B5W2: Customer Experience Analytics for Fintech Apps

A Real-World Data Engineering Challenge: Scraping, Analyzing, and Visualizing Google Play Store Reviews

CHALLENGE OVERVIEW

This week's challenge centers on analyzing customer satisfaction with mobile banking apps by collecting and processing user reviews from the Google Play Store for three Ethiopian banks:

- Commercial Bank of Ethiopia (CBE)
- Bank of Abyssinia (BOA)
- Dashen Bank

You'll scrape app reviews, analyze sentiments and themes, and visualize insights to simulate the role of a Data Analyst at Omega Consultancy, a firm advising banks.
Building on Week 1's foundational skills, this week introduces web scraping, thematic NLP analysis, and basic database engineering.

BUSINESS OBJECTIVE

Omega Consultancy is supporting banks to improve their mobile apps to enhance customer retention and satisfaction. Your role as a Data Analyst is to:

- Scrape user reviews from the Google Play Store.
- Analyze sentiment (positive/negative/neutral) and extract themes (e.g., "bugs", "UI").
- Identify satisfaction drivers (e.g., speed) and pain points (e.g., crashes).
- Store cleaned review data in a Oracle database.
- Deliver a report with visualizations and actionable recommendations.

Scenarios

These scenarios simulate real consulting tasks faced by product, marketing, and engineering teams in the banking.

1. Scenario 1: Retaining Users

CBE has a 4.4-star rating, BOA 2.8, and Dashen 4.0. Users complain about slow loading during transfers. Analyze if this is a broader issue and suggest areas for app investigation.

1. Scenario 2: Enhancing Features

Extract desired features (e.g., transfer, fingerprint login, faster loading times) through keyword and theme extraction. Recommend how each bank can stay competitive.

1. Scenario 3: Managing Complaints

Cluster and track complaints (e.g., "login error") to guide AI chatbot integration and faster support resolution strategies.

These scenarios highlight user retention, feature innovation, and support efficiency, aligning with fintech priorities.

DATASET OVERVIEW

You will scrape reviews from the Google Play Store for three banks' apps, collecting:

- Review Text: User feedback (e.g., "Love the UI, but it crashes often").
- Rating: 1–5 stars.
- Date: Posting date (time optional).
- Bank/App Name: E.g., "Commercial bank of Ethiopia Mobile".
- Source: Google Play.

Minimum: 400 reviews per bank (1200 total). If scraping is limited,

TEAM

Facilitators:

- Mahlet
- Kerod
- Rediet
- Rehmet

KEY DATES

- **Introduction**: 8:00 AM UTC, Wednesday, 04 June 2024
- **Interim Submission**: 8:00 PM UTC, Sunday, 08 June 2024
- **Final Submission**: 8:00 PM UTC, Tuesday, 10 June 2024

COMMUNICATION & SUPPORT

- **Slack channel:** *#all-week-2*
- **Office hours:** Mon–Fri, 08:00–15:00 UTC

LEARNING OBJECTIVES

By the end of this week, you will be able to:

- Scrape and preprocess user reviews from the Google Play Store to prepare text data for analysis.
- Apply NLP techniques to analyze review sentiment and identify key customer feedback themes.
- Design and implement a relational database schema in Oracle to store and manage review data.
- Derive actionable insights from analyzed data and create compelling visualizations for business stakeholders.
- Develop and present a data-driven report with recommendations for app improvement in a fintech context.
- Employ Git for version control and write unit tests to ensure the reliability of data processing scripts.

PROJECT PLANNING - DATA COLLECTION & ANALYSIS

Tasks:

- Scrape and preprocess reviews.
- Analyze sentiment and themes.
- Synthesize insights and visualize results.

KPIs:

- **Proactivity**: Sharing scraping/NLP references.
- **Data Quality**: 1,200+ clean reviews with <5% errors.
- **Insights**: 3+ drivers/pain points per bank.
- **Clarity**: Stakeholder-friendly visualizations.

DELIVERABLES AND TASKS

Task 1: Data Collection and Preprocessing

Scrape reviews from the Google Play Store, preprocess them for analysis, and manage code via GitHub.

Tasks:

- Git Setup:
  - Create a GitHub repository.
  - Include .gitignore, requirements.txt.
  - Use “task-1” branch,Commit frequently with meaningful messages, ideally after completing logical chunks of work or at the end of each work session.
- Web Scraping:
  - Use google-play-scraper to collect reviews, ratings, dates, and app names for three banks.
  - Target 400+ reviews per bank (1,200 total).
- Preprocessing:
  - Remove duplicates, handle missing data.
  - Normalize dates (e.g., to YYYY-MM-DD).
  - Save as CSV with columns: review, rating, date, bank, source.

KPIs:

- 1,200+ reviews collected with <5% missing data.
- Clean CSV dataset.
- Organized Git repo with clear commits.

Minimum Essential:

- Scrape 400 reviews per bank (1200 total).
- Commit a preprocessing script.
- Update README.md with methodology.

Task 2: Sentiment and Thematic Analysis

Description: Quantify review sentiment and identify themes to uncover satisfaction drivers and pain points.

Tasks:

- Sentiment Analysis:
  - Use distilbert-base-uncased-finetuned-sst-2-english to compute sentiment scores (positive, negative, neutral). Alternatively, you can start with simpler libraries like VADER or TextBlob and compare results if time permits.
  - Aggregate by bank and rating (e.g., mean sentiment for 1-star reviews).
- Thematic Analysis:
  - A theme refers to a recurring concept or topic within user reviews. For this challenge, themes will help summarize user feedback into actionable categories for the banks.
  - **Keyword Extraction & Manual/Rule-Based Clustering**:
    - Extract significant keywords and n-grams using TF-IDF or spaCy (e.g., “login error”, “slow transfer”, “good UI”).
    - To aid in grouping these keywords and understanding broader review topics, you can optionally employ topic modeling techniques
    - Group related keywords and phrases into 3-5 overarching themes per bank (e.g., 'Account Access Issues', 'Transaction Performance', 'User Interface & Experience', 'Customer Support', 'Feature Requests'). Document your grouping logic.
  - **Pipeline:**
    - Script preprocessing (tokenization, stop-word removal, lemmatization if useful) with Pandas and NLP libraries.
    - Save results as CSV (e.g., review_id, review_text, sentiment_label, sentiment_score, identified_theme(s)).

        - Extract keywords with spaCy or TF-IDF (e.g., “crash”, “support”).
        - Cluster into 3–5 themes per bank (e.g., UI, reliability).
- Git:
    - Use “task-2” branch, commit scripts, merge via pull request.

KPIs:

- Sentiment scores for 90%+ reviews.
- 3+ themes per bank with examples.
- Modular pipeline code.

Minimum Essential:

- Sentiment scores for 400 reviews.
- 2 themes per bank via keywords.
- Commit analysis script.

Task 3: Store Cleaned Data in Oracle

Design and implement a relational database in Oracle to persistently store the cleaned and processed review data. This step simulates real-world enterprise data engineering workflows, especially within banks where Oracle is commonly used.

Tasks:

- Oracle Database Setup (Oracle XE - Express Edition)
    - Oracle XE setup can sometimes be complex depending on your system configuration. If you encounter persistent issues, inform facilitators and you may use PostgreSQL as a fallback. However, attempting Oracle XE is highly encouraged.
- Create a database named bank_reviews
- Define schema:
    - Banks Table: Stores information about the banks.
    - Reviews Table: Stores the scraped and processed review data.
- Insert cleaned review data using Python

KPIs:

- Working connection + insert script
- Table populated with >1,000 entries
- SQL dump committed to GitHub

Task 4: Insights and Recommendations

Description: Derive insights from sentiment and themes, visualize results, and recommend app improvements.

Tasks:

- Insights:
    - Identify 2+ drivers (e.g., fast navigation) and pain points (e.g., crashes).
    - Compare banks (e.g., CBE vs. BOA).
    - Suggest 2+ improvements (e.g., add budgeting tool).
- Visualization:

    - Create 3–5 plots (Matplotlib, Seaborn): sentiment trends, rating distributions, keyword clouds.
- Ethics:
    - Note potential review biases (e.g., negative skew).
- Git:
    - Use “task-4” branch, commit visuals/reports, merge via pull request.

KPIs:

- 2+ drivers/pain points with evidence.
- Clear, labeled visualizations.
- Practical recommendations.

Minimum Essential:

- 1 driver, 1 pain point per bank.
- 2 plots (e.g., sentiment bar, keyword chart).
- 4-page final report

DUE DATES

- **Sunday, 08 June 2024, 8:00 PM UTC**:
    - **GitHub Link**: Main branch with “task-1” merged, partial “task-2”.
    - **Interim Report**: 2 pages, covering scraping and early analysis.
- **Tuesday, 10 June 2024, 8:00 PM UTC**:
    - **GitHub Link**: Main branch with all tasks.
    - **Final Report**: 7 pages, 7 plots, Medium style.

OTHER CONSIDERATIONS

- **Documentation**: Comment code, update README.md.
- **Collaboration**: Use GitHub Issues for coordination.
- **Professionalism**: Meet deadlines, learn proactively.
- **Flexibility**: Pivot to datasets if scraping fails.

TUTORIALS SCHEDULE

In the following, the **Bold** indicates morning sessions, and *Italic* indicates afternoon sessions.

- **Day 1**:
    - *Challenge Intro & UX in Fintech (Mahlet,)*
    - *Scraping Basics and Data Cleaning and Preprocessing Strategy (Rediet)*
- **Day 2**:
    - **Sentiment Analysis and Keyword Extraction (Kerod)**
    - Database Fundamentals & Oracle XE Setup (Rehmet)
- **Day 3**:
    - **Insight communication and Plotting for Stakeholders (Rehmet)**
    - Introduction to Unit Testing(Kerod)
- **Day 4**:
    - Q&A (Mahlet & Rediet)

REFERENCES

- **Web Scraping (Google Play Store):**
    - google-play-scraper library:

  - PyPI Page: https://pypi.org/project/google-play-scraper/
  - GitHub Repository: https://github.com/JoMingyu/google-play-scraper
- **Data Handling & Preprocessing:**
  - Pandas Documentation:
  - 10 Minutes to pandas: https://pandas.pydata.org/pandas-docs/stable/user_guide/10min.html
  - Working with Text Data: https://pandas.pydata.org/pandas-docs/stable/user_guide/text.html
  - Handling Missing Data: https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html
- **Sentiment Analysis & NLP:**
  - Hugging Face transformers (for distilbert-base-uncased-finetuned-sst-2-english):
  - Documentation - Pipelines (Sentiment Analysis): https://huggingface.co/docs/transformers/main_classes/pipelines#transformers.TextClassificationPipeline
  - Model Card (distilbert-base-uncased-finetuned-sst-2-english): https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english
  - VADER Sentiment :
    - GitHub Repository: https://github.com/cjhutto/vaderSentiment
  - TextBlob:
    - https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis
- **Thematic Analysis (Keyword Extraction & Topic Modeling):**
  - **spaCy (for tokenization, lemmatization, POS tagging):**
  - https://spacy.io/usage/spacy-101
  - **Scikit-learn (for TF-IDF, LDA, NMF):**
  - **TF-IDF (TfidfVectorizer):** https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
  - **LDA (LatentDirichletAllocation):** https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.LatentDirichletAllocation.html
  - **NMF (NMF):** https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html
  - **Hugging Face transformers (for Zero-Shot Classification - e.g., facebook/bart-large-mnli):**
  - ZeroShotClassificationPipeline: https://huggingface.co/docs/transformers/main_classes/pipelines#transformers.ZeroSh
  - **Sentence Transformers (for embeddings if doing embedding-based clustering):**
  - https://www.sbert.net/docs/quickstart.html
- **Database (Oracle XE):**
  - Oracle Database Express Edition (XE) Downloads:
  - https://www.oracle.com/database/technologies/xe-downloads.html (Select OS appropriate version)
  - oracledb (Python driver for Oracle Database):
  - https://python-oracledb.readthedocs.io/en/latest/
  - Oracle SQL Developer (GUI Tool):
  - https://www.oracle.com/database/sqldeveloper/technologies/download/
  - Basic SQL Tutorials (general, adapt syntax for Oracle):
  - W3Schools SQL Tutorial: https://www.w3schools.com/sql/
- **Unit Testing:**
  - https://docs.pytest.org/en/stable/getting-started.html
  - https://docs.python.org/3/library/unittest.html#basic-example