1. Data Loading and Exploration

1.1 Importing Libraries and Load the Data

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
import gensim
from gensim import corpora
from gensim.models import LdaModel
import string
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score
from imblearn.over_sampling import SMOTE
from collections import Counter
!pip install pyLDAvis
import pyLDAvis.gensim
import pyLDAvis.gensim_models as gensimvis
import pyLDAvis

# Load the data
data_path = '/content/dataset.csv'
data = pd.read_csv(data_path)
print(f"Dataset shape: {data.shape}")
print(data.head())
```

```
Requirement already satisfied: numexpr in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.10.0)
Requirement already satisfied: funcy in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (2.0)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (1.2.2)
Requirement already satisfied: gensim in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (4.3.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from pyLDAvis) (67.7.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=2.0.0->pyLDAvis) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->pyLDAvi
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.10/dist-packages (from gensim->pyLDAvis) (7.0.4)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->pyLDAvis) (2.1.5)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=2.0.0->p
Requirement already satisfied: wrapt in /usr/local/lib/python3.10/dist-packages (from smart-open>=1.8.1->gensim->pyLDAvis) (1.14.
Dataset shape: (3290, 21)
                  rating                                       rating title  \
0  Rated 4 out of 5 stars                                     So far so good!
1  Rated 5 out of 5 stars  Perfect bank for freelancers, solopreneurs and...
2  Rated 5 out of 5 stars                        Great for my small business
3  Rated 5 out of 5 stars                            Great banking platform
4  Rated 5 out of 5 stars                           Very easy to deal with.

                                        Review text  Review date  \
0  So far so good! A solid and easy banking exper...  Dec 1, 2023
1  I never thought I could love a bank, yet here ...  Dec 1, 2023
2  I've banked with finor for a few years now and...  Dec 1, 2023
3  I love how simple, easy and helpful finor has ...  Dec 1, 2023
4  Very easy to deal with. Called the customer de...  Dec 1, 2023

  Date of Experience  rating_procesed  Year of review   Year of experience  \
0  November 30, 2023               4             2023                2023
1  November 30, 2023               5             2023                2023
2  November 30, 2023               5             2023                2023
3  November 30, 2023               5             2023                2023
4  November 30, 2023               5             2023                2023

   DIff in months   Unnamed: 9  ...  Unnamed: 11  Unnamed: 12  Unnamed: 13  \
```

```
2    NaN          NaN          NaN          NaN          NaN          NaN
3    NaN          NaN          NaN          NaN          NaN          NaN
4    NaN          NaN          NaN          NaN          NaN          NaN

   Unnamed: 20
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN

[5 rows x 21 columns]
```

2. Data Preprocessing

2.1 Handle Missing Values

```
# Handle missing values
data.dropna(subset=['rating', 'rating title', 'Review text', 'Review date', 'Date of Experience', 'rating_proces
print(f"Dataset shape after dropping null values: {data.shape}")
```

```
Dataset shape after dropping null values: (3290, 21)
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

2.2 Replace NaN Values in Text Columns

```
# Replace NaN values with empty strings in text columns to avoid issues during text processing
text_columns = ['rating title', 'Review text', 'Review date', 'Date of Experience']
data[text_columns] = data[text_columns].fillna('')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

2.3 Data Types and Basic Information

```
# Data types of the features
print(data.dtypes)
len(data)
```

```
rating                 object
rating title           object
Review text            object
Review date            object
Date of Experience     object
rating_procesed         int64
Year of review          int64
Year of experience      int64
DIff in months          int64
Unnamed: 9            float64
Unnamed: 10           float64
Unnamed: 11           float64
Unnamed: 12           float64
Unnamed: 13           float64
Issue                 float64
Unnamed: 15           float64
Unnamed: 16           float64
Unnamed: 17           float64
Unnamed: 18           float64
Unnamed: 19           float64
Unnamed: 20           float64
dtype: object
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
3290
```

3. Data Analysis and Visualization

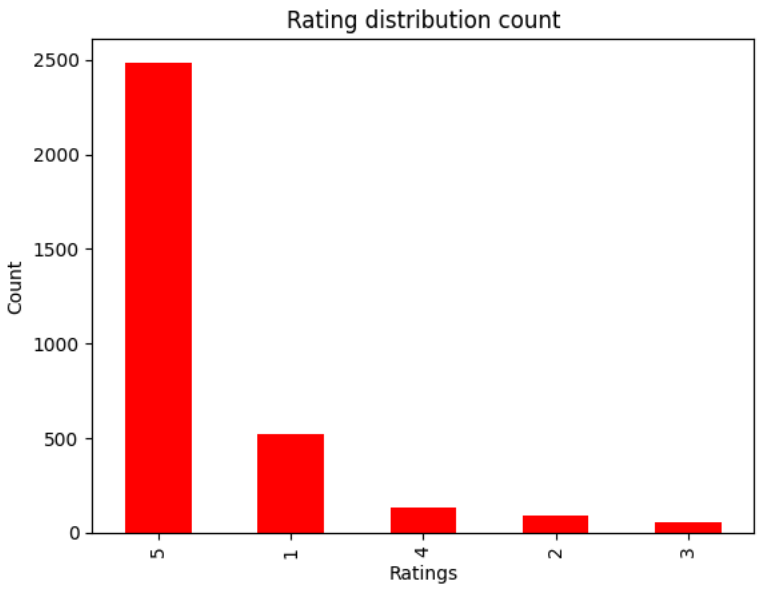
3.1 Analyze 'Rating' Column

```
#Distinct values of 'rating' and its count

print(f"Rating value count: \n{data['rating'].value_counts()}")


# Analyzing 'rating' column
data['rating_procesed'].value_counts().plot.bar(color='red')
plt.title('Rating distribution count')
plt.xlabel('Ratings')
plt.ylabel('Count')
plt.show()
```

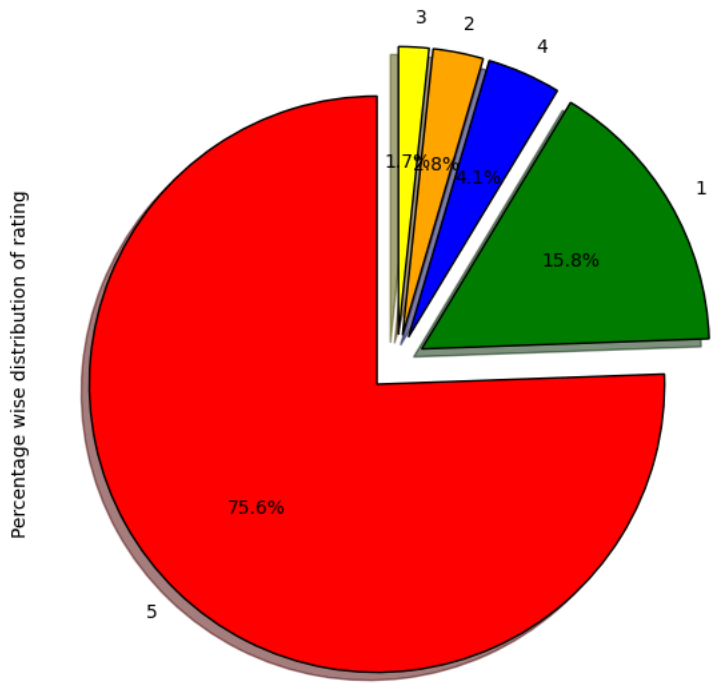
```
Rating value count:
rating
Rated 5 out of 5 stars    2486
Rated 1 out of 5 stars     519
Rated 4 out of 5 stars     136
Rated 2 out of 5 stars      93
Rated 3 out of 5 stars      56
Name: count, dtype: int64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```



3.2 Analyze 'Rating' Column in Pie Chart

```
print(f"Rating value count - percentage distribution: \n{round(data['rating_procesed'].value_counts()/data.shape
fig = plt.figure(figsize=(7,7))
colors = ('red', 'green', 'blue', 'orange', 'yellow')
wp = {'linewidth': 1, "edgecolor": 'black'}
tags = data['rating_procesed'].value_counts()/data.shape[0]
explode = (0.1, 0.1, 0.1, 0.1, 0.1)
tags.plot(kind='pie', autopct="%1.1f%%", shadow=True, colors=colors, startangle=90, wedgeprops=wp, explode=explo
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
Rating value count - percentage distribution:
rating_procesed
5    75.56
1    15.78
4     4.13
2     2.83
3     1.70
Name: count, dtype: float64
```



3.3 Analyze 'Review Text' Column

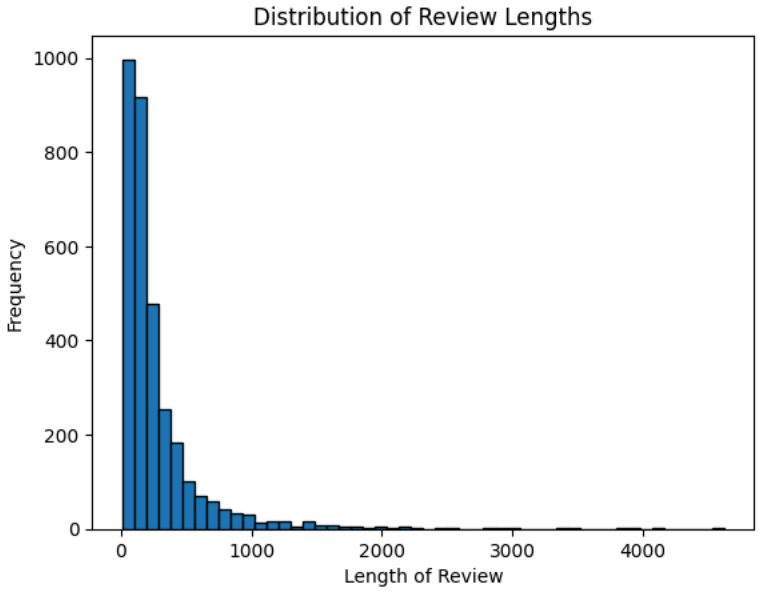
```
# Analyzing 'Review text' column
data['length'] = data['Review text'].apply(len)
# Descriptive statistics of the 'length' column
print(data['length'].describe())
```

```
count    3290.000000
mean      279.449240
std       372.609512
min        12.000000
25%        89.000000
50%       164.500000
75%       308.000000
max      4624.000000
Name: length, dtype: float64
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

3.4 Histogram of review lengths

```
# Histogram of review lengths
plt.hist(data['length'], bins=50, edgecolor='black')
plt.title('Distribution of Review Lengths')
plt.xlabel('Length of Review')
plt.ylabel('Frequency')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```



3.3 Wordcloud Visualization

3.3.1 Wordcloud for All Reviews

```
# Wordcloud for all reviews
cv = CountVectorizer(stop_words='english')
words = cv.fit_transform(data['Review text'])
reviews = " ".join([review for review in data['Review text']])
wc = WordCloud(background_color='white', max_words=50)
plt.figure(figsize=(10, 10))
plt.imshow(wc.generate(reviews))
plt.title('Wordcloud for all reviews', fontsize=10)
plt.axis('off')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

3.3.2 Wordcloud for High and Low Ratings

```
# Unique words in each rating category
low_ratings_reviews = " ".join([review for review in data[data['rating_procesed'] <= 3]['Review text']]).lower()
high_ratings_reviews = " ".join([review for review in data[data['rating_procesed'] > 3]['Review text']]).lower()
unique_low = " ".join([x for x in low_ratings_reviews if x not in high_ratings_reviews])
unique_high = " ".join([x for x in high_ratings_reviews if x not in low_ratings_reviews])
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

```
wc = WordCloud(background_color='white', max_words=50)
plt.figure(figsize=(10, 10))
plt.imshow(wc.generate(unique_low))
plt.title('Wordcloud for low ratings reviews', fontsize=10)
plt.axis('off')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

Wordcloud for low ratings reviews


```
wc = WordCloud(background_color='white', max_words=50)
plt.figure(figsize=(10, 10))
plt.imshow(wc.generate(unique_high))
plt.title('Wordcloud for high ratings reviews', fontsize=10)
plt.axis('off')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

Wordcloud for high ratings reviews

4. Text Preprocessing

4.1 Download and Load Stopwords

```
# Download stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

4.2 Preprocess Text Data

```
# Preprocess the text data
def preprocess(text):
    text = text.lower()  # Lowercase
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punctuation
    tokens = [word for word in text.split() if word not in stop_words]  # Tokenize and remove stopwords
    return tokens

data['tokens'] = data['Review text'].apply(preprocess)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

5. Topic Modeling Using LDA

5.1 Create Dictionary and Corpus

```
# Create a dictionary and corpus for LDA
dictionary = corpora.Dictionary(data['tokens'])
corpus = [dictionary.doc2bow(tokens) for tokens in data['tokens']]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

5.2 Build LDA Model

```
# Set parameters for LDA
num_topics = 10
passes = 15

# Build LDA model
lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=num_topics, passes=passes)

# Print topics with their probabilities
topics = lda_model.print_topics(num_words=200)
for topic in topics:
    print(topic)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
(0, '0.012*"finor" + 0.011*"account" + 0.008*"make" + 0.008*"even" + 0.007*"brick" + 0.007*"mortar" + 0.006*"money" + 0.005*"pretty
```

```
(1, '0.031*"account" + 0.026*"bank" + 0.013*"business" + 0.013*"card" + 0.013*"money" + 0.012*"customer" + 0.011*"service" + 0.011*
(2, '0.017*"overdraft" + 0.010*"wise" + 0.006*"superb" + 0.006*"extra" + 0.005*"functionalities" + 0.005*"concerns" + 0.005*"integra
(3, '0.056*"finor" + 0.019*"business" + 0.016*"bank" + 0.012*"account" + 0.010*"azlo" + 0.007*"use" + 0.007*"since" + 0.007*"card" +
(4, '0.061*"business" + 0.053*"finor" + 0.046*"easy" + 0.043*"great" + 0.035*"banking" + 0.031*"small" + 0.027*"use" + 0.023*"bank"
(5, '0.040*"finor" + 0.025*"business" + 0.018*"bank" + 0.016*"account" + 0.015*"easy" + 0.010*"experience" + 0.009*"im" + 0.009*"gre
(6, '0.024*"finor" + 0.018*"business" + 0.016*"account" + 0.016*"bank" + 0.014*"like" + 0.012*"money" + 0.011*"love" + 0.010*"able"
(7, '0.027*"finor" + 0.018*"business" + 0.013*"recommend" + 0.012*"money" + 0.011*"reserves" + 0.011*"bank" + 0.011*"set" + 0.011*"s
(8, '0.016*"fee" + 0.008*"thanks" + 0.007*"mobile" + 0.007*"approval" + 0.006*"accounting" + 0.006*"payment" + 0.005*"financial" + 0
(9, '0.024*"finor" + 0.015*"bank" + 0.014*"account" + 0.011*"support" + 0.008*"business" + 0.008*"fraud" + 0.007*"customer" + 0.007*
```

5.3 Map Topics to Intents

```
# Assuming data, lda_model, and corpus are already defined and processed correctly

# Function to map topic ID to intent
def map_topic_to_intent(topic):
    intent_mapping = {
        0: "General Setup and Recommendations",
        1: "Technical Issues - Bugs and Errors",
        2: "Customer Support Experience",
        3: "Account Management and Digital Banking",
        4: "Transaction Issues",
        5: "Access and Availability Issues",
        6: "Feature Requests and Improvements",
        7: "Positive Recommendations and Reviews",
        8: "Invoice Management",
        9: "Fraud and Security Concerns"
    }
    return intent_mapping.get(topic, "Unknown")  # Default to "Unknown" if topic ID not found

# Function to get topic distribution for corpus
def get_topic_distribution(model, corpus):
    topics = []
    for doc in corpus:
        topic_distribution = model.get_document_topics(doc)
        topics.append(sorted(topic_distribution, key=lambda x: x[1], reverse=True)[0][0])
    return topics

# Assign intents to reviews based on the most likely topic
data['topic'] = get_topic_distribution(lda_model, corpus)
data['intent'] = data['topic'].apply(map_topic_to_intent)

# Display the first few rows with intents
print(data[['Review text', 'tokens', 'topic', 'intent']].head())
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
                                        Review text  \
0  So far so good! A solid and easy banking exper...
1  I never thought I could love a bank, yet here ...
2  I've banked with finor for a few years now and...
3  I love how simple, easy and helpful finor has ...
4  Very easy to deal with. Called the customer de...

                                              tokens  topic  \
0  [far, good, solid, easy, banking, experience, ...      6
1  [never, thought, could, love, bank, yet, lovin...      6
2  [ive, banked, finor, years, perfect, small, bu...      4
3  [love, simple, easy, helpful, finor, business,...      3
4  [easy, deal, called, customer, department, occ...      4

                                 intent
0      Feature Requests and Improvements
1      Feature Requests and Improvements
2                     Transaction Issues
3  Account Management and Digital Banking
4                     Transaction Issues
```

5.4 Review Tokens

```
# Print all tokens for the first few reviews
print("First few reviews and their tokens:")
for index, row in data.head().iterrows():
    print(f"Review {index + 1}: {row['Review text']}")
    print(f"Tokens: {row['tokens']}\n")

# Print tokens for a specific review
specific_index = 0  # Change this to the index of the review you want to inspect
print(f"Specific Review: {data.loc[specific_index, 'Review text']}")
print(f"Tokens: {data.loc[specific_index, 'tokens']}")
```

```
First few reviews and their tokens:
Review 1: So far so good! A solid and easy banking experience. One thing I wish I could have is a separate savings account but they
Tokens: ['far', 'good', 'solid', 'easy', 'banking', 'experience', 'one', 'thing', 'wish', 'could', 'separate', 'savings', 'account',

Review 2: I never thought I could love a bank, yet here I am, loving you, finor. My single member LLC has been a customer for 2 year
Tokens: ['never', 'thought', 'could', 'love', 'bank', 'yet', 'loving', 'finor', 'single', 'member', 'llc', 'customer', '2', 'years',

Review 3: I've banked with finor for a few years now and it's perfect for my small business. I can deposit checks on the go, and int
Tokens: ['ive', 'banked', 'finor', 'years', 'perfect', 'small', 'business', 'deposit', 'checks', 'go', 'integrate', 'payment', 'proc

Review 4: I love how simple, easy and helpful finor has been for my business banking. Recently they implemented a feature which allo
Tokens: ['love', 'simple', 'easy', 'helpful', 'finor', 'business', 'banking', 'recently', 'implemented', 'feature', 'allows', 'autom

Review 5: Very easy to deal with. Called the customer department on few occasions and the issues were resolved quickly. Definitely r
Tokens: ['easy', 'deal', 'called', 'customer', 'department', 'occasions', 'issues', 'resolved', 'quickly', 'definitely', 'recommend

Specific Review: So far so good! A solid and easy banking experience. One thing I wish I could have is a separate savings account bu
Tokens: ['far', 'good', 'solid', 'easy', 'banking', 'experience', 'one', 'thing', 'wish', 'could', 'separate', 'savings', 'account',
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

6. Feature Extraction and Model Training

6.1 TF-IDF Vectorization

```
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from collections import Counter
import pandas as pd

# Create a TF-IDF Vectorizer
tfidf = TfidfVectorizer(max_features=5000, stop_words='english')

# Transform the review text into TF-IDF features
X = tfidf.fit_transform(data['Review text'])

# Target variable
y = data['intent']

# Check the class distribution before applying SMOTE
class_counts = Counter(y)
print(f"Class distribution before SMOTE: {class_counts}")

# Apply SMOTE to balance the dataset with k_neighbors set to 1
smote = SMOTE(random_state=42, k_neighbors=1)
X_resampled, y_resampled = smote.fit_resample(X, y)

# Check the new class distribution
resampled_class_counts = Counter(y_resampled)
print(f"Resampled dataset shape: {resampled_class_counts}")
```

```
Class distribution before SMOTE: Counter({'Transaction Issues': 1494, 'Technical Issues - Bugs and Errors': 614, 'Account Management
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
Resampled dataset shape: Counter({'Feature Requests and Improvements': 1494, 'Transaction Issues': 1494, 'Account Management and Dig
```

6.2 Train-Test Split Using Random Forest Model

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from collections import Counter
```

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)

# Train the Random Forest model
model = RandomForestClassifier(
    n_estimators=100,         # Increased number of trees for better performance
    random_state=2,          # For reproducibility
    max_depth=10,           # Allowing trees to grow fully
    min_samples_split=12,      # Minimum samples to split
    min_samples_leaf=8,       # Minimum samples at leaf
    max_features='log2',      # Number of features to consider when looking for the best split
    bootstrap=True,           # Use bootstrap samples
    oob_score=False,          # Do not use out-of-bag samples for validation
    criterion='gini',         # Use Gini impurity for split quality
    class_weight='balanced'         # All classes have the same weight
)
model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', max_depth=10,
                       max_features='log2', min_samples_leaf=8,
                       min_samples_split=12, random_state=2)
```

6.3 Model Evaluation

```
# Predict and Evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
```

```
                                   precision    recall  f1-score   support

         Access and Availability Issues       0.78      0.63      0.69       314
Account Management and Digital Banking       0.86      0.45      0.59       289
           Customer Support Experience       0.84      1.00      0.91       271
     Feature Requests and Improvements       0.80      0.77      0.78       307
           Fraud and Security Concerns       0.81      0.86      0.83       300
     General Setup and Recommendations       0.77      0.88      0.82       330
                    Invoice Management       0.97      1.00      0.99       299
  Positive Recommendations and Reviews       0.81      0.95      0.87       280
    Technical Issues - Bugs and Errors       0.81      0.60      0.69       306
                    Transaction Issues       0.55      0.77      0.64       292

                              accuracy                           0.79      2988
                             macro avg       0.80      0.79      0.78      2988
                          weighted avg       0.80      0.79      0.78      2988

Accuracy: 0.7878179384203481
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

SOME MORE ADDITIONAL CHANGES HERE TO EVALUATE MORE EASILY OUR RESULTS

7. Add Intent and Topic Columns to the Dataset

7.1 Assign Intents and Topics

```
# Assign intents to reviews based on the most likely topic
def get_topic_distribution(model, corpus):
    topics = []
    for doc in corpus:
        topic_distribution = model.get_document_topics(doc)
        topics.append(sorted(topic_distribution, key=lambda x: x[1], reverse=True)[0][0])
    return topics

data['topic'] = get_topic_distribution(lda_model, corpus)
data['intent'] = data['topic'].apply(map_topic_to_intent)

# Add intent number
data['intent_number'] = data['topic']
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

7.2 Display the Modified DataFrame

```
# Display the first few rows with intents
print(data[['Review text', 'tokens', 'topic', 'intent', 'intent_number']].head())
```

```
                                        Review text  \
0  So far so good! A solid and easy banking exper...
1  I never thought I could love a bank, yet here ...
2  I've banked with finor for a few years now and...
3  I love how simple, easy and helpful finor has ...
4  Very easy to deal with. Called the customer de...

                                              tokens  topic  \
0  [far, good, solid, easy, banking, experience, ...      6
1  [never, thought, could, love, bank, yet, lovin...      6
2  [ive, banked, finor, years, perfect, small, bu...      4
3  [love, simple, easy, helpful, finor, business,...      3
4  [easy, deal, called, customer, department, occ...      4

                                  intent  intent_number
0       Feature Requests and Improvements              6
1       Feature Requests and Improvements              6
2                     Transaction Issues              4
3  Account Management and Digital Banking              3
4                     Transaction Issues              4
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

7.3 Create a New Dataset with Additional Columns

```
# Create a new DataFrame with the additional columns
new_data = data.copy()

# Save the new dataset to an Excel file
new_data.to_excel('/content/new_dataset.xlsx', index=False)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

7.4 Save and Download the New Dataset to a EXCEL File

```
# Provide a button to download the new dataset
from google.colab import files
files.download('/content/new_dataset.xlsx')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

7.5 Save and Download the New Dataset to a CSV File

```
# Save the new dataset to a CSV file
new_data.to_csv('/content/new_dataset.csv', index=False)

### 6.2 Provide a Download Button in Colab
from google.colab import files

# Provide a button to download the new dataset
files.download('/content/new_dataset.csv')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```

7.6 Display the First and Last 5 Rows of the New DataFrame

```
# Display the first 5 rows of the new dataset
print("First 5 rows of the new dataset:")
print(new_data.head())

# Display the last 5 rows of the new dataset
print("Last 5 rows of the new dataset:")
print(new_data.tail())
```

```
3  Account Management and Digital Banking             3
4                     Transaction Issues             4

[5 rows x 26 columns]
```

```
3288  General Setup and Recommendations                0
3289  Technical Issues - Bugs and Errors               1

[5 rows x 26 columns]
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transfor
  and should_run_async(code)
```

Load the New Dataset

```
import pandas as pd

# Load the new dataset
new_data_path = '/content/new_dataset.csv'
new_data = pd.read_csv(new_data_path)
print(f"New dataset shape: {new_data.shape}")
```

```
New dataset shape: (3290, 26)
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_c
  and should_run_async(code)
```

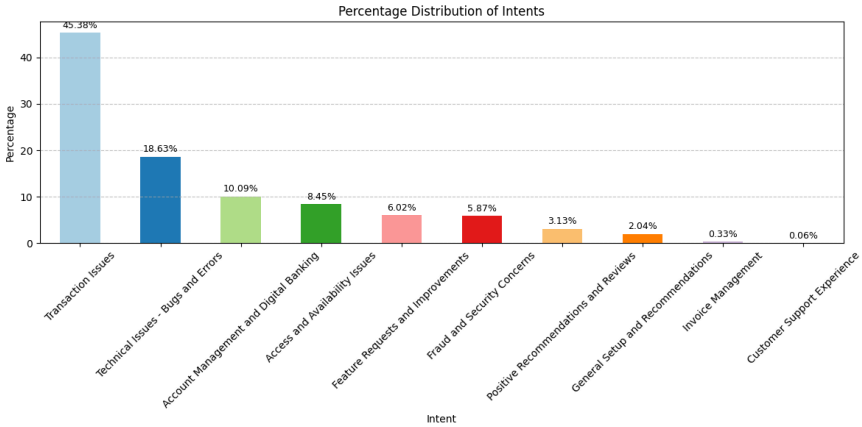
7.2 Visualize the Percentage of Each Intent

```
import matplotlib.pyplot as plt

# Calculate the percentage of each intent
intent_counts = new_data['intent'].value_counts()
intent_percentages = intent_counts / len(new_data) * 100

# Plot the bar graph
plt.figure(figsize=(12, 6))
intent_percentages.plot(kind='bar', color=plt.cm.Paired.colors)
plt.title('Percentage Distribution of Intents')
plt.xlabel('Intent')
plt.ylabel('Percentage')
plt.xticks(rotation=45)  # Rotate x-labels for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add horizontal grid lines
for i, v in enumerate(intent_percentages):
    plt.text(i, v + 0.5, f"{v:.2f}%", ha='center', va='bottom', fontsize=9)  # Display percentage on top of each
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
```



7.3 Visualize the Count of Each Intent

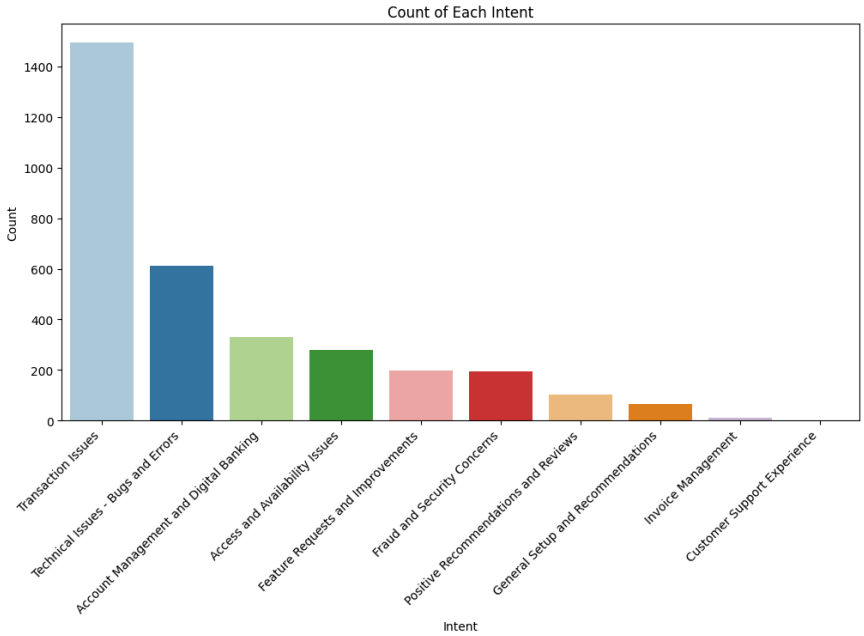
```
import seaborn as sns

# Plot the bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x=intent_counts.index, y=intent_counts.values, palette='Paired')
plt.title('Count of Each Intent')
plt.xlabel('Intent')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
<ipython-input-167-866247723d11>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.barplot(x=intent_counts.index, y=intent_counts.values, palette='Paired')
```



Analyze Review Length by Intent

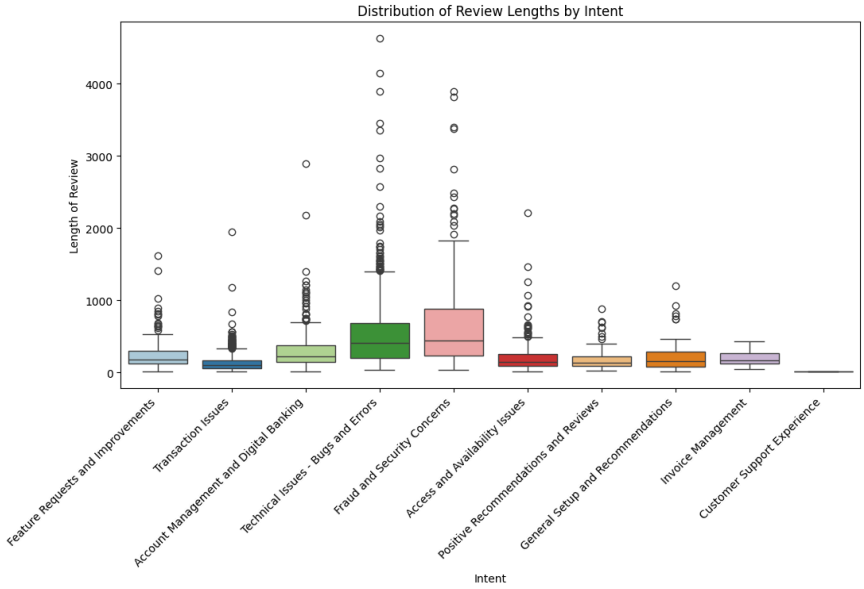
```
# Calculate the length of each review
new_data['length'] = new_data['Review text'].apply(len)

# Plot the distribution of review lengths by intent
plt.figure(figsize=(12, 6))
sns.boxplot(x='intent', y='length', data=new_data, palette='Paired')
plt.title('Distribution of Review Lengths by Intent')
plt.xlabel('Intent')
plt.ylabel('Length of Review')
plt.xticks(rotation=45, ha='right')
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning
  and should_run_async(code)
<ipython-input-168-13bccc58bf67>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.

  sns.boxplot(x='intent', y='length', data=new_data, palette='Paired')
```



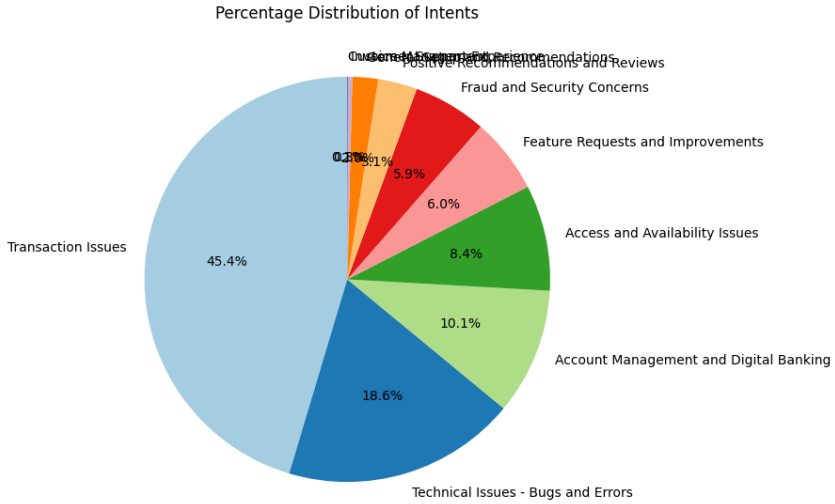
Save and Download Visualizations

```
# Save the pie chart
pie_chart_path = '/content/intent_pie_chart.png'
plt.figure(figsize=(10, 7))
intent_percentages.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=plt.cm.Paired.colors)
plt.title('Percentage Distribution of Intents')
plt.ylabel('')
plt.savefig(pie_chart_path)
plt.show()

# Save the bar chart
bar_chart_path = '/content/intent_bar_chart.png'
plt.figure(figsize=(12, 6))
sns.barplot(x=intent_counts.index, y=intent_counts.values, palette='Paired')
plt.title('Count of Each Intent')
plt.xlabel('Intent')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.savefig(bar_chart_path)
plt.show()

# Provide download buttons for the visualizations
files.download(pie_chart_path)
files.download(bar_chart_path)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarn
  and should_run_async(code)
```




```
<ipython-input-169-107d9d038e19>:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.

  sns.barplot(x=intent_counts.index, y=intent_counts.values, palette='Paired')
```