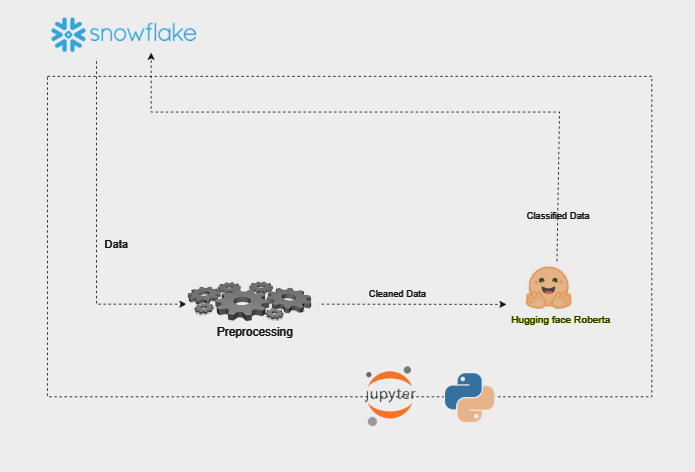
**Sentiment Analysis Project (Case Study- PiggyVest)**

Data Science Task Documentation

# Overview



# 1. Introduction

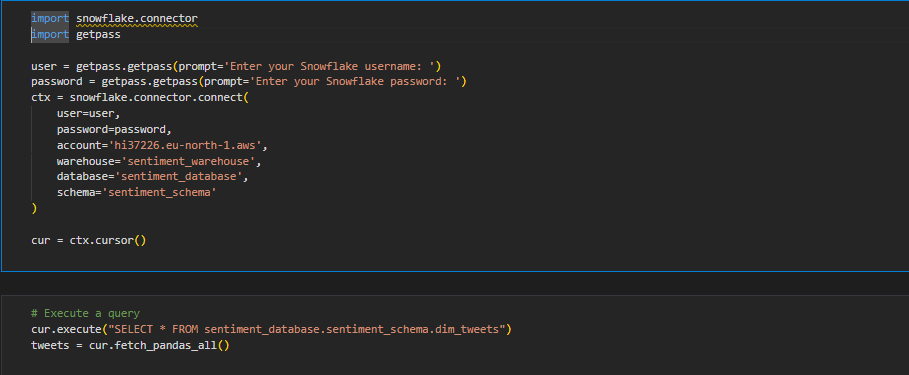
This project demonstrates the end-to-end process of developing a sentiment analysis model, starting from data collection from a Snowflake warehouse, to data description, data cleaning and preprocessing, model building, and finally deployment. The project utilizes several technologies such as Snowflake, Pandas, NLTK, spaCy, Huggingface Transformers, and Steamlit for deployment.

# 2. Loading the data

The data used in this project was scraped from Twitter, which involved extracting tweets related to #PiggyVest. This data collection process includes capturing relevant tweet text, metadata (such as timestamps and user information), and potentially other details like tweet engagement metrics. Read the complete data collection documentation [here](https://github.com/Chideraozigbo/sentiment-analysis-project?tab=readme-ov-file#system-architecture).

The first step in the process is collecting the necessary data for analysis. We use Snowflake's cloud-based data warehouse to store and retrieve the data. Using the Snowflake connector for Python, we query and extract raw data from our warehouse.

Here’s an example of how to collect data from Snowflake using the Python connector:



# 3. Data Description

Once the data is retrieved, the next step is to describe and understand its structure. This includes examining the columns, identifying data types, and checking for missing or erroneous values. The `pandas` library is used to perform data exploration and provide basic descriptive statistics.

The dataset contains 1363 records with 13 columns features, 'tweet\_id', 'user\_id', 'created\_at', 'text', 'url', 'mentions', 'lang', 'favourites', 'retweets', 'replies', 'quotes', 'view\_count', 'hashtags'.

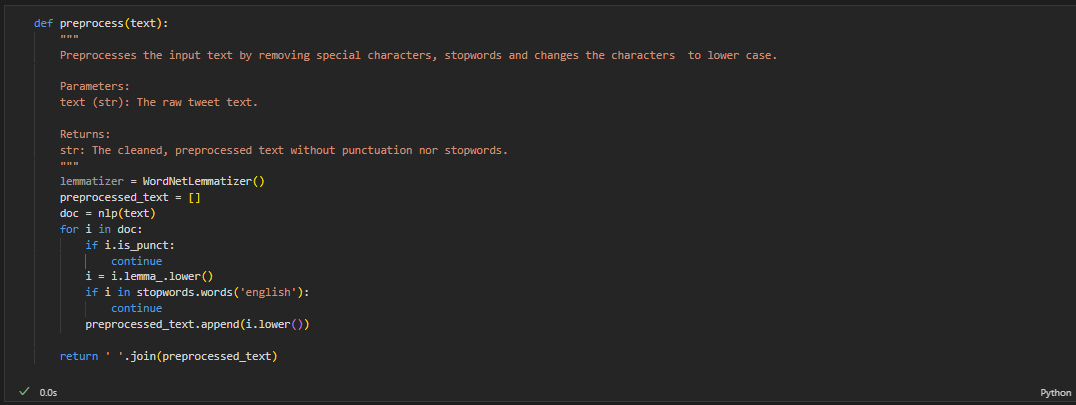
**Table 1.1 Dataset Description**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column | Data Type | Constraints | Description | Null Values |
| tweet\_id | INTEGER | PRIMARY KEY, NOT NULL | Unique identifier for the tweet | 0 |
| user\_id | INTEGER | FOREIGN KEY, NOT NULL | ID of the user who posted the tweet | 0 |
| created\_at | TIMESTAMP | NOT NULL | Timestamp when the tweet was created | 0 |
| text | TEXT | NOT NULL | Cleaned text of the tweet | 0 |
| url | TEXT | NULL | URL present in the tweet | 773 |
| mentions | TEXT | NULL | User mentions in the tweet | 1153 |
| lang | TEXT | NOT NULL | Language of the tweet | 0 |
| favorites | INTEGER | NOT NULL | Number of favorites the tweet received | 0 |
| retweets | INTEGER | NOT NULL | Number of retweets | 0 |
| replies | INTEGER | NOT NULL | Number of replies | 0 |
| quotes | INTEGER | NOT NULL | Number of quotes | 0 |
| views | INTEGER | NOT NULL | Number of views | 575 |
| hashtags | TEXT | NULL | Hashtags used in the tweet | 1204 |

# 4. Data Cleaning and Preprocessing

The next step is to clean and preprocess the data for sentiment analysis. This involves removing special characters, stopwords, and punctuation from the text, as well as lemmatizing the words to ensure consistency. We use a combination of the NLTK and spaCy libraries to perform these tasks.

Stopwords are words that are commonly found in sentences. They add little or no context to a sentence. Example: **is, I, be, and,** etc**.**

By removing common words and symbols, we reduce noise in the data, allowing the model to focus on the more meaningful and contextually significant parts of the text. This helps the model to better understand the underlying sentiment and improve overall performance.

The resulting dataset from this step included additional *'date\_created'*, *'processed\_text'* columns which would be used for the EDA.

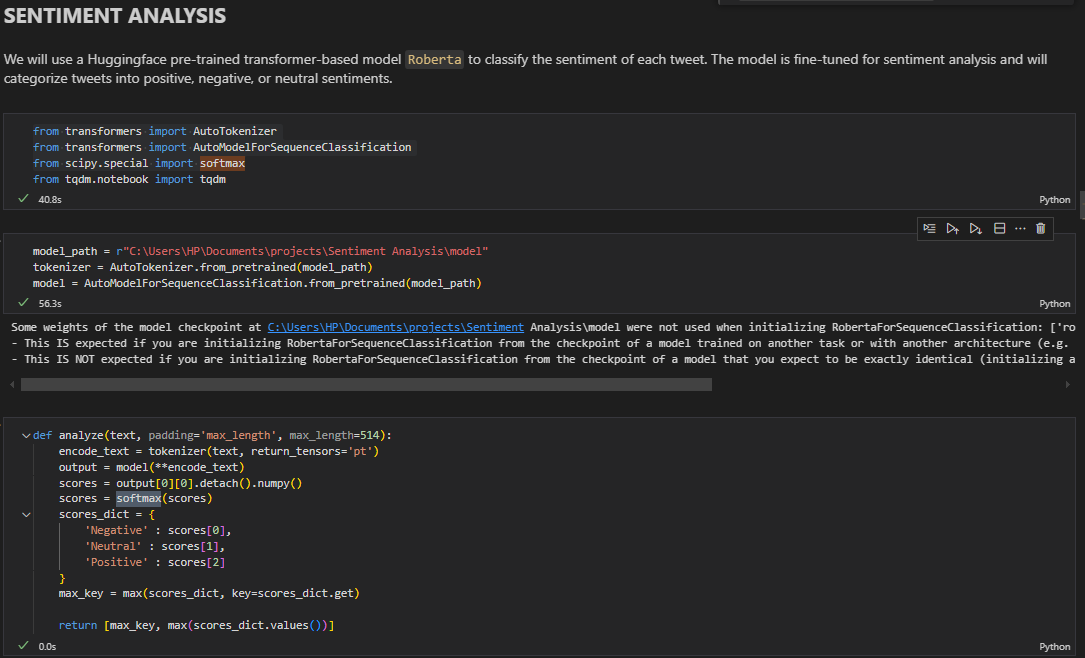
# 5. Sentiment Extraction

After cleaning the data, we used a pretrained model for sentiment analysis. We used Huggingface's Transformers library to leverage a pretrained model (e.g., DistilBERT) that is fine-tuned for sentiment classification. The tokenizer and model are used to process the text data and generate predictions.

This model which originally returns a dictionary containing sentiment probabilities for a text was modified to return a Sentiment based on the maximum probability, followed by its confidence score.

The implementation of the model involves 3 steps/functions

* **AutoTokenizer**: Transforms text into token, which are numerical representations of the texts that the model can process. It also handles tasks like padding and truncation to ensure consistent input size into the model.
* **AutoModelForSequenceClassification**: Takes the tokenized text and processes it through a neural network to produce logits. It’s specifically designed for tasks where the goal is to classify the text into predefined categories like sentiment in this case.
* **softmax**: Converts the raw model outputs (logits) into probabilities by normalizing them. This makes it easier to interpret the model's predictions, as the output will be a probability distribution over the possible sentiment classes, with values summing up to 1.

AutoTokenizer and AutoModelForSequenceClassification are embedded into the Tranformer model while the softmax is a scipy library function

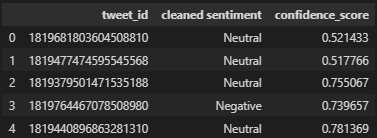
The result of this function is a dictionary in the following format:

{

‘tweet\_id’ : [cleaned sentiment, confidence\_score]

}

This dictionary is then converted into a DataFrame for easier manipulation and analysis. The resulting DataFrame is merged with the original dataset



The output dataset from this phase has additional *'cleaned sentiment'* and *'confidence\_score'* columns

# 6. Deployment

The final step is to deploy the sentiment analysis model as a web service. We use Streamlit, a lightweight web framework, to create an interactive web application that allows users to input text and receive sentiment predictions in real-time.

# 7. Technologies Used

The following technologies were used in this project:  
1. **Snowflake**: Cloud-based data warehouse used to store and retrieve the raw data.  
2. **Pandas**: Data manipulation and analysis library used for exploring and cleaning the data.  
3. **NLTK** and **spaCy**: Libraries for natural language processing, used for text preprocessing (lemmatization, stopword removal).  
4. **Huggingface Transformers**: Provides pretrained language models for performing sentiment analysis.  
5. **Streamlit**: Web framework used to deploy and create an interactive UI for the model.