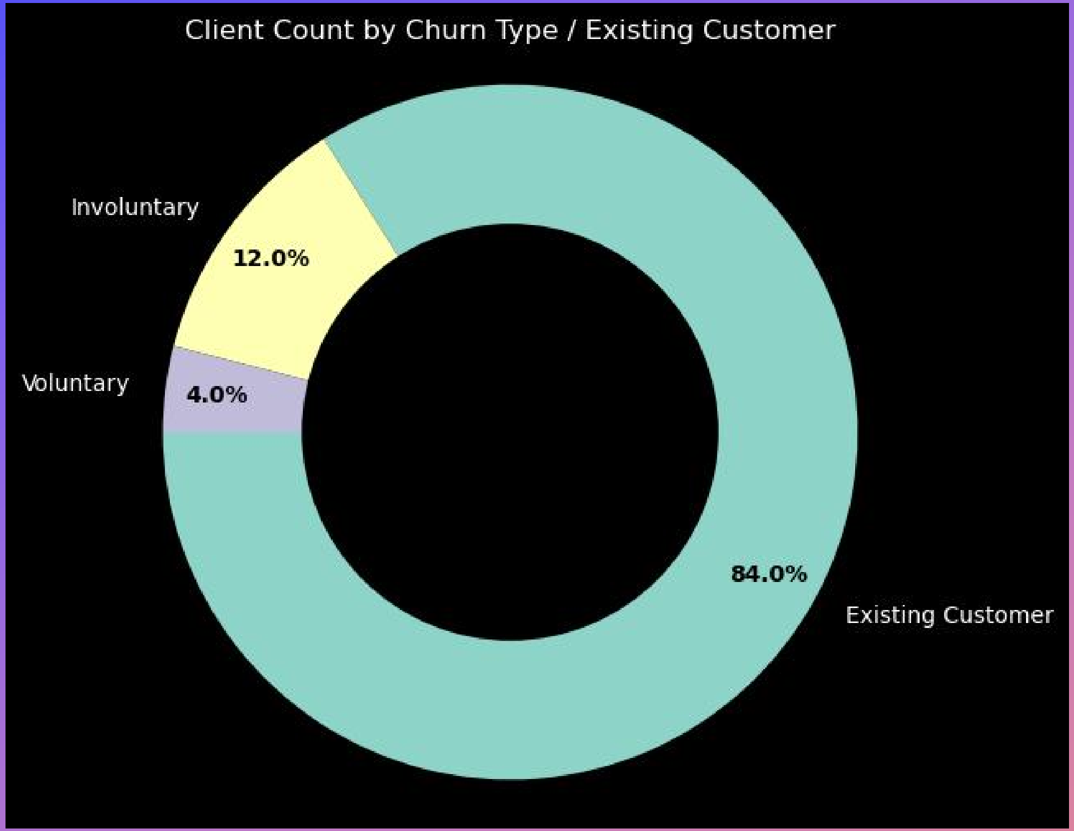


Sentiment and NPS Analysis for Voluntary Churn Customers

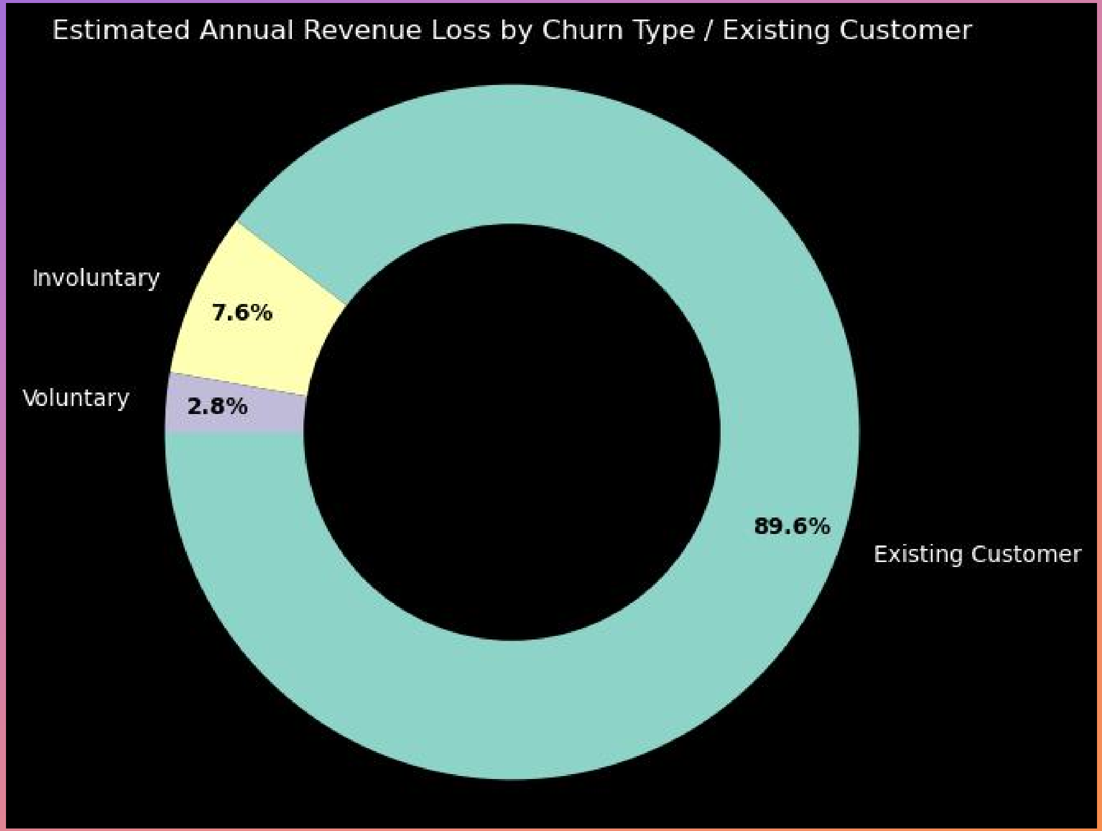
- By Hugging Face

CLIENTNUM	review	sentiment_label	sentiment_score	sentiment_numeric	nps_category	nps_summar	nps_value
771795933	Unable to make a payment from another credit card to close the balance because the card number is too long. I could only make a payment via DD, which I DON'T want to do. There is no information about this on their website. Can't wait to close this thing down. Customer Service - took me 10 min to go through security, clueless staff. Website appalling, no point using it as all managed by the app, which is painfully slow, not working most of the time and very limited.	NEGATIVE	1.000	-1.000	Detractor	93.67	0
713390433	always paid on time because I'm very organised. However yesterday I had an email off then saying they had started a credit plan and my first payment was coming out on 25th and it was for a very shocking ammount of money. I originally thought it was fraud so I phoned up argos to then find it was a purchase I made last year that I hadn't paid off. Having looked back through my bank statements it looks like this wasn't paid however I had paid other stuff off on there more recently and cleared the card so I would have noticed the outstanding payment and paid it off. I think argos didn't show that I had this left to pay until my year was up then automatically charged me. I have no proof of any of this though because they send no emails all I have is an email of me ordering it and then my bank statement to show I never paid it	NEGATIVE	0.998	-0.998	Detractor	93.67	0
721243083	I Spoke with them twice this week both times the call handler was unfriendly and unhelpful.	NEGATIVE	0.997	-0.997	Detractor	93.67	0
787921383	went on the phone talk to 3 different people at the customer service at Argos credit card about my card payment . All of them customer service is very poor Not freindly no helpful . 3 of them the same .	NEGATIVE	0.999	-0.999	Detractor	93.67	0
710047533	Customer service number not working properly can not make a payment.	NEGATIVE	1.000	-1.000	Detractor	93.67	0
789684258	They say buy now pay later with no interest, when you pay after the year they will claim huge amount of interest for that year, they will take the money without even telling you or sending you an email, when you ask they will say it is the interest for that year. They are very cunning beware.	NEGATIVE	0.957	-0.957	Detractor	93.67	0
716124408	Unable to make a payment from another credit card to close the balance because the card number is too long. I could only make a payment via DD, which I DON'T want to do. There is no information about this on their website. Can't wait to close this thing down. Customer Service - took me 10 min to go through security, clueless staff. Website appalling, no point using it as all managed by the app, which is painfully slow, not working most of the time and very limited.	NEGATIVE	1.000	-1.000	Detractor	93.67	0

Client Counts by Churn Type / Existing Customer

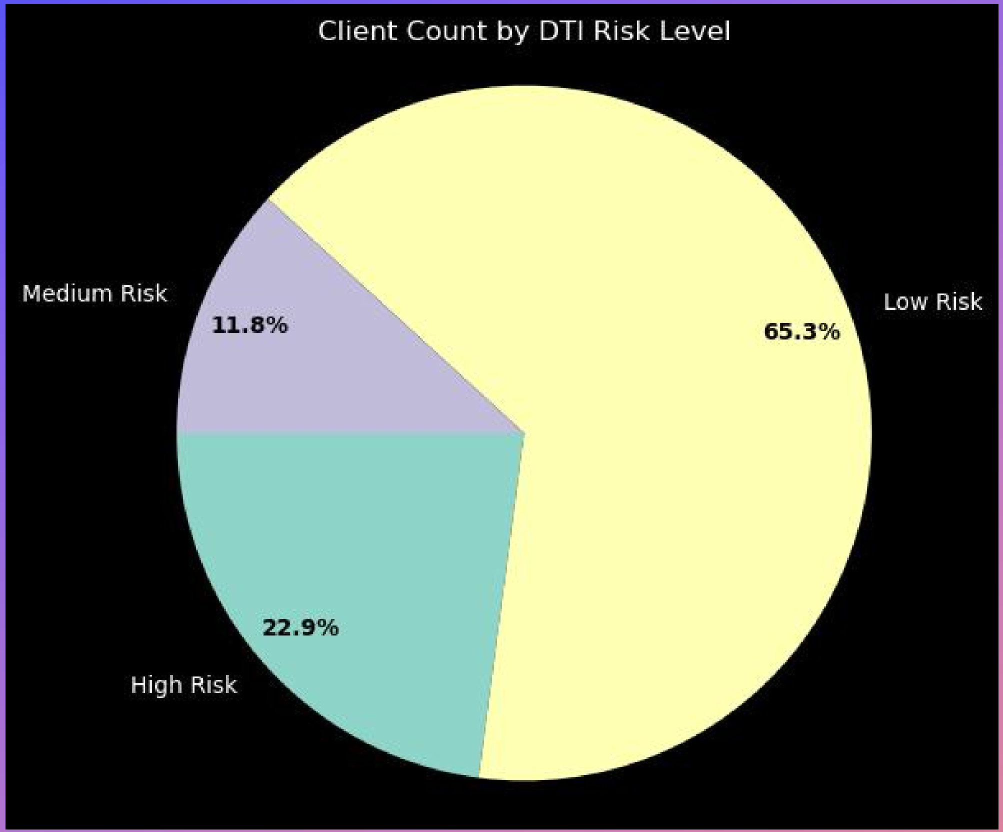


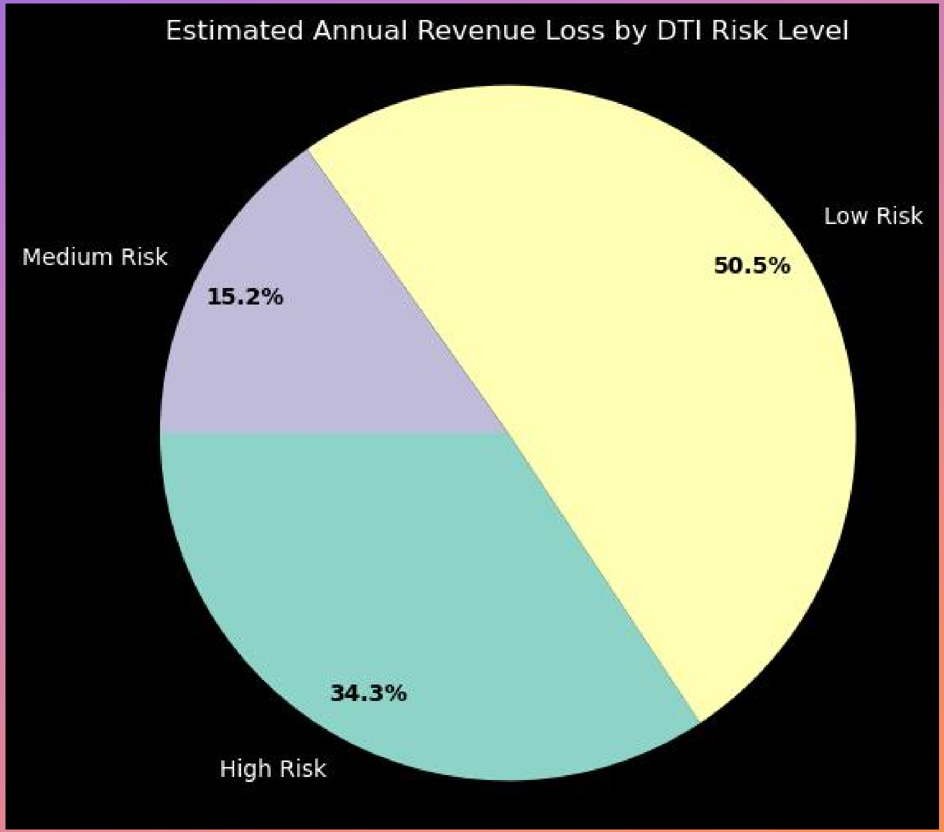
Estimated Annual Revenue by Churn Type / Existing Customer

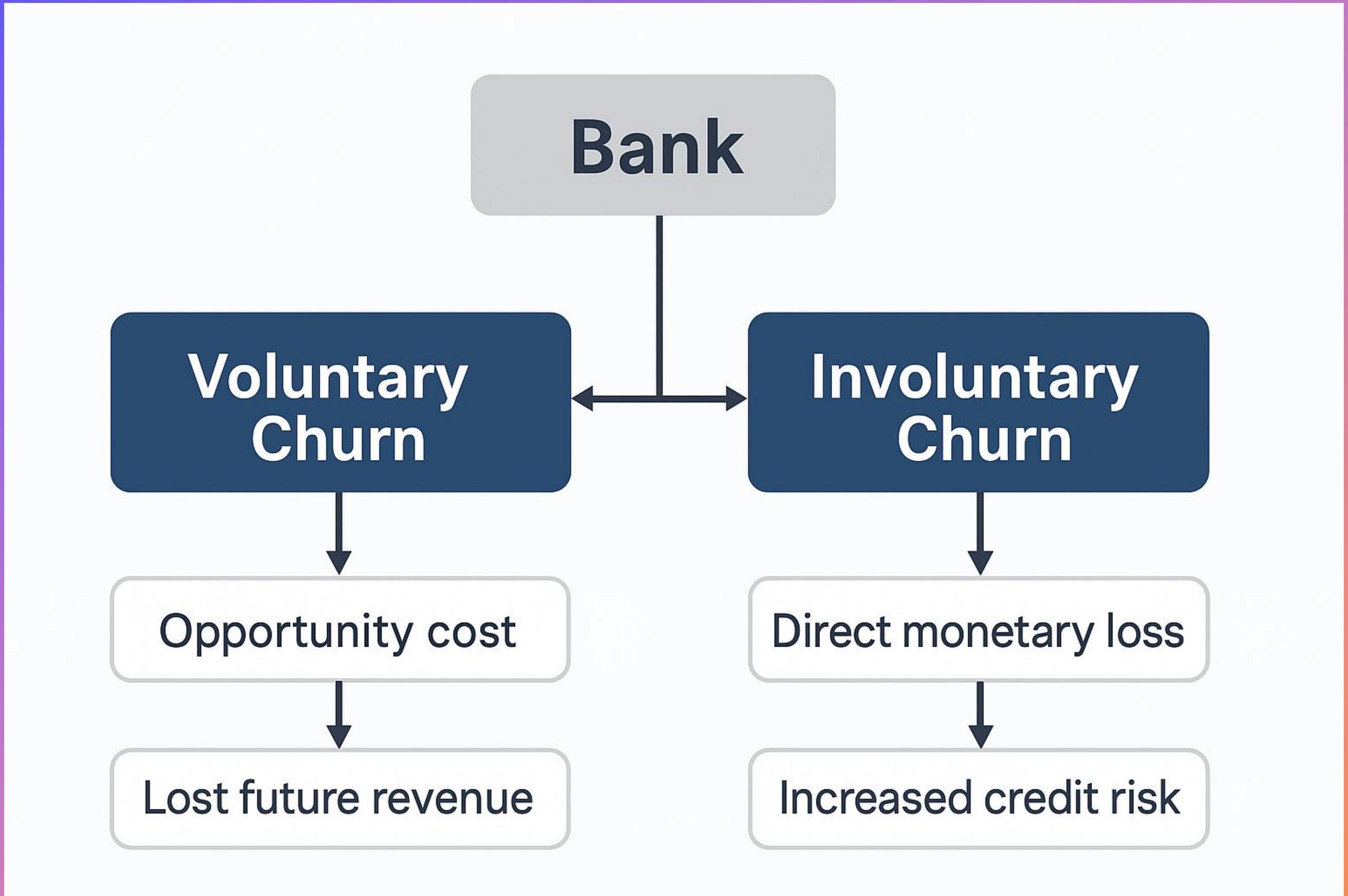


Client Counts by DTI Risk Level

Estimated Annual Revenue By DTI Risk Level

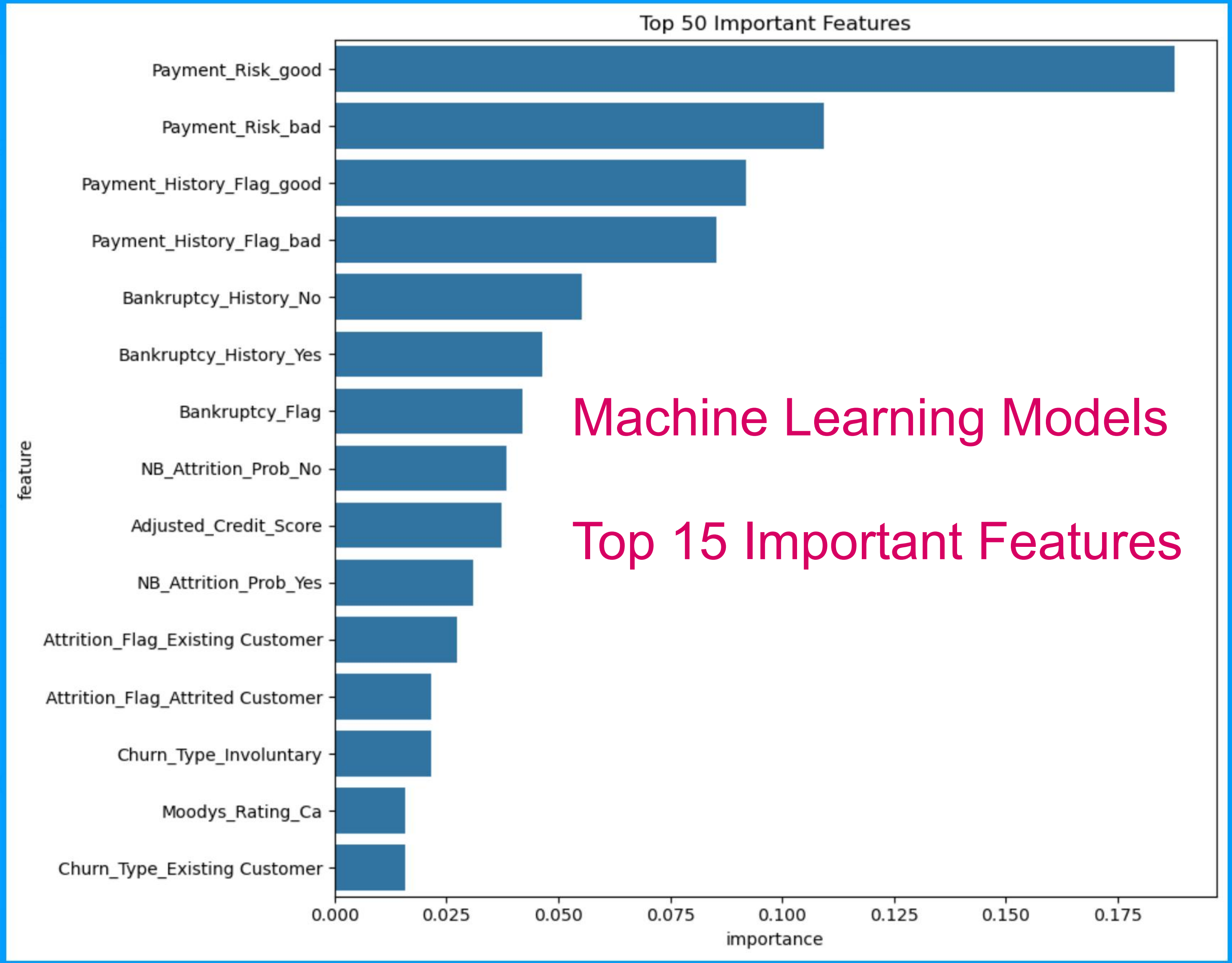





Bank
Voluntary Churn
Involuntary Churn
Opportunity cost
Direct monetary loss
Lost future revenue
Increased credit risk

Machine Learning Models
(Logistic Regression, XGBoost)

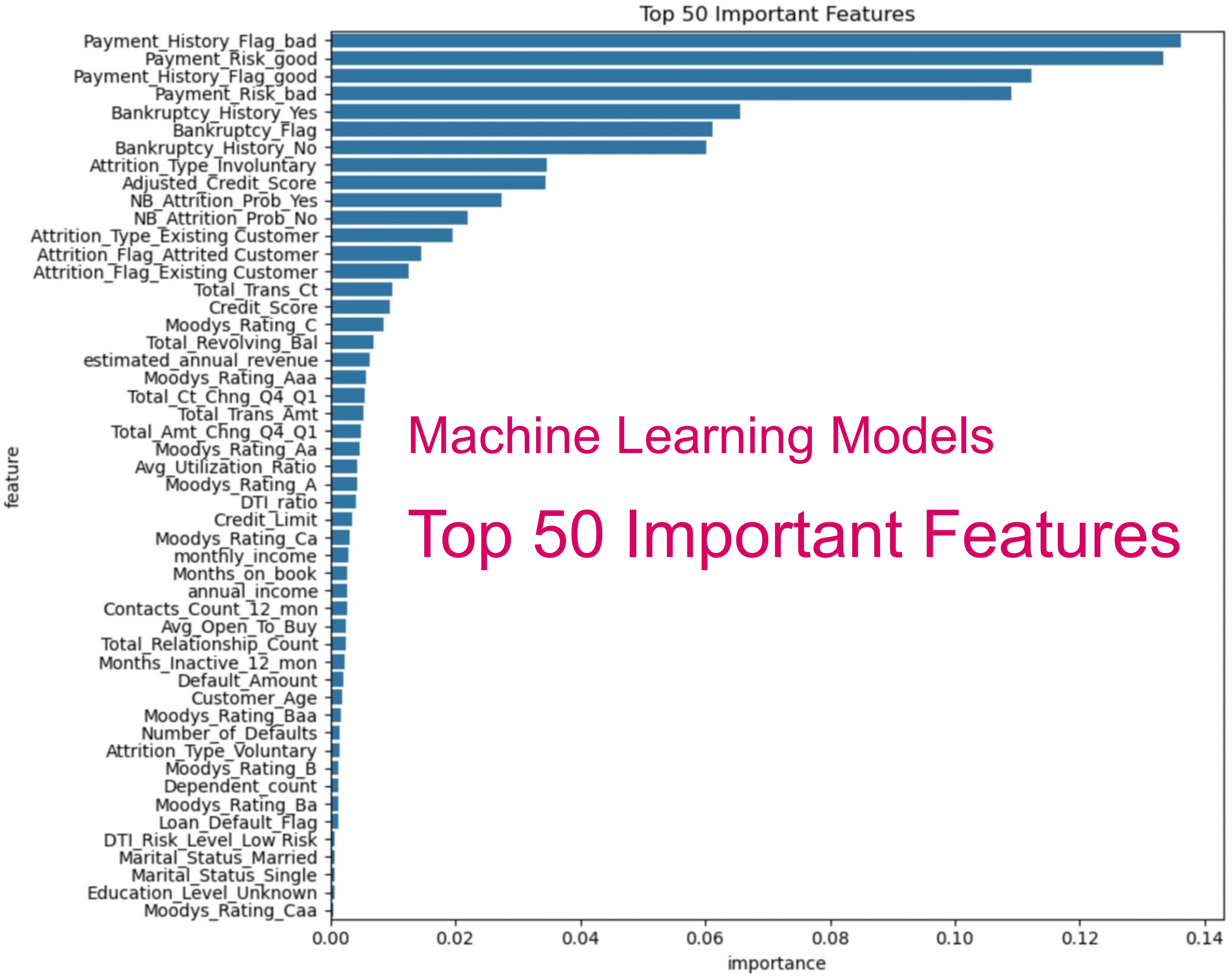
```
# Step 5: Feature Importance

# Get feature names after one-hot encodin
preprocessor.fit(X_train)
feature_names = (numerical_cols.tolist()
                 list(preprocessor.named_

# Get feature importances from the best m
importances = best_model.named_steps['cla

# Create a DataFrame for visualization
feature_importances = pd.DataFrame({'feat
feature_importances = feature_importances

# Plot top 20 features
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.barplot(x='importance', y='feature',
            data=feature_importances.head
plt.title('Top 50 Important Features')
plt.tight_layout()
plt.show()
```



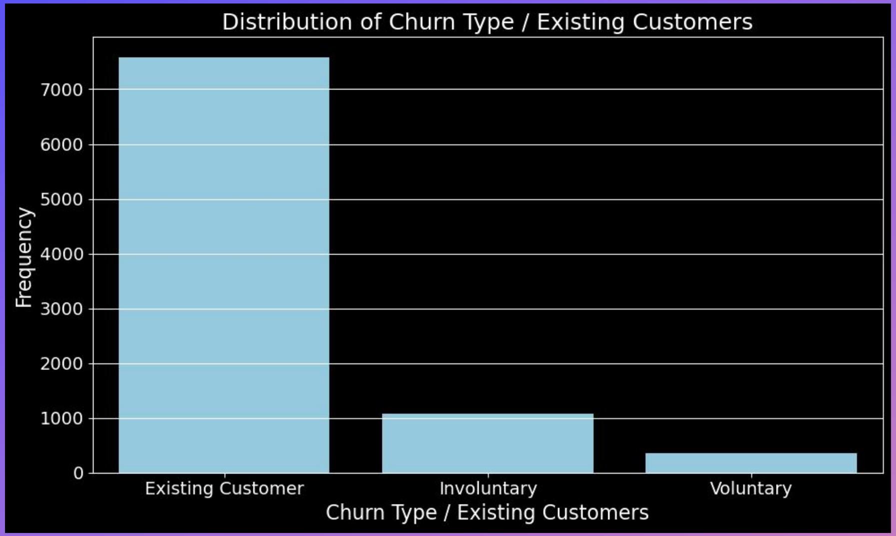
Machine Learning Models

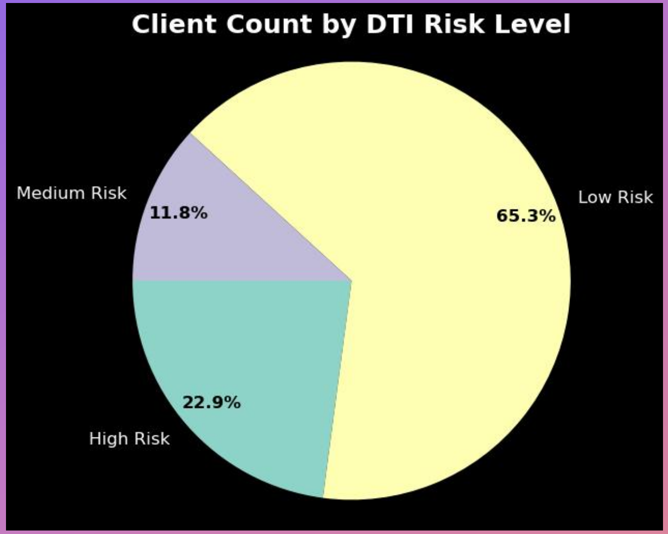
Top 50 Important Features

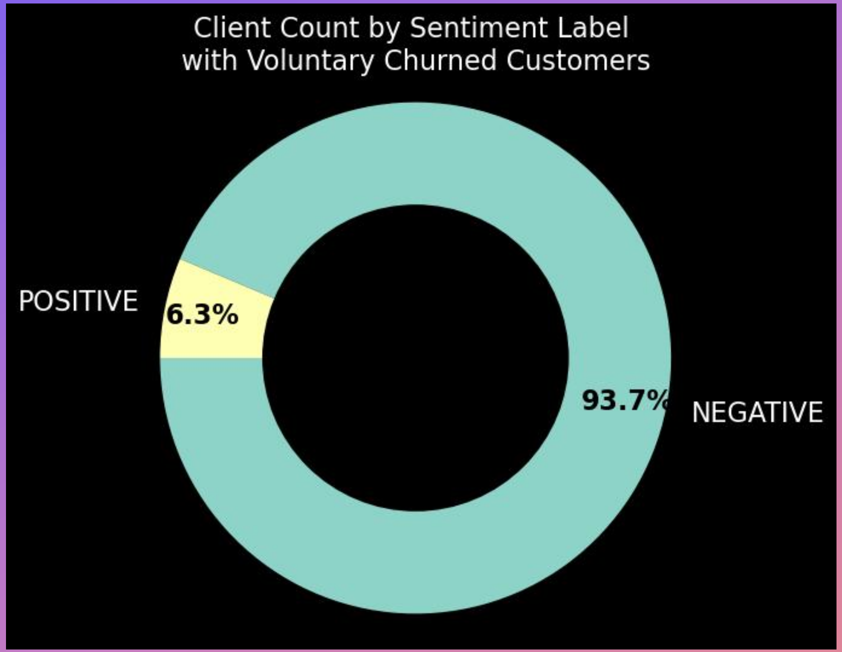
```
# Step 5: Feature Importance

# Get feature names after one-hot encoding
preprocessor.fit(X_train)
feature_names = (numerical_cols.tolist() +
                 list(preprocessor.named_tr

# Get feature importances from the best mod
importances = best_model.named_steps['class

# Create a DataFrame for visualization
feature_importances = pd.DataFrame({'featur
feature_importances = feature_importances.s

# Plot top 20 features
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.barplot(x='importance', y='feature',
            data=feature_importances.head(2
plt.title('Top 20 Important Features')
plt.tight_layout()
plt.show()
```

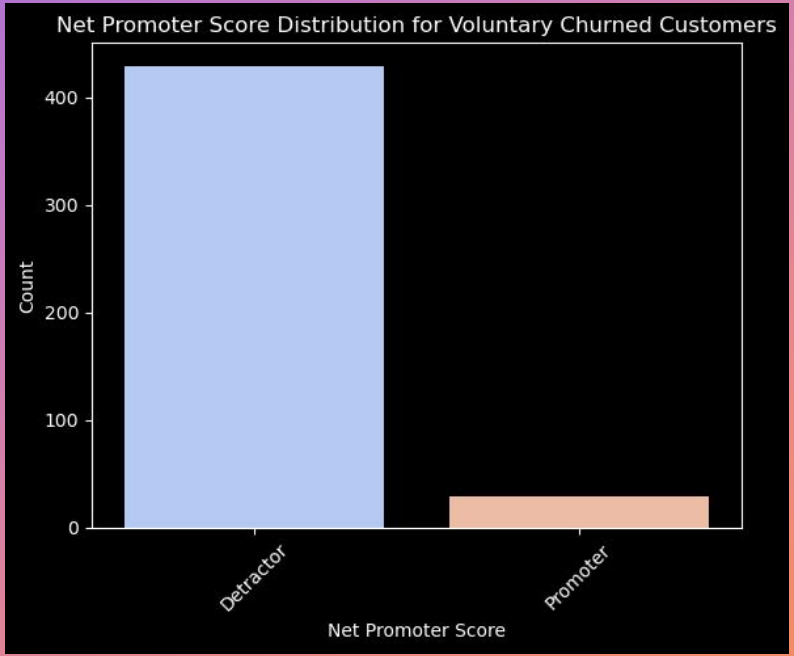


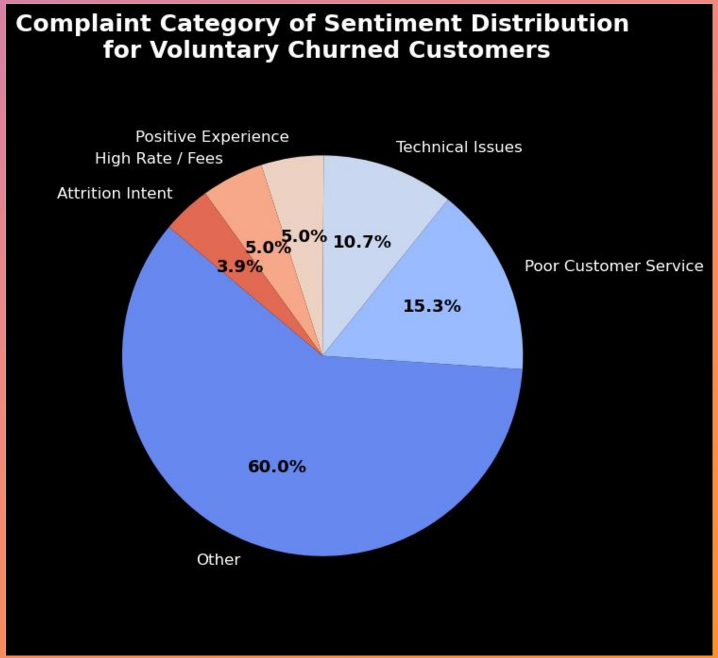
Executive Summary (1)





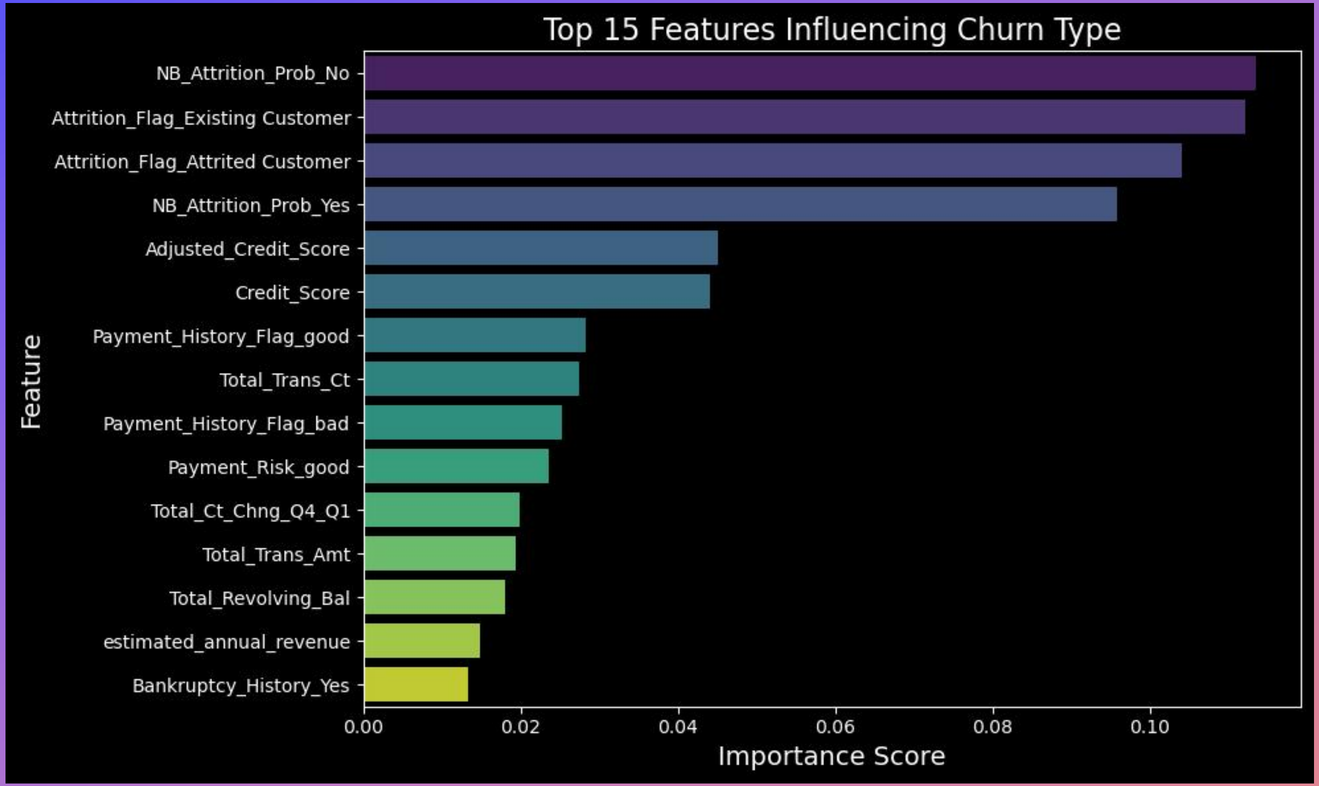


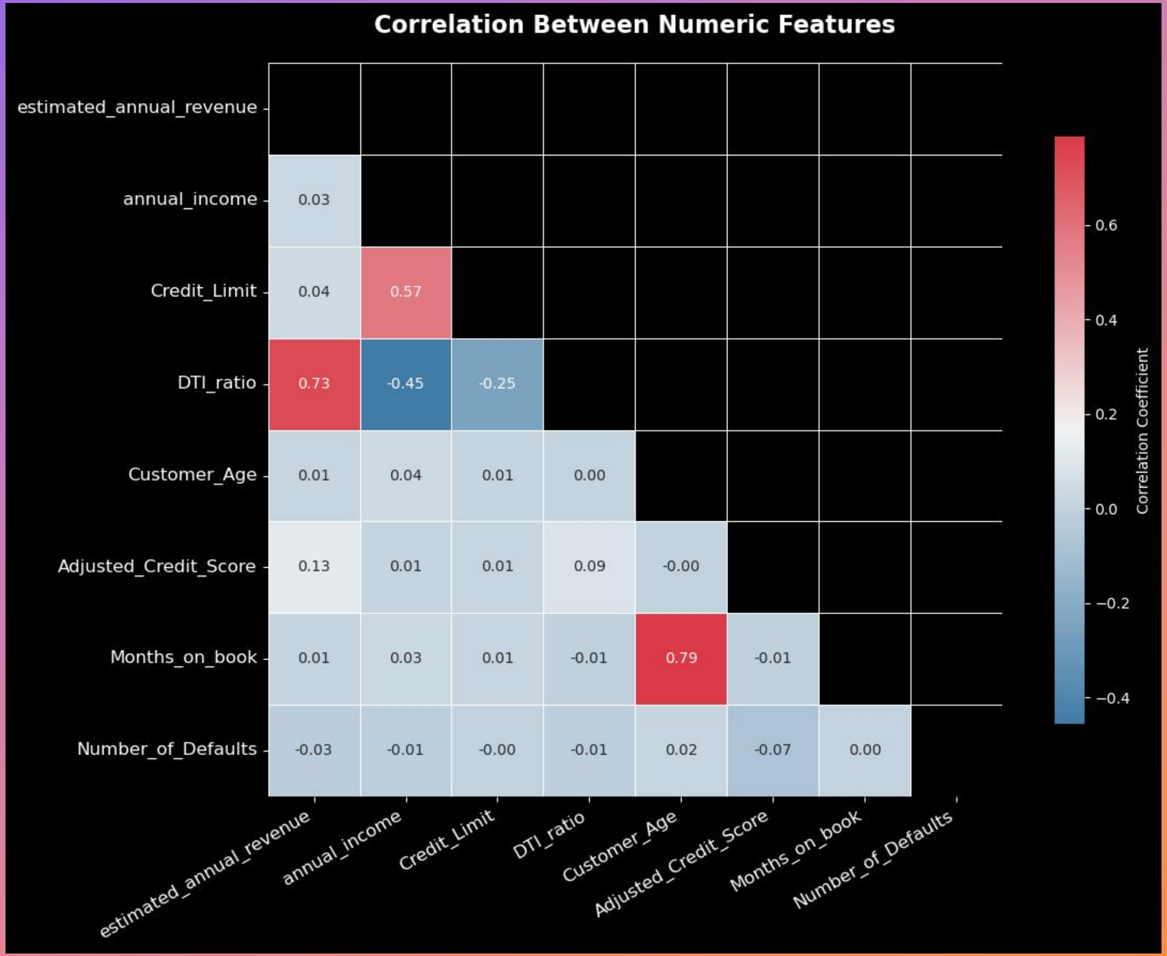




Executive Summary (2):

Machine Learning Modeling





THANK YOU ALL

[Tricia](#)

Turning data into meaningful stories with actionable insights

Data Analyst | Python • Power BI • SAS • SQL • Sentiment Analysis • NPS