📑 Project Report: SMS Spam Classification

# 🔹 Objectives

The objective of this project is to extract SMS messages from backups, process them into structured datasets, and apply automated classification to identify spam and scam messages reliably. The aim is to reduce manual effort, ensure consistency, and create a scalable solution for handling large SMS datasets.

# 🔹 Parsing SMS Data from XML Backup

Purpose  
This step prepares the raw SMS dataset for classification by extracting messages from an XML backup file (e.g., from “SMS Backup & Restore”).

Workflow  
1. Read XML File → Loaded the backup file (Rishabh.xml).  
2. Message Type Mapping → Converted numeric message types (Inbox, Sent, Draft, etc.) into readable categories.  
3. Parse and Clean Data → Extracted sender, timestamp (converted to date/time), message body, and type.  
4. Default Spam Label → Assigned spam = 0 initially (to be updated later via classification).  
5. Output → Converted all messages into a structured Pandas DataFrame and displayed sample rows.

Error Handling  
- Handled invalid/missing dates gracefully.  
- Replaced bad encoding characters to prevent crashes.  
- Could fail if XML structure differs (e.g., <message> instead of <sms>).

# 🔹 SMS Classification

Purpose  
This step focuses on automatically classifying SMS messages into predefined categories such as:  
- Phishing  
- Smishing  
- Promotional  
- Loan/Financial Scam  
- Job Scam  
- Crypto/Investment Scam  
- No Spam

Workflow  
1. Load Dataset → CSV file containing SMS texts (message\_body).  
2. Define Categories → Prepared a set of seven classification labels.  
3. Classification Function → Used a language model to classify each SMS.  
4. Iterative Processing → Processed messages one by one, adding a short delay to avoid rate limits.  
5. Output → Exported results to annotated\_dataset.csv with predicted labels.

⚠️ Error Encountered  
The script hit an API rate limit error (429):  
  
Rate limit reached for requests per minute...  
Limit 3 requests per minute.  
  
This happened because the free tier allows only 3 requests/minute, but the dataset required many more queries.

# 🔹 Challenges Faced

- API rate limits restricted the number of SMS messages that could be classified per minute.  
- XML backup formats can vary, and any deviation in structure may cause parsing issues.  
- Some SMS messages may have missing or corrupted timestamps, requiring error handling.  
- Large dataset sizes increase processing time and require automation.

# 🔹 Applications

- Mobile spam filtering systems to protect users from scams and fraud.  
- Telecom providers analyzing SMS data for fraudulent activity detection.  
- Enterprise communication monitoring to prevent phishing or scam attempts.  
- Research projects involving natural language processing on SMS datasets.

# 🔹 Limitations

- Classification relies on external language models which may have usage limits.  
- Mixed or ambiguous SMS content can lead to misclassification.  
- Dataset parsing depends on the specific XML structure, which may not be universal.  
- Current setup does not support real-time SMS filtering.

# 🔹 Why Automated Labeling is Needed

- Volume of Data → A typical SMS backup may contain thousands of messages. Manually reading and labeling each one is impractical.  
- Human Error → Manual labeling is prone to mistakes and inconsistent category assignment.  
- Scalability → As the dataset grows, the effort required increases exponentially, making manual labeling infeasible.  
- Time Constraint → Labeling each message individually would take an enormous amount of time, which is not realistic for projects with deadlines.

# 🔹 Conclusion

This project shows how raw SMS data can be transformed into structured datasets and then automatically classified into spam and scam categories. By combining parsing and classification, the process becomes scalable and reliable. Manual labeling was avoided due to the high volume of data, inconsistency risks, and strict time constraints, making automation the practical choice.