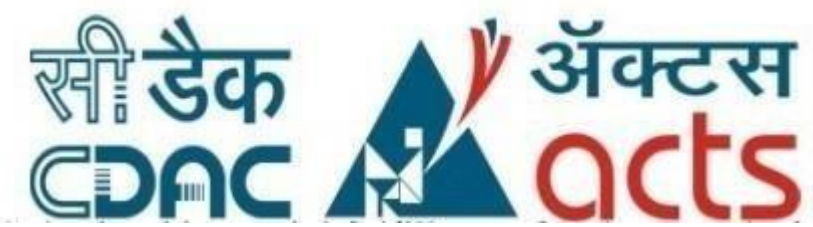
Project Report

Real-Time Data Streaming using Spark, Kafka, OpenAI, Elastic Search, Kibana



Submitted
In partial fulfilment
For the award of the Degree of

PG-Diploma in Big Data Analytics (PG-DBDA)

C-DAC, ACTS (Pune)

Guided By: Milind Khapse

Submitted By:

Pulkit Kushwaha	230940125038
Gaurav Bisht	230940125017
Yash Bhardwaj	230940125056
Akash Gurav	230940125001

Centre for Development of Advanced Computing(C-DAC), ACTS

(Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Mr. Milind Khapse** C-DAC ACTS, Pune for her constant guidance and helpful suggestion for preparing this project **Real-Time Data Streaming using Spark, Kafka, OpenAI, Elastic Search .** We express our deep gratitude towards her for inspiration, personal involvement, constructive criticism that she provided us along with technical guidance during the course of this project.

We take this opportunity to thank Head of the department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mrs. Namrata Ailawar**, Process Owner, for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P R (Program Head)** and **Mrs. Srujana Bhamidi** (Course Coordinator, PG-DBDA) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

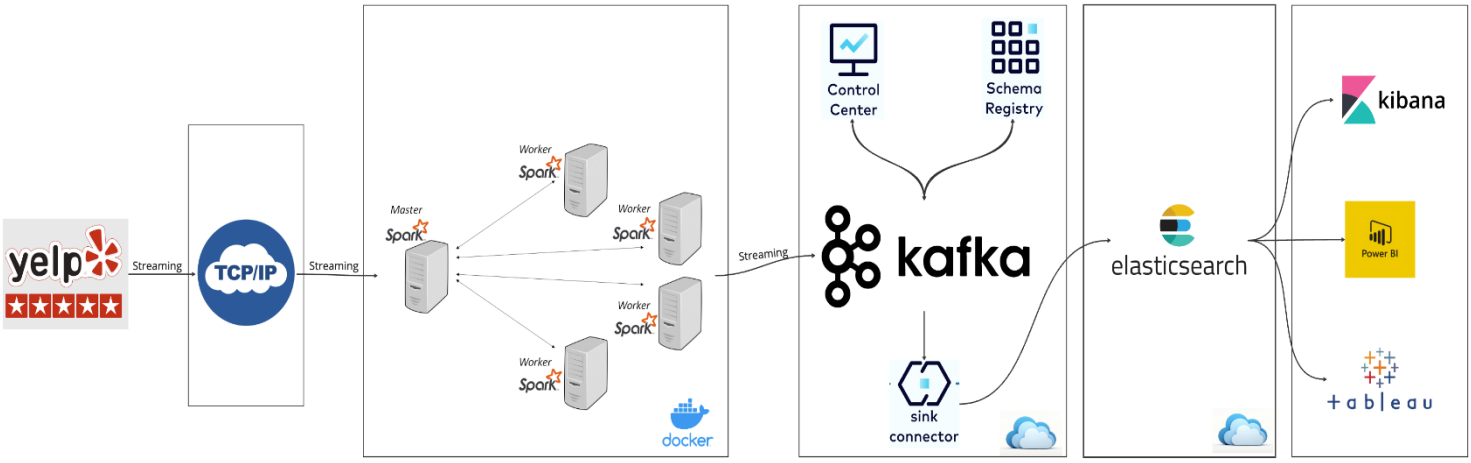
Also, our warm thanks to **C-DAC ACTS Pune**, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Pulkit Kushwaha	230940125038
Gaurav Bisht	230940125017
Yash Bhardwaj	230940125056
Akash Gurav	230940125001

ABSTRACT

In this project, we aim to perform sentiment analysis on customer reviews using Chat-GPT and Text-mining , a natural language processing model. We use real-time data streaming from yelp.com/dataset, which contains millions of reviews from different businesses. We use the following tools and technologies for our pipeline:

- **TCP/IP Socket**: This allows us to stream data over the network in chunks from the data source to our Spark cluster.
- **Docker Desktop**: This enables us to create a containerized environment for our Spark cluster, which consists of one master node and two worker nodes.
- **Apache Spark**: This is the core of our data processing engine, which handles the ingestion, transformation and analysis of the streaming data using Spark Streaming and Spark SQL.
- **OpenAI'gpt-3.5-turbo**: This is the model that we use to get the training data for text mining perform sentiment analysis on the reviews. It is a pre-trained generative model that we use to classify the reviews into positive, negative or neutral categories based on the tone and content of the text.
- **Text-mining:** plays a crucial role in sentiment analysis, which involves identifying and extracting sentiments or opinions expressed into positive, negative or neutral categories.
- **Confluent Kafka**: This is our cloud-based platform that provides a scalable and reliable messaging system for our data streams. It also offers various features such as Control Center and Schema Registry, which help us monitor and manage our Kafka streams and schemas.
- **Kafka Connect**: This is a tool that connects our Kafka cluster to our Elasticsearch cluster, which acts as a sink for our data streams.
- **Elasticsearch**: This is a distributed search and analytics engine that indexes and stores our data streams for fast and flexible querying.
- **Kibana**: This is a visualization and exploration tool that connects to our Elasticsearch cluster and allows us to create dashboards and charts to display the results of our sentiment analysis.



System Architecture

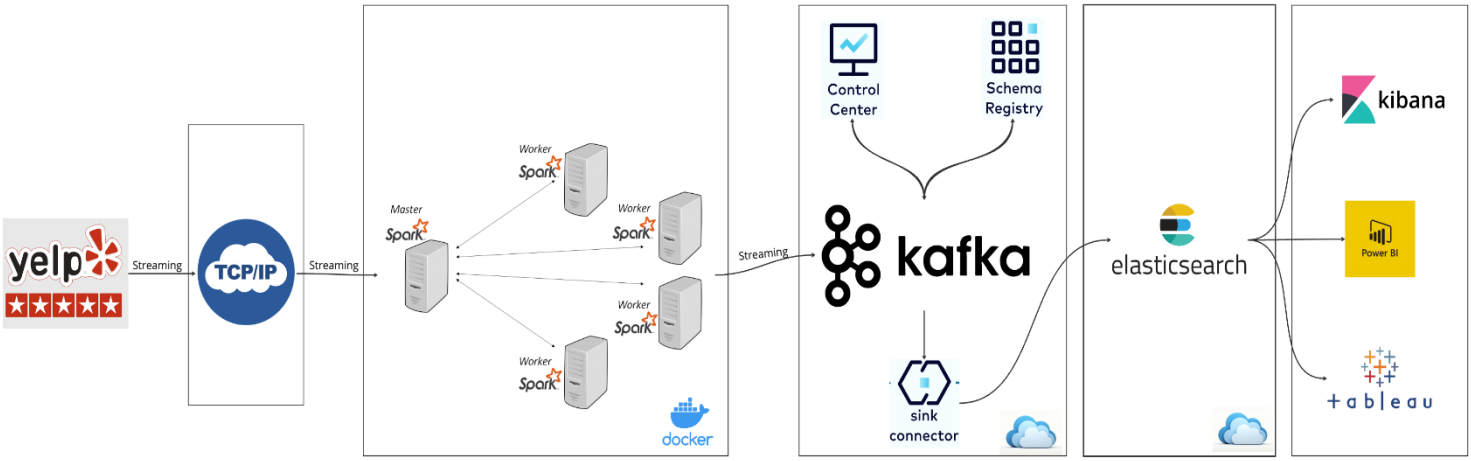
CONTENT

Introduction

In this project, we embark on a journey to delve into the realm of sentiment analysis on customer reviews leveraging the power of Chat-GPT and Text Mining techniques. Our goal is to extract valuable insights from a vast pool of customer feedback sourced from Yelp.com/dataset, which hosts millions of reviews spanning across various businesses.

Real-time Data Analysis:

Real-time data analysis involves processing and analyzing data as it is generated or received, enabling organizations to make timely decisions and take immediate action based on the insights derived from the data. Real-time data analysis enables organizations to respond quickly to changing conditions, detect and mitigate risks, optimize processes, and capitalize on emerging opportunities in today's fast-paced and data-driven business environment. By harnessing the power of real-time analytics, organizations can gain a competitive edge and drive innovation in their respective industries.



System Architecture

Methodology/ Techniques

Data Set:

This Project dataset containing reviews from a platform known Yelp. Each entry in the dataset represents a single review and contains the following fields:

1. **review_id (string):** A unique identifier for each review.
2. **user_id (string):** The identifier of the user who submitted the review.
3. **business_id (string):** The identifier of the business being reviewed.
4. **stars (float):** The rating given to the business by the user. This is typically on a scale of 1 to 5 stars.
5. **useful (float):** The number of users who found the review useful. This could be a measure of the review's helpfulness to other users.
6. **funny (float):** The number of users who found the review funny. This could indicate the review's entertainment value.
7. **cool (float):** The number of users who found the review cool. This could indicate the review's positivity or overall appeal.
8. **text (string):** The actual text content of the review. This field contains the written feedback or comments provided by the user.
9. **date (string):** The date and time when the review was submitted. This provides a timestamp for when the review was written.

This dataset appears to be well-structured and suitable for various types of analysis, including sentiment analysis and recommendation systems. It contains rich information about user opinions and preferences, which can be valuable for understanding customer sentiment, identifying trends, and making data-driven business decisions.

Approach and Methodology/ Techniques

1.Transmission Control Protocol/Internet Protocol (TCP/IP) sockets are a fundamental technology for network communication in computer networks, including the internet. In the context of streaming data to a Spark cluster, TCP/IP sockets facilitate the transfer of data chunks from a data source to the Spark cluster over the network.

Here's how TCP/IP sockets work in this scenario:

1. **Socket Creation:** In the data source (e.g., a server or a device), a TCP/IP socket is created to establish a communication endpoint. This socket is identified by an IP address and a port number.
2. **Connection Establishment:** The Spark cluster, which acts as the receiver, listens for incoming connections on a specified port. When the data source wants to send data, it establishes a TCP connection with the Spark cluster by specifying the IP address and port number of the Spark cluster's socket.
3. **Data Transfer:** Once the connection is established, data is sent from the data source to the Spark cluster in the form of data chunks or packets. TCP/IP ensures reliable, ordered, and error-checked delivery of these data packets over the network.
4. **Buffering and Flow Control:** TCP/IP includes mechanisms for buffering and flow control to manage the transfer of data between the data source and the Spark cluster efficiently. This ensures that data is transferred at an optimal rate and prevents overwhelming the receiver with data.
5. **Socket Closure:** Once all data has been transferred, or when the communication session is complete, the TCP connection is terminated, and the sockets are closed.

By leveraging TCP/IP sockets, data can be streamed in real time or near real time from a data source to a Spark cluster, enabling continuous processing and analysis of streaming data. This facilitates tasks such as real-time analytics, monitoring, and alerting, where timely insights are crucial for decision-making.

2.Docker Desktop: is a tool that allows users to create, manage, and run containerized applications on their local machines. In the context of setting up a Spark cluster, Docker Desktop enables the creation of a containerized environment where Spark can be deployed with ease. Here's how Docker Desktop facilitates the

creation of a containerized Spark cluster with one master node and two worker nodes:

1. **Containerization:** Docker Desktop uses containerization technology to package applications and their dependencies into lightweight, portable containers. Each container encapsulates the application along with its runtime environment, libraries, and dependencies, ensuring consistency and portability across different environments.
2. **Dockerfile:** To create a containerized Spark cluster, a Dockerfile is written to define the specifications of the Spark cluster environment. The Dockerfile specifies the base image, installation of Spark, configuration of master and worker nodes, and any additional dependencies or configurations required for the Spark cluster.
3. **Building Docker Images:** Once the Dockerfile is defined, Docker Desktop is used to build Docker images based on the specifications provided in the Dockerfile. These Docker images serve as blueprints for creating container instances of the Spark cluster.
4. **Container Orchestration:** Docker Desktop provides tools for orchestrating containers, including Docker Compose or Docker Swarm. With Docker Compose, a YAML file is used to define the services (i.e., containers) comprising the Spark cluster, including the master node and worker nodes. Docker Compose then orchestrates the deployment and management of these containers.
5. **Networking and Communication:** Docker Desktop configures networking between containers to facilitate communication within the Spark cluster. This allows the master node to communicate with the worker nodes and distribute tasks for parallel processing.
6. **Scalability and Flexibility:** Docker Desktop enables easy scalability of the Spark cluster by adding or removing worker nodes as needed. This flexibility allows users to adjust the computing resources allocated to the Spark cluster based on workload requirements.
7. **Isolation and Portability:** By encapsulating the Spark cluster within containers, Docker Desktop provides isolation from the underlying host system and ensures portability across different environments. This enables users to run the Spark cluster consistently across development, testing, and production environments.

Overall, Docker Desktop simplifies the process of creating and managing containerized environments for Spark clusters, providing developers and data engineers with a convenient and efficient way to deploy and scale Spark applications.

3.Apache Spark is a powerful open-source distributed computing system designed for processing large-scale data sets and performing complex data analytics tasks. In the context of real-time data analysis, Spark serves as the core data processing engine, providing capabilities for ingesting, transforming, and analyzing streaming data. Here's an overview of how Apache Spark facilitates real-time data processing using Spark Streaming and Spark SQL:

1. **Distributed Computing:** Spark is built on the concept of distributed computing, allowing it to efficiently process large volumes of data across a cluster of machines. It distributes data processing tasks across multiple nodes in the cluster, enabling parallel execution and high throughput.
2. **Spark Streaming:** Spark Streaming is an extension of the core Spark API that enables real-time processing of streaming data. It ingests data streams from various sources such as Kafka, Flume, or TCP sockets, and processes the data in micro-batch intervals. Each micro-batch of data is treated as a Resilient Distributed Dataset (RDD), allowing Spark to apply batch processing operations such as transformations and aggregations.
3. **Spark SQL:** Spark SQL is a module in Spark that provides support for structured data processing and SQL queries. It allows users to query and analyze streaming data using SQL syntax, making it easier to express complex data processing logic. Spark SQL seamlessly integrates with other Spark components, enabling users to combine SQL queries with DataFrame and Dataset APIs for advanced analytics.
4. **Data Ingestion:** Spark provides connectors and libraries for ingesting streaming data from various sources, including Apache Kafka, Apache Flume, Amazon Kinesis, TCP sockets, and more. These connectors enable Spark to seamlessly integrate with different data streaming platforms and systems, allowing users to ingest data in real time from diverse sources.
5. **Transformation and Analysis:** Spark offers a rich set of transformation and analysis functions that can be applied to streaming data in real time. These functions include map, filter, reduce, join, windowing operations, and more. Users can perform complex data transformations, aggregations, and analytics on streaming data using familiar programming constructs and APIs provided by Spark.
6. **Fault Tolerance and Scalability:** Spark provides built-in fault tolerance and scalability features, ensuring reliable and efficient processing of streaming data. It replicates data partitions across multiple nodes in the cluster and maintains lineage information to recover from failures automatically. Spark's scalability allows it to scale seamlessly from small-scale deployments to large-scale production clusters, handling terabytes or petabytes of data with ease.

Overall, Apache Spark serves as a robust and versatile platform for real-time data processing, enabling organizations to analyze streaming data streams, extract actionable insights, and make informed decisions in real time. Its flexible architecture, rich feature set, and seamless integration with other data processing frameworks make it a popular choice for real-time analytics applications.

4.OpenAI'gpt-3.5-turbo: Sentiment analysis using ChatGPT, which is based on OpenAI's GPT architecture, involves leveraging the model's natural language understanding capabilities to classify text into different sentiment categories such as positive, negative, or neutral. Here's how you can perform sentiment analysis using ChatGPT:

1. **Data Preparation:** Gather a dataset of text samples with their corresponding sentiment labels (positive, negative, neutral). Ensure that the dataset is properly labeled for training and evaluation purposes. You may use existing sentiment analysis datasets or create your own dataset by manually labeling text samples.
2. **Fine-Tuning ChatGPT:** Fine-tune the ChatGPT model on the sentiment analysis dataset. This involves providing the model with input-output pairs (text samples and their corresponding sentiment labels) and fine-tuning its parameters to minimize prediction errors. You can fine-tune ChatGPT using techniques such as transfer learning, where you start with pre-trained weights and fine-tune them on your specific task.
3. **Model Deployment:** Once the fine-tuned ChatGPT model achieves satisfactory performance, deploy it to a production environment where it can classify sentiment in real-time. This may involve integrating the model into an application or service using APIs or libraries provided by OpenAI or other deployment frameworks.

Use the deployed ChatGPT model to perform sentiment analysis on new text samples. Input the text sample into the model, and the model will output the predicted sentiment category (positive, negative, or neutral) based on its learned patterns and context understanding.

5.Text mining: is a fundamental component of sentiment analysis, enabling the extraction of sentiments or opinions from textual data and categorizing them into positive, negative, or neutral categories. Here's how text mining contributes to sentiment analysis:

1. **Text Preprocessing:** Text mining techniques are applied to preprocess textual data before sentiment analysis. This includes tasks such as removing punctuation, converting text to lowercase, tokenization (splitting text into words or tokens), removing stopwords, and stemming or lemmatization (reducing words to their root form). Preprocessing ensures that the text data is clean and standardized for analysis.
2. **Feature Extraction:** Text mining methods are used to extract relevant features from the preprocessed text data. These features may include word frequency counts, TF-IDF (term frequency-inverse document frequency) scores, n-grams (sequences of adjacent words), and word embeddings (dense vector representations of words
3. **Sentiment Lexicons:** Text mining techniques are employed to create sentiment lexicons or dictionaries containing words and their associated sentiment scores (positive, negative, or neutral). These lexicons serve as a reference for sentiment analysis algorithms to determine the sentiment polarity of words in the text. Lexicon-based methods assign sentiment scores to text based on the presence of sentiment-bearing words and their associated scores.
4. **Machine Learning Models:** Text mining facilitates the training of machine learning models for sentiment analysis. Supervised learning algorithms such as support vector machines (SVM), logistic regression, and neural networks are commonly used to classify text into sentiment categories based on features extracted from the text data. Text mining provides the necessary preprocessing and feature extraction steps to prepare the data for training these models.
5. **Sentiment Analysis Algorithms:** Text mining algorithms are specifically designed for sentiment analysis tasks. These algorithms analyze the text data to detect sentiment-bearing words, phrases, or patterns and classify the overall sentiment expressed in the text as positive, negative, or neutral. Sentiment analysis algorithms leverage text mining techniques for preprocessing, feature extraction, and sentiment classification.

Overall, text mining serves as the foundation for sentiment analysis by providing the necessary tools and techniques to process, analyze, and categorize textual data according to the sentiments expressed within. It enables organizations to gain valuable insights into customer opinions, brand perception, and market trends, ultimately empowering data-driven decision-making processes.

6.Confluent Kafka Cloud: is a cloud-based platform that offers Apache Kafka as a managed service. Here's a breakdown of its key features:

1. **Scalable and Reliable Messaging System:** Confluent Kafka Cloud provides Apache Kafka as a service, offering a scalable and reliable messaging system for handling data streams. Kafka's distributed architecture allows it to scale horizontally, making it suitable for ingesting and processing large volumes of data in real-time.
2. **Publish, Subscribe, and Process Data Streams:** With Kafka Cloud, users can publish data to Kafka topics, subscribe to topics to consume data, and process data streams in real-time. Kafka's pub/sub model enables decoupling of data producers and consumers, facilitating asynchronous communication and data processing.
3. **Control Center:** Confluent Kafka Cloud includes Control Center, a web-based tool that provides monitoring and management capabilities for Kafka clusters. Control Center allows users to monitor cluster health, track message throughput and latency, and troubleshoot issues. It also offers features for managing topics, partitions, and consumer groups.
4. **Schema Registry:** Schema Registry is a feature of Confluent Kafka Cloud that helps in managing the schemas of data streams. It ensures compatibility and consistency of data schemas by storing and versioning schemas for Kafka topics. Schema Registry enables producers and consumers to serialize and deserialize data in a consistent format, facilitating interoperability and data governance.

Overall, Confluent Kafka Cloud simplifies the deployment and management of Apache Kafka clusters in the cloud, providing users with a scalable and reliable messaging platform for building real-time data pipelines. With features like Control Center and Schema Registry, users can effectively monitor, manage, and ensure the compatibility of their data streams, making it easier to develop and maintain Kafka-based applications.

7.Elasticsearch Cloud: is a managed service that offers Elasticsearch, a distributed search and analytics engine, as a service in the cloud. Here's an overview of its key features:

1. **Distributed Search and Analytics Engine:** Elasticsearch is a powerful search and analytics engine that allows you to index, store, and analyse large volumes of data in real-time. It supports full-text search, aggregations, geospatial search, and more, making it suitable for a wide range of use cases, including log analysis, monitoring, and business analytics.
2. **Scalability:** Elasticsearch Cloud offers scalability, allowing you to scale your cluster up or down based on your workload and storage requirements. You can easily add or remove nodes from your cluster to accommodate changes in data volume and query load, ensuring that your Elasticsearch cluster can handle growing data streams.
3. **High Availability:** Elasticsearch Cloud provides high availability features to ensure that your data is always accessible and your applications remain operational. It replicates data across multiple nodes in the cluster, automatically handling node failures and ensuring that data is not lost. Additionally, it supports features like shard allocation awareness and zone awareness to improve fault tolerance and resilience.
4. **Security Features:** Elasticsearch Cloud offers security features to protect your data and cluster from unauthorized access and malicious attacks. It supports encryption in transit and at rest, role-based access control (RBAC), and integration with identity providers such as Active Directory and LDAP. You can also enable features like audit logging and field- and document-level security to enforce data protection policies.

Overall, Elasticsearch Cloud provides a robust and scalable platform for storing and querying data streams generated by Kafka. It offers advanced search and analytics capabilities, high availability, and security features, making it suitable for building real-time data analytics applications and dashboards. By leveraging Elasticsearch Cloud, organizations can gain valuable insights from their data and make informed decisions to drive business growth and innovation.

8.Kibana: is a visualization and exploration tool that integrates with Elasticsearch. It allows you to create interactive dashboards, visualizations, and reports based on the data stored in Elasticsearch. With Kibana, you can explore and analyze your data, including the results of sentiment analysis performed on text data stored in Elasticsearch. Kibana provides a user-friendly interface for creating custom dashboards and charts to visualize the insights derived from your data.

Implementation

Device Specifications

Device name:	Pulkit-laptop
Processor:	12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHz
Installed RAM:	16.0 GB (15.7 GB usable)
Device ID:	F05A65BB-D22A-4E0C-AD47-949981E08928
System type:	64-bit operating system, x64-based processor
OS Edition:	Windows 11 Home Single Language
Version:	23H2
OS build:	22631.3155
Serial number:	PF4A1YSQ
GPU:	Intel(R) Iris(R) Xe Graphics 8GB

TCP/IP Socket

```
import socket
def send_data_over_socket(file_path, host='spark-master', port=9999,chunk=3):
    # Create a socket object
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((host, port))
    s.listen(1)
    print(f" Listening for connections on {host}:{port}")
```

Apache Spark

Reading from socket to spark

```
stream_df = (spark.readStream.format("socket")
             .option("host", "0.0.0.0")
             .option("port", 9999)
             .load()
             )

stream_df = stream_df.select(from_json(col('value'), schema).alias("data")).select(("data.*"))
```

Data Preparation

Preparing data for sentiment analysis with OpenAI model Chat-gpt 3.5 turbo

```
def sentiment_analysis1(comment,row) -> str:
    if comment:
        #openai.api_key = config['openai']['api_key']
        if row == 1:
            client = OpenAI(api_key=[redacted])
        elif row==2:
            client=OpenAI(api_key='[redacted]')
        #elif row==2:
        #    client=OpenAI(api_key="")
        else:
            client=OpenAI(api_key='[redacted]')

        completion = client.chat.completions.create(
        model="gpt-3.5-turbo",
            messages= [
                {
                    "role": "system",
                    "content": """
                        You're a machine learning model with a task of classifying comments into POSITIVE, NEGATIVE, NEUTRAL.
                        You are to respond with one word from the option specified above, do not add anything else.
                        Here is the comment:

                        {comment}
                    """.format(comment=comment)
                }
            ]
        )
        return completion.choices[0].message.content
    return "Empty"
```

From ChatGPT we prepare the training data-set (Y_train, Y_test) for text mining

Text-mining

```
import pandas as pd
import json
import joblib
data1=pd.read_json(r'D:/project-1/train.json')
data2=data1['value']
# Convert string representations of dictionaries to dictionaries
dic = [json.loads(d) if isinstance(d, str) else d for d in data2]

# Convert list of dictionaries to DataFrame
data = pd.DataFrame(dic)
text=data['text']
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer(max_features = 1000)
X = tf.fit_transform(text).toarray()
X = pd.DataFrame(X,columns=tf.get_feature_names_out())
y = data.iloc[:, -1]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y , test_size=0.3, random_state=123)

from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))

joblib.dump(clf, 'model1.pkl')
```

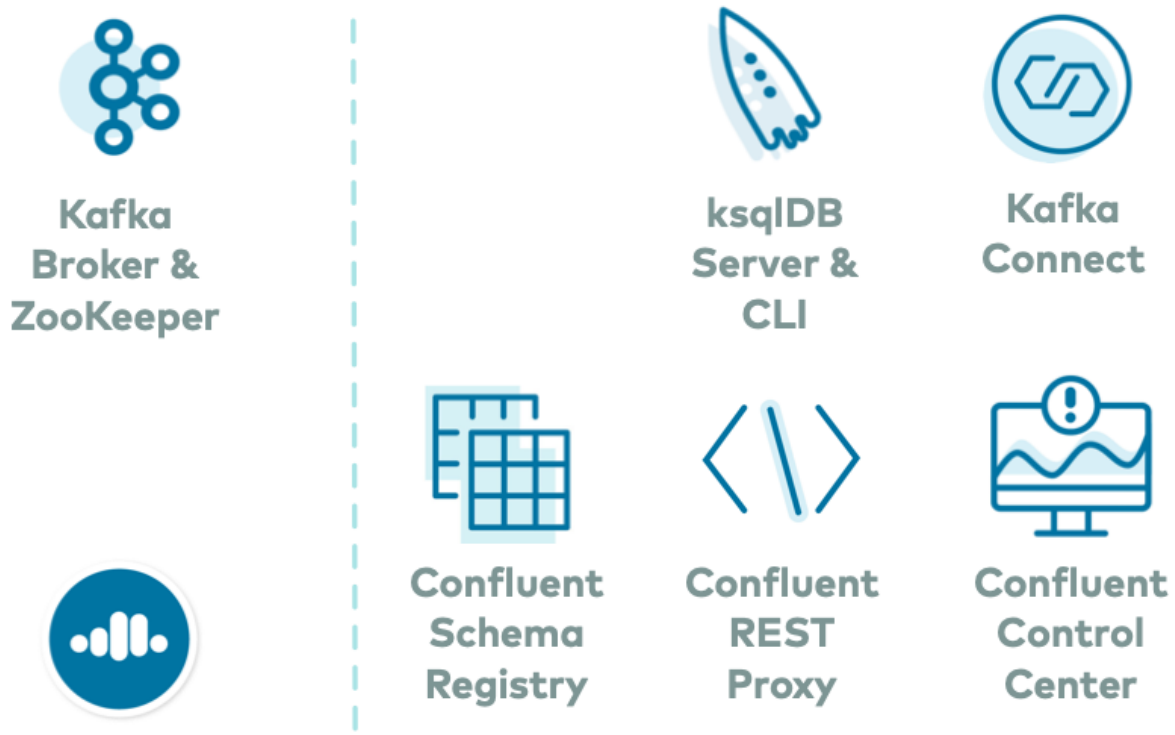
Results-

```
In [37]: runfile('D:/stat/untitled1.py', wdir='D:/stat')
MultinomialNB Accuracy: 0.8666666666666667

In [38]:
```

Sending data stream to confluent kafka cloud

```
kafka_df = stream_df.selectExpr("CAST(review_id AS STRING) AS key", "to_json(struct(*)) AS value")

query = (kafka_df.writeStream
        .format("kafka")
        .option("kafka.bootstrap.servers", config['kafka']['bootstrap.servers'])
        .option("kafka.security.protocol", config['kafka']['security.protocol'])
        .option('kafka.sasl.mechanism', config['kafka']['sasl.mechanisms'])
        .option('kafka.sasl.jaas.config',
                'org.apache.kafka.common.security.plain.PlainLoginModule required username="{username}" '
                'password="{password}";'.format(
                    username=config['kafka']['sasl.username'],
                    password=config['kafka']['sasl.password']
                ))
        .option('checkpointLocation', '/tmp/checkpoint')
        .option('topic',topic)
        .start()
        .awaitTermination()
    )
```

Confluent Kafka



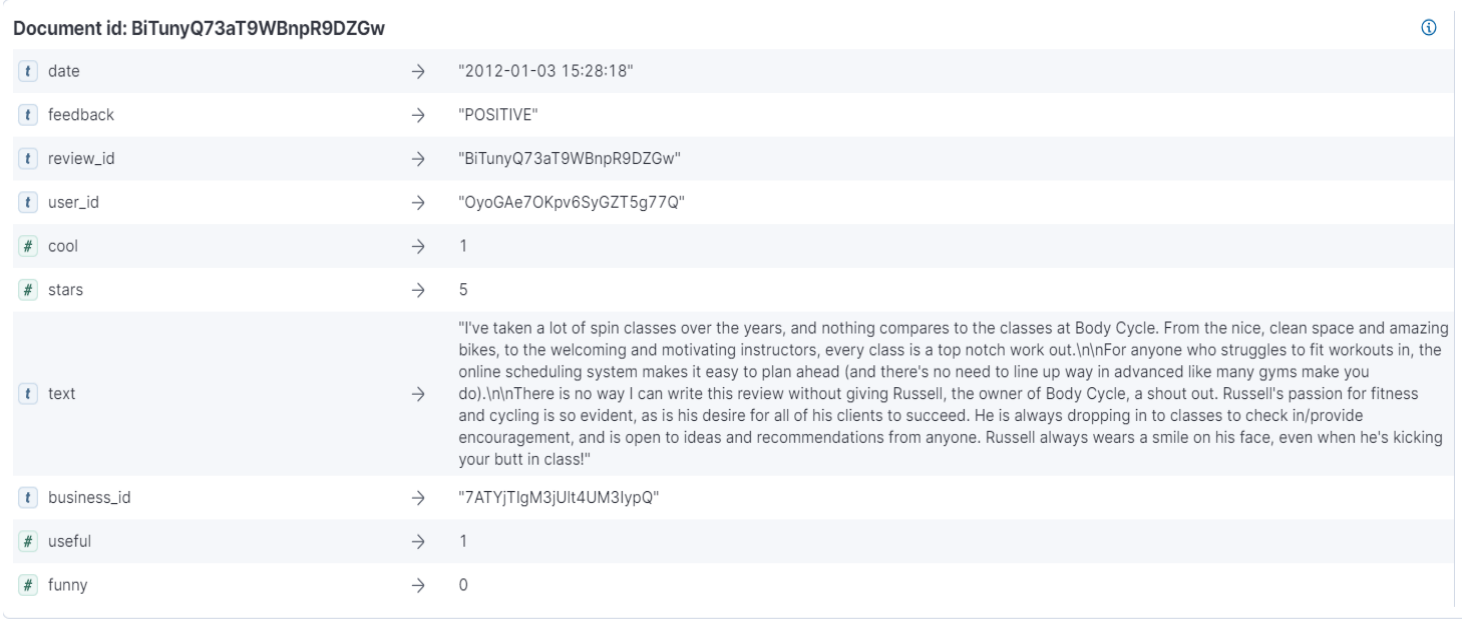
Confluent Schema Registry: is a tool that we use to convert the streaming data from Json format to Avro format at the source, and from Avro format to Json format at the sink. This allows us to have a consistent and efficient data schema across different applications and platforms.

Confluent Control Centre: Control Center allows users to monitor cluster health, track message throughput and latency, and troubleshoot issues. It also offers features for managing topics, partitions, and consumer groups.

Kafka connect: we use its "Elasticsearch service sink" to connect to Elasticsearch Cloud.

Elasticsearch Cloud

Elasticsearch is a powerful search and analytics engine that allows you to index, store, and analyse large volumes of data in real-time. It supports full-text search, aggregations, geospatial search, and more, making it suitable for a wide range of use cases, including log analysis, monitoring, and business analytics.



Elasticsearch stores the incoming data as **Documents.**

Elasticsearch query

```
GET customer_review_topic/_search
{
  "query":{
    "match_all": {}
  }
}
GET customer_review_topic/_count
{
  "query":{
    "match_phrase": {
      "stars": "5"
    }
  }
}
GET customer_review_topic/_count
{
  "query":{
    "match_phrase": {
      "feedback": "negative"
    }
  }
}
GET customer_review_topic/_count
{
  "query":{
    "match_phrase": {
      "feedback": "neutral"
    }
  }
}
```

results

->

```
{
  "count": 45,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  }
}
```

->

```
{
  "count": 19,
  "_shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  }
}
```

Kibana:

DashBoard:


Number of Count for types of Feedback
Count of count
Top 3 values of feedback.keyword
Pie Chart for number of Stars
Count of each type of feedback for every Star rating
Count of records
stars
Distribution of Feedback
NEGATIVE
22%
NEUTRAL 1%
POSITIVE 76%
Count of Feedback for different Usefulness
Count of records
Count of Feedback for different Funny Rating
Count of records

Conclusion:

We learn:

- Setting up data pipeline with TCP/IP socket.
- Real-time data streaming with Apache Kafka.
- Data processing techniques with Apache Spark.
- Realtime sentiment analysis with OpenAI ChatGPT.
- Sentiment analysis using text-mining.
- Synchronising data from Kafka to Elasticsearch.
- Indexing and Querying data on Elasticsearch.
- Data visualisation using Kibana.

References:

- ☑ Customer Reviews Dataset:
  https://www.yelp.com/dataset/
- ☑ Confluent Cloud Docs:
  https://docs.confluent.io/home/overview.html
- ☑ Elasticsearch Documentation:
  https://www.elastic.co/guide/en/elasticsearch/reference/current/docs.html
- ☑ Docker Compose Documentation:
  https://docs.docker.com/compose/
- ☑ Apache Kafka Official Site:
  https://kafka.apache.org/
- ☑ Apache Spark Official Site:
  https://spark.apache.org/
- ☑ OpenAI Chat GPT
  https://stackoverflow.com/questions/75774873/openai-chatgpt-gpt-3-5-api-error-this-is-a-chat-model-and-not-supported-in-t