Introduction

Problem Statement:

With millions of apps on the Google Play Store, manually analysing user reviews is *time-consuming* and *inefficient*. Users, developers, and marketers need an automated solution to quickly extract meaningful insights and make informed decisions on **App Performance** and **User Satisfaction**.

This project focuses on a Sentiment analysis application that uses Google Play Store reviews to provide insights about the Applications. The system comprises two main components:

1. Backend (Python with Flask)

The backend is built using **Flask** to serve APIs, and integrates several libraries like **transformers**, **google-play-scraper**, and **pyngrok** to perform sentiment analysis and data extraction.

- **Purpose:**
  The backend provides two key functionalities:
  - **Extracting reviews:** From a given Google Play Store app URL, it fetches all user reviews using the **google-play-scraper** library.
  - **Analysing sentiment:** Using the **Sentiment-RoBERTa Transformer model**, it determines whether reviews are **positive** or **negative**.
- **Working:**

- Users input the **Google Play Store app URL**.
- The app extracts the **app ID** from the URL and retrieves all the reviews after **NGROK** tunnel and **Flask API** with the help of **CORS** for cross platform is created for interacting with HTML.
- Reviews are cleaned and preprocessed:-
  Irrelevant columns like **reviewId, userImage, thumbUpCount, reviewCreatedVersion, at, replyContent, repliedAt** and **appVersion** are dropped.
- Then the reviews are passed to the sentiment analysis model, which evaluates each review to assign a **sentiment score**.
- The app calculates the **percentage of positive and negative sentiments** and provides a **decision**.

- ## Decision Logic:
  Based on the proportion of positive reviews:
  - **Positive ≥ 70%:** App is worth downloading.
  - **Positive between 50% and 70%:** App is good, but users should consider their preferences.
  - **Positive between 40% and 50%:** Mixed reception; personal preferences play a role.
  - **Positive < 40%:** App likely has significant issues.

2. Frontend (HTML + JavaScript)

The frontend provides a user-friendly interface for interacting with the sentiment analysis system.

- **Features:**
  - Users enter the Google Play Store app link in an input field.
  - Two buttons allow users to either:
    - **Generate reviews:** Fetch and display reviews as cards.
    - **Generate sentiment:** Analyze reviews and display the percentage of positive/negative reviews and a decision.
- **Working:**
  - The frontend sends **POST requests** to the Flask backend using **Fetch API**.
  - The backend processes the request and returns the results (reviews or sentiment analysis).
  - Results are dynamically displayed:
    - Reviews appear as cards showing the user's name and their review content.
    - Sentiment percentages and the final decision are displayed below.
- **Dynamic Updates:**
  Loading indicators ensure users see progress when reviews or sentiments are being fetched. Results are rendered dynamically with clean formatting and interactivity.

Overall Workflow

- **Input:** User provides a Google Play Store URL.
- **Process**:
  - Extract the app ID.
  - Fetch reviews using the app ID.
  - Preprocess reviews.
  - Perform sentiment analysis on the reviews.
  - Calculate sentiment percentages and provide a decision.
- **Output**: Sentiment analysis results, raw reviews, and a recommendation.

Analysis of the Application

- **Strengths**
  - **Seamless Integration of Backend and Frontend:**

    The project efficiently combines Python's powerful ML libraries with a user-friendly HTML/JavaScript frontend, ensuring smooth communication via Flask's API endpoints.

  - **State-of-the-Art Sentiment Model:**

    The use of the **Sentiment-RoBERTa model** ensures highly accurate sentiment classification, as this model is fine-tuned specifically for English sentiment analysis tasks.

  - **Real-Time Insights:**

The app provides real-time data processing and sentiment analysis, making it a practical tool for app developers or users evaluating an app.

- **Decision-Making Framework:**

The decision logic simplifies the interpretation of results by non-technical users, providing actionable recommendations based on sentiment trends.

- # Weaknesses

- **Dependence on the Google Play Store:**

If the app lacks reviews or the Google Play Store API changes, the system may fail to retrieve data.

- **Single Sentiment Metric:**

The analysis focuses only on binary sentiment classification (positive/negative). It does not account for neutral or mixed sentiments, which could add valuable nuance.

- **Processing Speed:**

Fetching and analysing a large number of reviews could become time-consuming, especially on systems with limited computational power.

- **Limited Language Support:**

The system only processes English-language reviews (Lang='en'), which limits its usability for non-English apps or regions.

- **Ngrok Dependency:**

While ngrok simplifies hosting, relying on it for production is not ideal due to stability and security concerns. Using free version, the **Public URL** is variable so It has to be changed every time in the **JavaScript** variable.

- **Applications**
  - **App Developers:**

    Developers can use this tool to monitor user feedback and make informed decisions about updates or bug fixes.

  - **App Reviewers:**

    Bloggers or content creators can analyze app performance and user satisfaction trends for their audience.

  - **Marketing Teams:**

    Marketing professionals can assess the general reception of their apps and design better promotional strategies.

Conclusion

This tool combines machine learning, web scraping, and web development to automate review analysis and sentiment prediction, offering insights for app developers. Future improvements in scalability and language support could make it a robust platform for app performance evaluation.