DPDzero

DPDzero Data Analysis

Case Study and Recommendations for Loan Portfolio

Created By:
Anjali Roy

Introduction

About DPDzero

DPDzero is a Collections As a Service (CaaS) offering that enables lenders scale their collections with zero effort. The partners are financial institutions like Banks, FinTechs, NBFCs and MFIs.

Context

To study the given loan portfolio and perform data analysis, giving recommendations for the problem set.

Target

Analyze the given loan portfolio dataset and derive insightful information from raw data and generate relevant recommendations for spending on channels for loan repayment.

Understanding the problem set

Challenge 1

Calculate risk labels for all the borrowers

1. Unknown
2. Low
3. Medium
4. High

Challenge 2

Label all customers based on their tenure

1. Early
2. Mid
3. Late

Challenge 3

Segment borrowers into 3 cohorts based on the ticket size

1. Low
2. Medium
3. High

Challenge 4

Give channel spend recommendations

1. WhatsApp Bot
2. Voice Bot
3. Human/ Tele Calling

Solution

https://colab.research.google.com/drive/1FNSqaEjJDpj-pGtzhgkxoqURQHfiUWhy?usp=sharing

This analysis has been performed on Jupyter notebook, using pandas and matplotlib libraries of Python.

STEP 1

Preparing the dataset

1. Import the required libraries
2. Import the csv dataset
3. Check for the shape of data
4. Eliminate NA and duplicate values if any

```
In [1]: #Importing libraries
        import pandas as pd
        import matplotlib.pyplot as plt

        # Read the csv dataset
        df = pd.read_csv('./Data_Analyst_Assignment_Dataset.csv')

        print(df.head())
        df.shape

   Amount Pending      State  Tenure  Interest Rate       City Bounce String  \
0             963  Karnataka      11           7.69  Bangalore           SSS
1            1194  Karnataka      11           6.16  Bangalore           SSB
2            1807  Karnataka      14           4.24     Hassan           BBS
3            2451  Karnataka      10           4.70  Bangalore           SSS
4            2611  Karnataka      10           4.41     Mysore           SSB

   Disbursed Amount  Loan Number
0             10197       JZ6FS
1             12738       RDIOY
2             24640       WNW4L
3             23990       6LBJS
4             25590       ZFZUA

Out[1]: (24582, 8)
```

24582 rows of data and 8 columns

```
In [2]: #Clean the dataset: Checking for NA and duplicate values
        df = df.dropna()
        df = df.drop_duplicates()

In [3]: df.shape

Out[3]: (24582, 8)
```

STEP 2

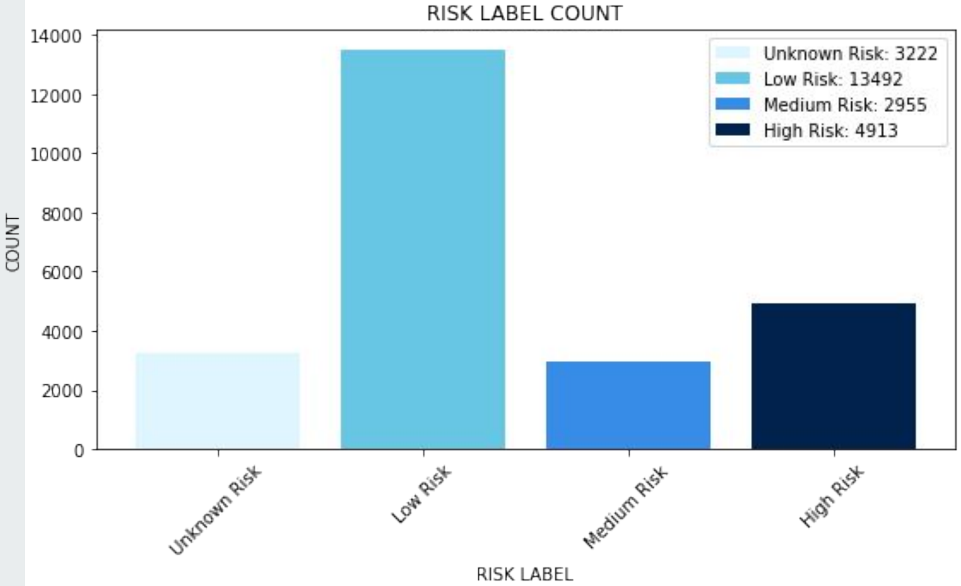
Processing the dataset

Challenge - 1: Calculate risk labels for all the borrowers.

Solution: Analyzing the bounce string for all the customers

1. **Unknown Risk** - FEMI
2. **Low Risk** - 0 Bounce in the last 6 months
3. **Medium Risk** - 1 Bounce in the last 6 months with no bounce in the last month.
   (Here, we are not including 0 bounce since that is already covered in low risk and we are calculating the string for 6 months only)
4. **High Risk** - Rest all conditions

```
In [6]: # Get the count for each type of risk
        risk_count = df['Risk Label'].value_counts()
        labels_ordered = ['Unknown Risk', 'Low Risk', 'Medium Risk', 'High Risk']
        risk_count = risk_count.reindex(labels_ordered)

        print(risk_count)

        Unknown Risk     3222
        Low Risk        13492
        Medium Risk      2955
        High Risk        4913
        Name: Risk Label, dtype: int64
```




```
In [4]: def calculate_risk_label(row):
            bounce_string = row['Bounce String']
            last_6_months = bounce_string[-6:]
            no_of_bounce = last_6_months.count('B') + last_6_months.count('L')

            if bounce_string == 'FEMI':
                return 'Unknown Risk'
            elif no_of_bounce == 0:
                return 'Low Risk'
            elif no_of_bounce == 1 and last_6_months[-1] not in ['B', 'L']:
                return 'Medium Risk'
            else:
                return 'High Risk'

        df['Risk Label'] = df.apply(calculate_risk_label, axis=1)
```

STEP 2

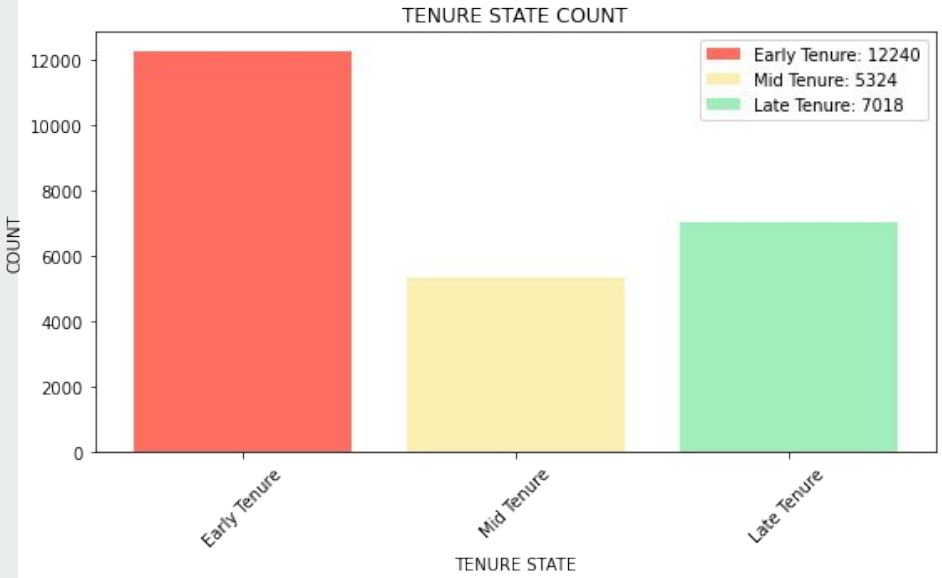
Processing the dataset

Challenge - 2: Label all customers based on their tenure.

Solution: Analyzing the bounce string and tenure period for all the customers

Length of string denotes the total number of payments made by the customer. So SSSB (acc. to the document) implies the customer has made 4 payments and has been in the book for 5 months (since the first month is FEMI). Therefore, the **total number of months a customer has been in the book = Length of string + 1**.

1. **Early Tenure** - Bounce String = FEMI **and** Length of Bounce String <= 2
2. **Late Tenure** - Length of (Tenure - (Bounce String+1)) <= 3
3. **Mid Tenure** - Every other customer




```
In [8]: def label_tenure(row):
            bounce_string = row['Bounce String']
            tenure = row['Tenure']

            # Adjusting bounce length to not count 'FEMI' and to be 0 if 'FEMI'
            if bounce_string == 'FEMI':
                bounce_length = 0
            else:
                bounce_length = len(bounce_string.replace('FEMI', ''))

            if bounce_string == 'FEMI' or bounce_length <= 2:
                return 'Early Tenure'
            elif tenure - (bounce_length + 1) <= 3:
                return 'Late Tenure'
            else:
                return 'Mid Tenure'

        df['Tenure Label'] = df.apply(label_tenure, axis=1)
```

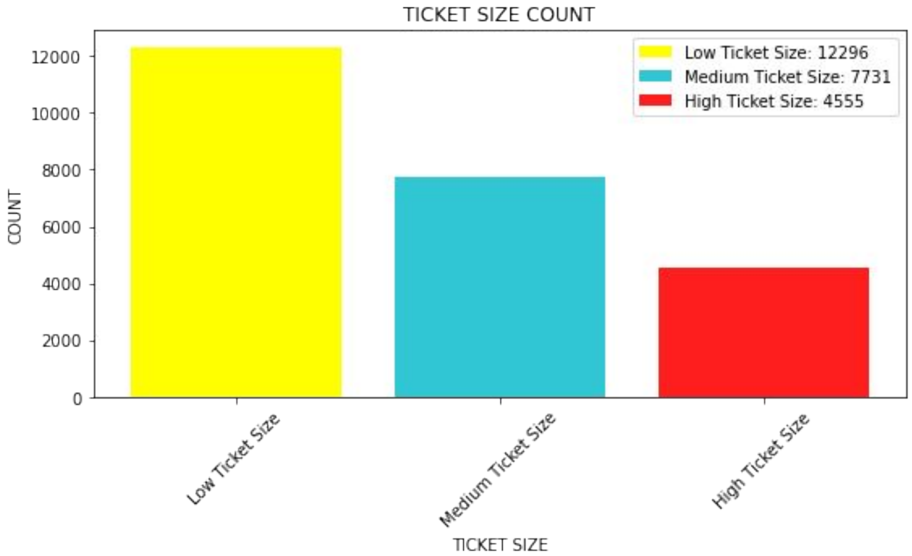
STEP 2

Processing the dataset

Challenge - 3: Segment borrowers based on ticket size.

Solution: Analyzing the amount pending for all the customers

We will first sort the amount pending for each customer in ascending order. This is required for categorization. Then we will find out the sum and divide it roughly in 3 equal parts. The first distribution will then be for low ticket size, second for medium and third will be for high ticket size. Then we will get the count from a counter and order the data frame based on the ticket sizes.




```
In [12]: df_sorted = df.sort_values(by='Amount Pending')
         print(df_sorted.head())
```

```
In [13]: total_amount_pending = df_sorted['Amount Pending'].sum()
distribute_low = total_amount_pending/3
distribute_medium = 2 * total_amount_pending/3

sum_current = 0
low_idx, medium_idx, high_idx = [], [], []

for idx, row in df_sorted.iterrows():
    sum_current += row['Amount Pending']
    if sum_current <= distribute_low:
        low_idx.append(idx)
    elif sum_current <= distribute_medium:
        medium_idx.append(idx)
    else:
        high_idx.append(idx)

df_sorted.loc[low_idx, 'Ticket Size'] = 'Low Ticket Size'
df_sorted.loc[medium_idx, 'Ticket Size'] = 'Medium Ticket Size'
df_sorted.loc[high_idx, 'Ticket Size'] = 'High Ticket Size'

sum_low = df_sorted[df_sorted['Ticket Size'] == 'Low Ticket Size']['Amount Pending'].sum()
sum_medium = df_sorted[df_sorted['Ticket Size'] == 'Medium Ticket Size']['Amount Pending'].sum()
sum_high = df_sorted[df_sorted['Ticket Size'] == 'High Ticket Size']['Amount Pending'].sum()
print(f"Sum_Low: {sum_low}\nSum_Medium: {sum_medium}\nSum_High: {sum_high}\n")

print(df_sorted.head())

Sum_Low: 14676828
Sum_Medium: 14676862
Sum_High: 14676917
```

STEP 2

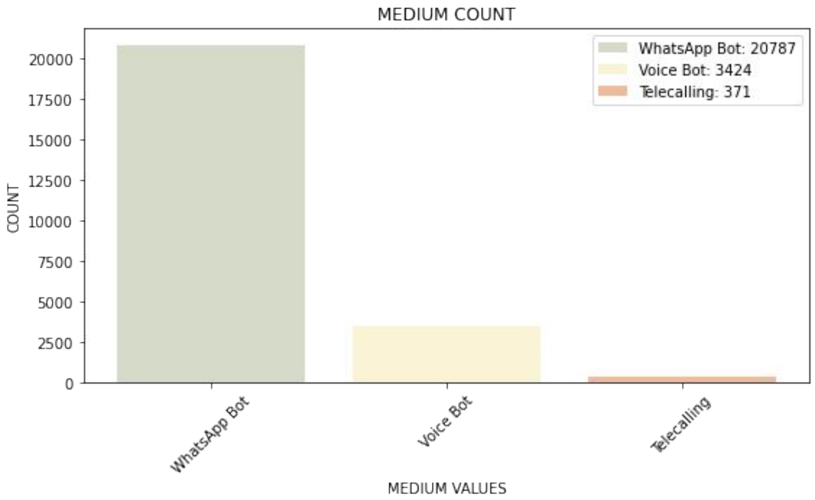
Processing the dataset

Challenge - 4: Segment borrowers based on ticket size.

Solution: Analyzing the previous solutions and recommending based on the given conditions

1. **WhatsApp Bot Medium** - Customers with FEMI or low risk and low ticket size
2. **Voice Bot Medium** - Customers in metro, low to medium risk and ticket size
3. **Telecalling** - Everyone else.

For Voice Bot Medium, I will first find out the states and their cities. From that, assumption is made that metro cities will be the capitals and major cities in the states apart from capitals.

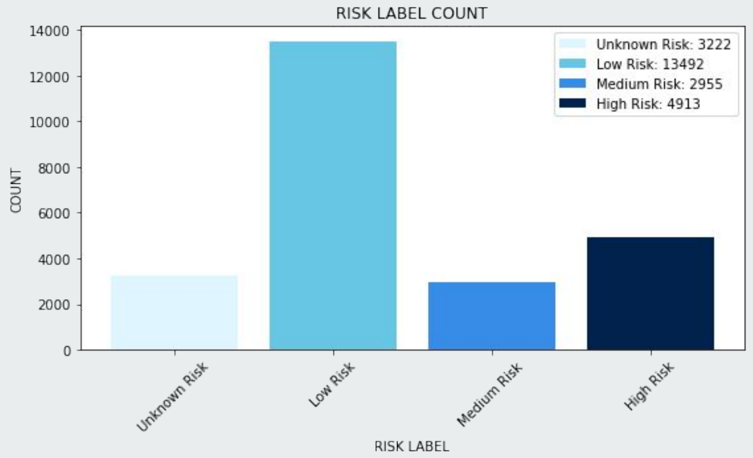


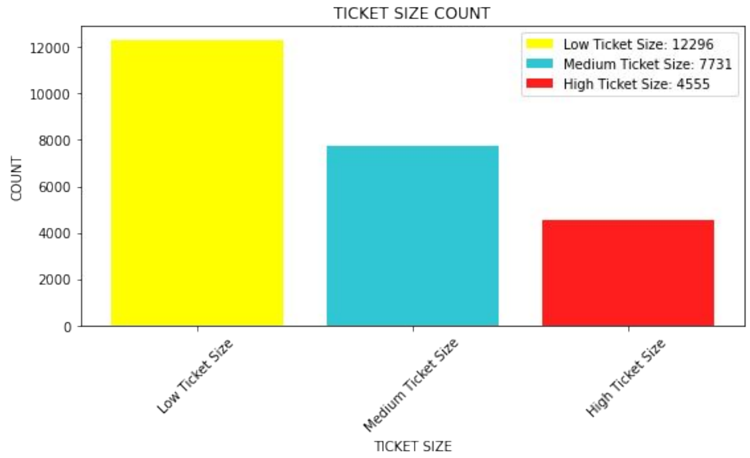

```
In [23]: # Get the total cost
costs = {'WhatsApp Bot': 5, 'Voice Bot': 10, 'Telecalling': 50}
adjusted_counts = medium * [costs[label] for label in labels_ordered]
cost_table = pd.DataFrame({'Count': medium, 'Adjusted Count': adjusted_counts})

total_count = cost_table['Count'].sum()
total_adjusted_count = cost_table['Adjusted Count'].sum()
cost_table.loc['Total'] = [total_count, total_adjusted_count]

print(cost_table)
```

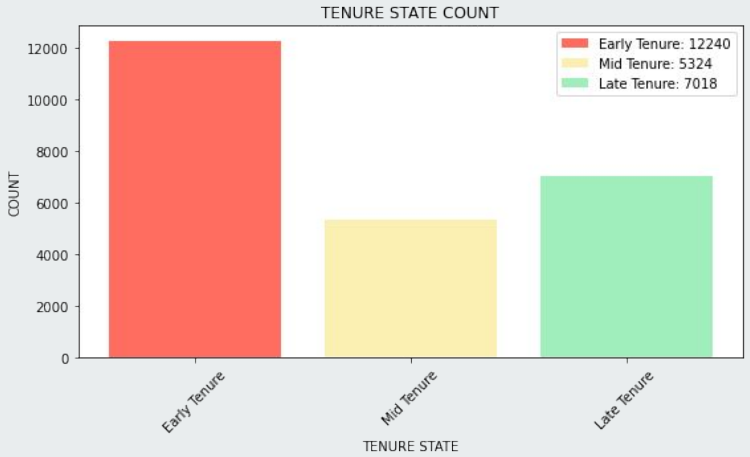
```
              Count  Adjusted Count
WhatsApp Bot  20787          103935
Voice Bot      3424           34240
Telecalling     371           18550
Total         24582          156725
```

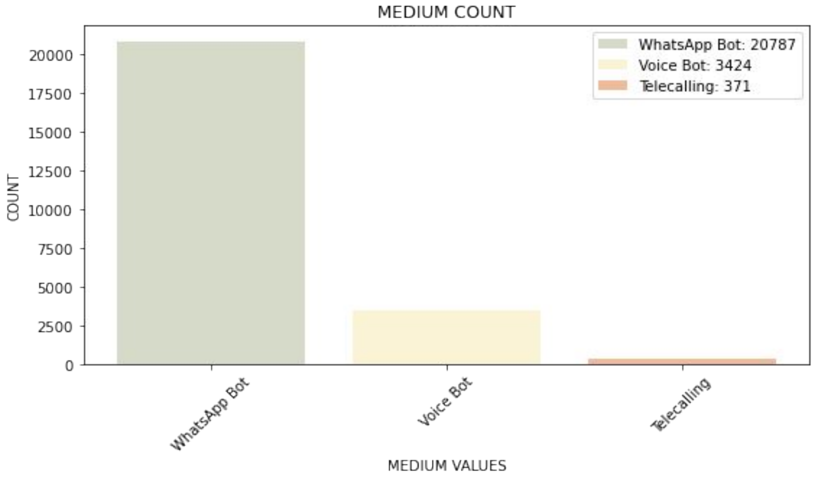
Therefore, total cost incurred is ₹ 1,56,725

Analyzing the results



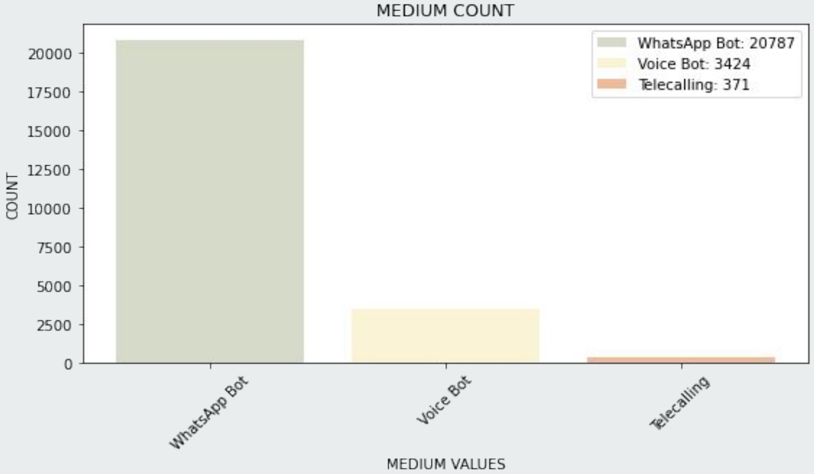






STEP 3

Analyzing the results



	Count	Adjusted Count
WhatsApp Bot	20787	103935
Voice Bot	3424	34240
Telecalling	371	18550
Total	24582	156725

Therefore, total cost incurred is ₹ 1,56,725

Target: To minimise total spending for loan recovery while maximizing time repayment.

Our target is to maximize WhatsApp medium and then Voice Bot so that Telecalling has the least number of borrowers. Then for Voice Bot, we need to find out maximum number of cities that speak English/Hindi in majority so that the results are further narrowed down.

Selecting cities for Voice Bot Medium

1. State Capitals: **Amravati, Bangalore, Mumbai, Chennai, Hyderabad** (Excluded Bhopal and Thiruvananthapuram)
2. All the cities of **Madhya Pradesh** since it is primarily a Hindi speaking state.
3. All the cities of **Kerala** since it is the most literate state, i.e., most of them can be safely assumed to speak English.
4. Big Cities: Cities which are culturally and economically important but are not the capitals of their respective states:

   a. Andhra Pradesh - Visakhapatnam, Pondicherry*
   b. Karnataka - Bangalore Rural, Mysore
   c. Maharashtra - Nagpur, Pune, Aurangabad, Thane, North Goa, South Goa
   d. Tamil Nadu - Madurai, Coimbatore, Vellore, Pondicherry*
   e. Telangana - Warangal

Total Cities in consideration = 83