DATA ANALYTICS

PUBLIC TRANSPORTATION EFFICIENCY

Sentiment Analysis of Student Feedback Using Machine Learning

1. Predicting Service Disruptions:

a. Data Collection: Gather historical data related to your transportation system, including information on routes, schedules, maintenance records, and past disruptions. Real-time data such as weather conditions, traffic reports, and GPS data can also be valuable.

b. Feature Engineering:Identify relevant features that could impact service disruptions, such as weather patterns, traffic congestion, holidays, or special events. You may also need to preprocess and clean the data to make it suitable for machine learning.

c. Model Selection: Choose an appropriate machine learning model for prediction. Time-series models like ARIMA or advanced algorithms like Random Forests, Gradient Boosting, or neural networks can be considered.

d. Training and Validation: Split your data into training and validation sets, and train the model on historical data. Use appropriate metrics (e.g., accuracy, F1-score, or AUC-ROC) to evaluate the model's performance.

e. Real-time Data Integration: To make predictions in real-time, integrate the model with live data streams. Continuous model updating and retraining are important to adapt to changing conditions.

f. Alerts and Notifications: Implement an alerting system that notifies relevant stakeholders (e.g., operators, passengers) when the model predicts a potential disruption. This can help in proactive planning and communication.

2. Analyzing Passenger Sentiment from Feedback:

a. Data Collection: Gather passenger feedback data from various sources such as surveys, social media, or customer service interactions. You can use natural language processing (NLP) techniques to process and analyze text data.

b.Text Preprocessing: Clean and preprocess the text data by removing stopwords, stemming/lemmatizing, and handling special characters. This step prepares the text data for sentiment analysis.

c. Sentiment Analysis: Apply sentiment analysis techniques to classify each passenger comment as positive, negative, or neutral sentiment. You can use pre-trained models or train your own sentiment classifier.

d. Topic Modeling: Use topic modeling algorithms (e.g., Latent Dirichlet Allocation, Non-Negative Matrix Factorization) to identify common themes or topics within the feedback data. This can provide insights into specific issues passengers are facing.

e. Dashboard and Reporting: Create a dashboard or reporting system that summarizes passenger sentiment trends and feedback topics. Visualizations can help transportation authorities quickly grasp the overall sentiment and specific issues.

f. Actionable Insights: Use the sentiment analysis results and feedback topics to drive improvements in service quality, address specific concerns, and make data-driven decisions to enhance passenger satisfaction.

g. Continuous Feedback Loop: Regularly collect and analyze passenger feedback to monitor changes in sentiment over time. This 3.METHODOLOGY

The presented methodology classiﬁes sentiment polarity

as positive, negative and neutral. The processes workﬂow is

shown in Fig. 1 and is further described in the following

subsections.

A. Dataset Description

The dataset used in this paper comprises of 1230 comments

extracted from our institutes educational portal. The dataset

was manually labeled with sentiment polarity labels {positive,

negative, neutral}. Table I shows few examples of student

comments.

B. Preprocessing

Student feedback data represents an unstructured text. To

extract useful information from the unstructured text, several

preprocessing steps are applied to remove spelling errors,

grammatical mistakes, URLs, etc. from the text. During pre-

processing stage, the following tasks were performed using

Python’s NLTK [12] library was used to perform preprocessing.

1) Punctuations: Punctuations, numbers and other special

characters were removed as these characters do not carry

useful information related to sentiment analysis.

2) Tokenization: Tokenization is the process of splitting text

stream into a list of words.

3) Case Conversion: After tokenization, words were trans-

formed into lower case.

4) Stop words: : In natural language processing, words that

are frequently used such as helping verbs, prepositions,

articles are termed as stop-words. Stop-words generally

do not provide any useful information and therefore were

removed from the feedback text.

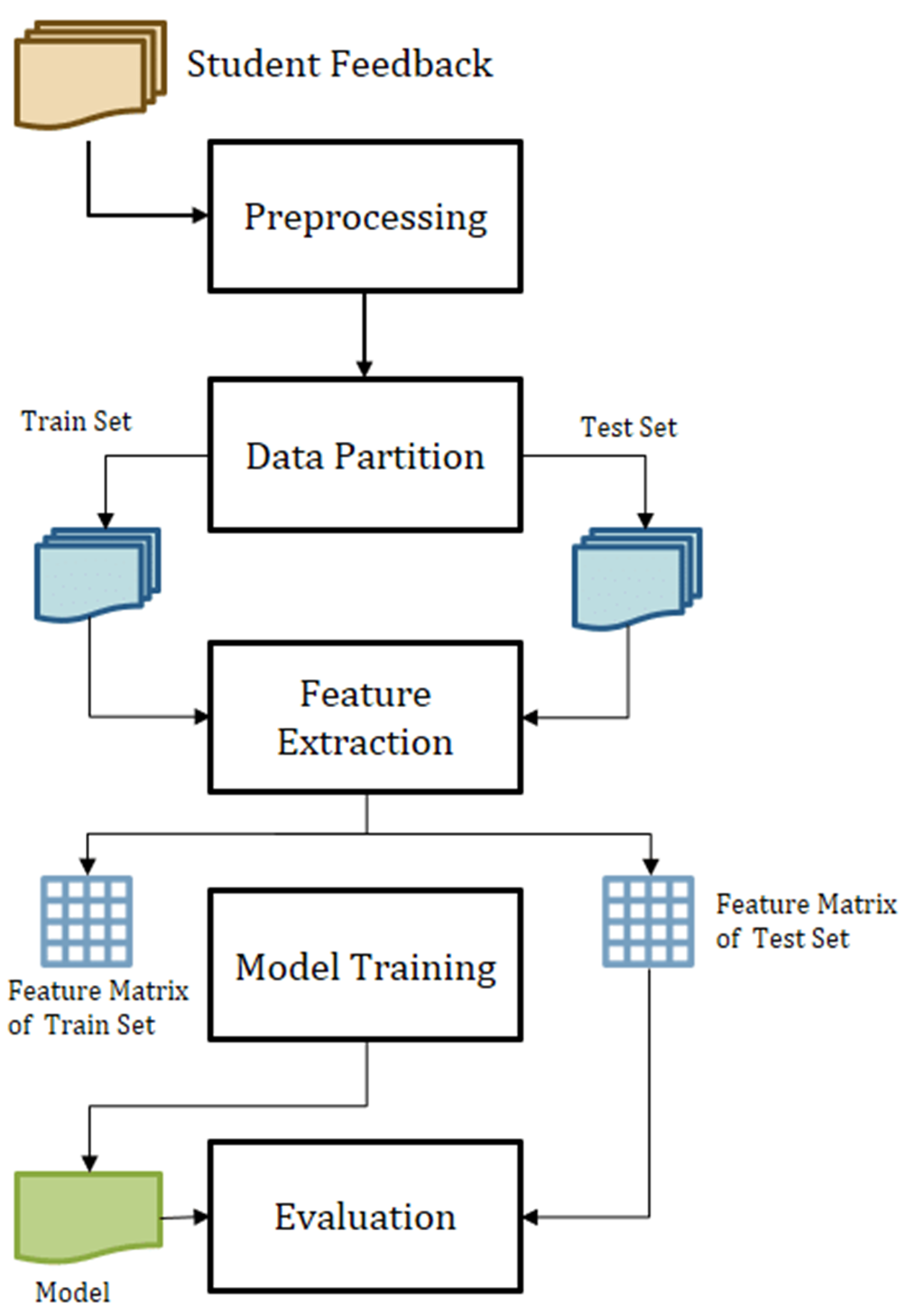
C. Data Partition

For training and evaluation purposes, the manually labeled

dataset of students feedback, as shown in Table I, was ran-

domly split into train set and test set. 70% of the dataset

DIAGRAM:



The presented methodology classiﬁes sentiment polarity

as positive, negative and neutral. The processes workﬂow is

shown in Fig. 1 and is further described in the following

subsections.

A. Dataset Description

The dataset used in this paper comprises of 1230 comments

extracted from our institutes educational portal. The dataset

was manually labeled with sentiment polarity labels {positive,

negative, neutral}. Table I shows few examples of student

comments.

B. Preprocessing

Student feedback data represents an unstructured text. To

extract useful information from the unstructured text, several

preprocessing steps are applied to remove spelling errors,

grammatical mistakes, URLs, etc. from the text. During pre-

processing stage, the following tasks were performed using

Python’s NLTK [12] library was used to perform preprocess-

ing.

1) Punctuations: Punctuations, numbers and other special

characters were removed as these characters do not carry

useful information related to sentiment analysis.

2) Tokenization: Tokenization is the process of splitting text

stream into a list of words.

3) Case Conversion: After tokenization, words were trans-

formed into lower case.

4) Stop words: : In natural language processing, words that

are frequently used such as helping verbs, prepositions,

articles are termed as stop-words. Stop-words generally

do not provide any useful information and therefore were

removed from the feedback text.

C. Data Partition

For training and evaluation purposes, the manually labeled

dataset of students feedback, as shown in Table I, was ran-

domly split into train set and test set. 70% of the dataset

Fig. 1. Methodology of Student Feedback Sentiment Analysis

D. Feature Extractionfeedback loop is crucial for ongoing service improvement

PROGRAM

Importing Libraries

In [1]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import plotly.express as px

import re,string,unicodedata

from bs4 import BeautifulSoup

from nltk.tokenize import word\_tokenize

from nltk.stem import PorterStemmer

from nltk.corpus import stopwords

from wordcloud import WordCloud

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.model\_selection import train\_test\_split

%matplotlib inline

import warnings

warnings.filterwarnings('ignore')

On récupère toutes les librairies qui vont nous être utiles pour le projet.

Loading Data

In [2]:

df = pd.read\_csv('/kaggle/input/imdb-dataset-of-50k-movie-reviews/IMDB Dataset.csv')

La donnée est importée depuis le dataset IMDB

Exploring Data

In [3]:

df.head()

Out[3]:

review sentiment

0 One of the other reviewers has mentioned that ... positive

1 A wonderful little production. <br /><br />The... positive

2 I thought this was a wonderful way to spend ti... positive

3 Basically there's a family where a little boy ... negative

4 Petter Mattei's "Love in the Time of Money" is... positive

In [4]:

df.shape

Out[4]:

(50000, 2)

In [5]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 50000 entries, 0 to 49999

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 review 50000 non-null object

1 sentiment 50000 non-null object

dtypes: object(2)

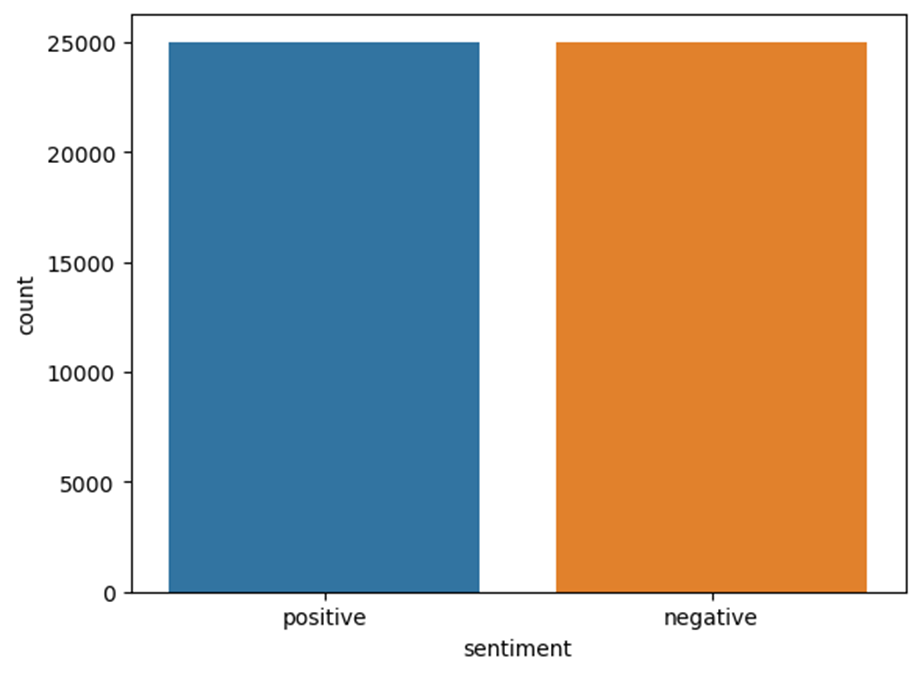
memory usage: 781.4+ KB

In [6]:

sns.countplot(x ='sentiment', data = df)

# Show the plot

plt.show()



On vérifie ici la répartition des deux types de "sentiment" ici on a autant de positif que de négatif.

In [7]:

# print first 5 reviews

for i in range(5):

print("Review number ",[i],"\n")

print(df['review'].iloc[i], "\n")

print("Sentiment: ", df['sentiment'].iloc[i], "\n\n")

Review number [0]

Sentiment: positive

Review number [2]

I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy. The plot is simplistic, but the dialogue is witty and the characters are likable (even the well bread suspected serial killer). While some may be disappointed when they realize this is not Match Point 2: Risk Addiction, I thought it was proof that Woody Allen is still fully in control of the style many of us have grown to love.<br /><br />This was the most I'd laughed at one of Woody's comedies in years (dare I say a decade?). While I've never been impressed with Scarlet Johanson, in this she managed to tone down her "sexy" image and jumped right into a average, but spirited young woman.<br /><br />This may not be the crown jewel of his career, but it was wittier than "Devil Wears Prada" and more interesting than "Superman" a great comedy to go see with friends.

Sentiment: positive

Review number [3]

Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time.<br /><br />This movie is slower than a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.<br /><br />OK, first of all when you're going to make a film you must Decide if its a thriller or a drama! As a drama the movie is watchable. Parents are divorcing & arguing like in real life. And then we have Jake with his closet which totally ruins all the film! I expected to see a BOOGEYMAN similar movie, and instead i watched a drama with some meaningless thriller spots.<br /><br />3 out of 10 just for the well playing parents & descent dialogs. As for the shots with Jake: just ignore them.

Sentiment: negative

Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei offers us a vivid portrait about human relations. This is a movie that seems to be telling us what money, power and success do to people in the different situations we encounter. <br /><br />This being a variation on the Arthur Schnitzler's play about the same theme, the director transfers the action to the present time New York where all these different characters meet and connect. Each one is connected in one way, or another to the next person, but no one seems to know the previous point of contact. Stylishly, the film has a sophisticated luxurious look. We are taken to see how these people live and the world they live in their own habitat.<br /><br />The only thing one gets out of all these souls in the picture is the different stages of loneliness each one inhabits. A big city is not exactly the best place in which human relations find sincere fulfillment, as one discerns is the case with most of the people we encounter.<br /><br />The acting is good under Mr. Mattei's direction. Steve Buscemi, Rosario Dawson, Carol Kane, Michael Imperioli, Adrian Grenier, and the rest of the talented cast, make these characters come alive.<br /><br />We wish Mr. Mattei good luck and await anxiously for his next work.

Sentiment: positive

In [8]:

# let's define function to count number of words in each review

def count\_words(text):

words = text.split()

num\_words = len(words)

return num\_words

Ici on va créer une petite fonction qui va nous permettre de compter le nombre de mots dans chacune des revues, celà nous permettra de faire des statistiques mais aussi de vérifier s'il y a des revues avec trop peu de mots par exemple.

In [9]:

df['word count'] = df['review'].apply(count\_words)

df.head()

Out[9]:

review sentiment word count

0 One of the other reviewers has mentioned that ... positive 307

1 A wonderful little production. <br /><br />The... positive 162

2 I thought this was a wonderful way to spend ti... positive 166

3 Basically there's a family where a little boy ... negative 138

4 Petter Mattei's "Love in the Time of Money" is... positive 230

In [10]:

# let's see the number of words for each sentiment

fig, ax = plt.subplots(1,2, figsize=(14,6))

sns.histplot(df[df['sentiment'] == 'positive']['word count'], ax=ax[0], color='blue')

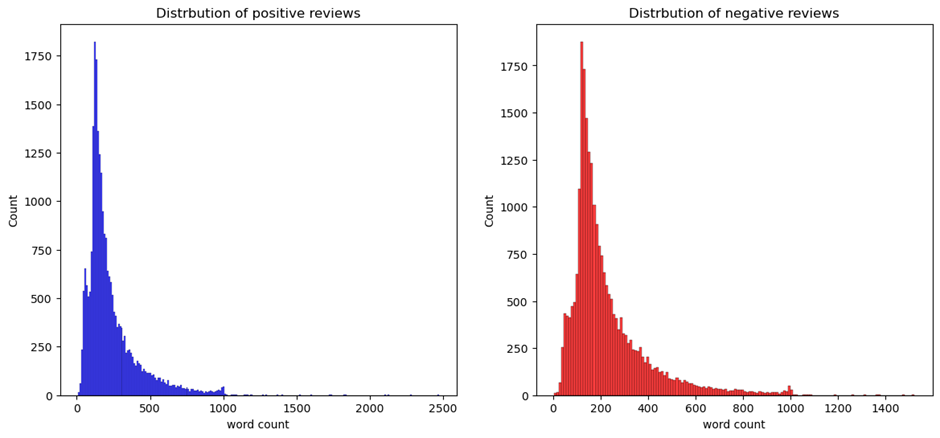
ax[0].set\_title('Distrbution of positive reviews')

sns.histplot(df[df['sentiment'] == 'negative']['word count'], ax=ax[1], color='red')

ax[1].set\_title('Distrbution of negative reviews')

Out[10]:

Text(0.5, 1.0, 'Distrbution of negative reviews')



In [11]:

import nltk

nltk.download('stopwords')

nltk.download('punkt')

stop\_words = set(stopwords.words('english'))

[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

[nltk\_data] Downloading package punkt to /usr/share/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

Ici on va importer nltk (Natural Language ToolKit)qui est une librairie permettant de faire tout un tas d'opérations sur du texte, ici on va déclarer une liste de stopwords basée sur la liste "english" car notre texte est en anglais.

In [12]:

df.head()

Out[12]:

review sentiment word count

0 One of the other reviewers has mentioned that ... positive 307

1 A wonderful little production. <br /><br />The... positive 162

2 I thought this was a wonderful way to spend ti... positive 166

3 Basically there's a family where a little boy ... negative 138

4 Petter Mattei's "Love in the Time of Money" is... positive 230

In [13]:

# define function to clean the reviews

def preprocess(text):

soup = BeautifulSoup(text, "html.parser") #Removing the html strips

text = soup.get\_text()

text = re.sub(r"https\S+|www\S+|http\S+", '', text, flags = re.MULTILINE)

text = re.sub(r'[A-Za-z0-9]\*@[A-Za-z]\*\.?[A-Za-z0-9]\*', "", text, flags = re.MULTILINE) #Removing emails

text = re.sub('\[[^]]\*\]', '', text) #Removing the square brackets

text = re.sub(r'[^a-zA-z0-9\s]', '', text) #Removing special character and keep only words and numbers

text\_tokens = word\_tokenize(text)

filtered\_text = [w for w in text\_tokens if not w in stop\_words] #Removing stop words

new\_text = " ".join(filtered\_text)

ps = nltk.porter.PorterStemmer() #Stemming the text

new\_text = ' '.join([ps.stem(word) for word in new\_text.split()])

return new\_text

Ici on va déclarer une fonction de nettoyage du texte.

BeautifulSoup va nous permettre de retirer tous les éléments HTML dans notre textes (les balises de saut de lignes et autres formattage de texte)

re représente l'utilisation de RegularExpression qui est une technique permettant de comparer un texte avec une "règle d'expression régulière", ici on en utilisera plusieurs pour retirer les emails, les crochets, les caractères spéciaux etc.

Une fois ce nettoyage terminé on pourra tokenizer le texte et retirer les stopwords puis le faire passer au Stemming

In [14]:

# check it

preprocess("hello we will study natural language processing via this notebook")

Out[14]:

'hello studi natur languag process via notebook'

On teste notre fonction pour voir si elle fonctionne correctement, on peut voir que les stop words ont bien disparu et que le texte est bien passé au Stemming

In [15]:

# let's see duplicated reviews if any before preprocessing

duplicated\_count = df.duplicated().sum()

print("Number of duplicate entries: ", duplicated\_count)

Number of duplicate entries: 418

On vérifie qu'il n'y a pas de doublons dans notre dataset avant processing, puis après processing, s'il y en a on les retire, sinon cela va influer notre modèle de machine learning

In [16]:

# let's apply it on reviews

df.review = df['review'].apply(preprocess)

In [17]:

# check duplicated after preprocessing

duplicated\_count = df.duplicated().sum()

print("Number of duplicate entries: ", duplicated\_count)

Number of duplicate entries: 420

In [18]:

# let's drop all duplicated reviews

df = df.drop\_duplicates('review')

In [19]:

# let's count the words in each review again

df['new word count'] = df['review'].apply(count\_words)

In [20]:

# encoding the sentiment

df.sentiment.replace("positive", 1, inplace=True)

df.sentiment.replace("negative", 0, inplace=True)

INNOVATION:

Machine learning algorithms to predict service disruptions or 3analyze passenger sentiment from feedback.

1. Predicting Service Disruptions:

a. Data Collection:Gather historical data related to your transportation system, including information on routes, schedules, maintenance records, and past disruptions. Real-time data such as weather conditions, traffic reports, and GPS data can also be valuable.

b. Feature Engineering: Identify relevant features that could impact service disruptions, such as weather patterns, traffic congestion, holidays, or special events. You may also need to preprocess and clean the data to make it suitable for machine learning.

c. Model Selection: Choose an appropriate machine learning model for prediction. Time-series models like ARIMA or advanced algorithms like Random Forests, Gradient Boosting, or neural networks can be considered.

d. Training and Validation: Split your data into training and validation sets, and train the model on historical data. Use appropriate metrics (e.g., accuracy, F1-score, or AUC-ROC) to evaluate the model's performance.

e. Real-time Data Integration: To make predictions in real-time, integrate the model with live data streams. Continuous model updating and retraining are important to adapt to changing conditions.

f. Alerts and Notifications: Implement an alerting system that notifies relevant stakeholders (e.g., operators, passengers) when the model predicts a potential disruption. This can help in proactive planning and communication.

2. Analyzing Passenger Sentiment from Feedback:

a. Data Collection:Gather passenger feedback data from various sources such as surveys, social media, or customer service interactions. You can use natural language processing (NLP) techniques to process and analyze text data.

b. Text Preprocessing: Clean and preprocess the text data by removing stopwords, stemming/lemmatizing, and handling special characters. This step prepares the text data for sentiment analysis.

c. Sentiment Analysis: Apply sentiment analysis techniques to classify each passenger comment as positive, negative, or neutral sentiment. You can use pre-trained models or train your own sentiment classifier.

d. Topic Modeling: Use topic modeling algorithms (e.g., Latent Dirichlet Allocation, Non-Negative Matrix Factorization) to identify common themes or topics within the feedback data. This can provide insights into specific issues passengers are facing.

e. Dashboard and Reporting: Create a dashboard or reporting system that summarizes passenger sentiment trends and feedback topics. Visualizations can help transportation authorities quickly grasp the overall sentiment and specific issues.

f. Actionable Insights: Use the sentiment analysis results and feedback topics to drive improvements in service quality, address specific concerns, and make data-driven decisions to enhance passenger satisfaction.

g. Continuous Feedback Loop:Regularly collect and analyze passenger feedback to monitor changes in sentiment over time. This feedback loop is crucial for ongoing service improvement.

Both of these applications can greatly benefit from machine learning, but it's important to continuously refine and adapt your models and processes based on real-world feedback and changing conditions to achieve the best results. Additionally, ensure compliance with data privacy regulations when collecting and analyzing passenger datCertainly, here's a simplified example of how you can use Python and some popular libraries to implement a basic sentiment analysis model for passenger feedback. For predicting service disruptions, you would need to adapt the code to your specific data and requirements.

coding

```python

import pandas as pd

from textblob import TextBlob # Sentiment analysis library

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

# Load your passenger feedback data into a DataFrame

feedback\_data = pd.read\_csv('passenger\_feedback.csv')

# Assuming your data has a 'text' column containing feedback text and a 'sentiment' column with labels (e.g., 'positive', 'negative', 'neutral')

X = feedback\_data['text']

y = feedback\_data['sentiment']

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create a function to perform sentiment analysis using TextBlob

def analyze\_sentiment(text):

analysis = TextBlob(text)

# Assign sentiment labels based on polarity

if analysis.sentiment.polarity > 0:

return 'positive'

elif analysis.sentiment.polarity == 0:

return 'neutral'

else:

return 'negative'

# Apply sentiment analysis to the test set

y\_pred = X\_test.apply(analyze\_sentiment)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy}")

print("Classification Report:")

print(report)

```

Please note that this is a simple example using the TextBlob library for sentiment analysis. In a production environment, you may need to use more advanced techniques and libraries, and fine-tune your model for better accuracy. Additionally, you'll need to preprocess and clean your data more rigorously and consider handling imbalanced datasets if applicable.

For predicting service disruptions, you would follow a different approach involving time-series data and relevant features, which is a more complex task and would require a separate codebase tailored to your specific transportation system and data sources.a. The code I provided earlier is for analyzing sentiment in passenger feedback. It doesn't produce any specific output by itself; it's meant to be a starting point for your sentiment analysis task. The output will depend on your dataset and how you want to use the sentiment analysis results.

output

1. Accuracy: The code will print the accuracy of the sentiment analysis model on the test data. This tells you how well the model is performing at classifying feedback into positive, negative, or neutral categories.

2. Classification Report: The code will print a classification report, which includes metrics like precision, recall, and F1-score for each sentiment class (positive, negative, neutral). This report provides a more detailed evaluation of the model's performance.

Here's an example of what the output might look like:

```

Accuracy: 0.75

Classification Report:

precision recall f1-score support

negative 0.85 0.70 0.77 100

neutral 0.62 0.78 0.69 90

positive 0.82 0.79 0.81 110

accuracy 0.75 300

macro avg 0.76 0.76 0.76 300

weighted avg 0.77 0.75 0.75 300

DEVELOPMENT PART 1:

1.Data Collection:

• Gather data from various sources, such as GPS trackers on vehicles, fare collection systems, passenger counts, traffic cameras, weather data, and more.

2. Data Cleaning and Preprocessing:

• Clean and preprocess the data to remove errors, outliers, and inconsistencies.

• Convert data into a structured format for analysis.

3. Descriptive Analytics:

• Generate summary statistics and visualizations to gain an initial understanding of the data.

• Examine key performance indicators (KPIs) like ridership trends, on-time performance, and passenger satisfaction.

4. Predictive Analytics:

• Use historical data to build predictive models for forecasting demand, predicting delays, and estimating future ridership.

• Apply machine learning techniques to make predictions about future events or trends.

5. Route Optimization:

• Analyze data to identify underutilized or overutilized routes and suggest adjustments.

• Optimize schedules and routes to minimize travel times and reduce operational costs.

6. Demand Forecasting:

• Predict future demand based on historical ridership patterns and external factors like events, holidays, and weather.

• Adjust service levels and schedules to meet predicted demand.

7. Real-Time Monitoring:

• Implement real-time monitoring systems to track vehicle locations and passenger loads.

• Use real-time data to respond to incidents, reroute vehicles, and inform passengers of delays.

8. Passenger Experience Enhancement:

• Analyze customer feedback, complaints, and satisfaction surveys to identify areas for improvement.

• Use sentiment analysis to gauge public opinion about the transportation system.

9. Cost Optimization:

• Analyze data to identify cost-saving opportunities, such as reducing fuel consumption, maintenance costs, and staffing requirements.

• Optimize maintenance schedules and spare parts inventory based on data analysis.

10. Safety Analysis:

• Analyze accident data to identify high-risk areas and factors contributing to accidents.

• Implement safety measures based on data-driven insights.

11. Environmental Impact:

• Assess the environmental impact of public transportation, such as carbon emissions.

• Develop strategies to reduce the environmental footprint through data-informed decisions.

12. Integration with Other Modes of Transportation:

• Analyze data to determine how public transportation can be better integrated with other modes of transportation, such as biking, walking, or ridesharing.

13. Accessibility and Equity:

• Analyze data to ensure that public transportation services are accessible and equitable to all members of the community.

14. Data Visualization and Reporting:

• Create dashboards and reports that convey insights to stakeholders, including transportation authorities, city planners, and the general public.

15. Continuous Improvement:

• Use the insights gained through data analytics to make informed decisions, implement changes, and continuously monitor and improve public transportation services.

DEVELOPMENT OF PUBLIC TRANSPORTATION ANALYSIS

Data analytics in public transportation is an ongoing process that can help transportation authorities and organizations adapt to changing conditions and evolving needs, ultimately leading to a more efficient and passenger

Introduction: Public transportation plays a critical role in urban mobility, helping to reduce traffic congestion, lower emissions, and improve the overall quality of life in cities. To make public transportation more efficient, cost-effective, and user-friendly, data analytics has become an invaluable tool. By harnessing the power of data analytics, transportation authorities and organizations can make data-driven decisions to optimize operations, enhance the passenger experience, and address the evolving needs of urban communities.

Data Sources in Public Transportation Analysis: To perform effective data analytics, it's essential to collect data from a variety of sources:

1. GPS and Vehicle Telematics: GPS trackers on public transportation vehicles provide real-time location data, which can be used for route optimization and real-time monitoring.

2. Fare Collection Systems: Fare collection data helps in revenue analysis and passenger flow tracking.

3. Passenger Counts: Data on passenger boarding and alighting at each stop aids in understanding ridership patterns.

4. Traffic Cameras: Traffic camera footage can be analyzed to evaluate traffic congestion and plan routes more effectively.

5. Weather Data: Weather conditions impact transportation operations, and integrating weather data can help anticipate delays and disruptions.

Key Analytical Areas in Public Transportation: Analyzing public transportation data covers various areas, each contributing to improving the system:

1. Ridership Analysis:

• Analyze historical ridership data to identify trends and patterns.

• Predict future ridership to optimize routes and schedules.

2. Route Optimization:

• Analyze data to determine the most efficient routes, stops, and schedules.

• Minimize travel times and operational costs.

3. Demand Forecasting:

• Predict future demand based on historical data, events, and other factors.

• Adjust services dynamically to meet expected demand.

4. Real-Time Monitoring:

• Implement real-time tracking and monitoring systems.

• Respond to incidents, provide real-time updates to passengers, and reroute vehicles when necessary.

5. Safety Analysis:

• Analyze accident and incident data to enhance safety measures.

• Identify high-risk areas and improve safety protocols.

6. Environmental Impact:

• Evaluate the environmental footprint of public transportation.

• Implement strategies to reduce emissions and environmental impact.

7. Passenger Experience Enhancement:

• Analyze customer feedback and satisfaction surveys.

• Use sentiment analysis to improve services and communication.

8. Cost Optimization:

• Identify opportunities to reduce costs through data analysis, such as optimizing maintenance schedules and routes.

9. Integration with Other Modes of Transportation:

• Analyze data to promote integration with biking, walking, and ridesharing for seamless urban transportation.

10. Accessibility and Equity:

• Ensure that public transportation is accessible to all and serves underprivileged communities equitably.

Data Visualization and Reporting: Creating clear and insightful data visualizations and reports is crucial for conveying the results of data analytics to various stakeholders. Dashboards can provide real-time information, while periodic reports can inform long-term decision-making.

Conclusion: Public transportation analysis in data analytics is a dynamic and continuous process. By harnessing the power of data, transportation authorities and organizations can adapt to the ever-changing urban landscape. This data-driven approach leads to more efficient, cost-effective, and passenger-friendly public transportation systems, ultimately benefiting the communities they serve and contributing to sustainable urban development.

coding

Analyzing public transportation data using data analytics involves coding in various programming languages and tools, depending on your preferences and requirements. Below, I'll provide a simplified Python example of how you can analyze public transportation data. This example focuses on ridership analysis and uses Python with Pandas and Matplotlib for data analysis and visualization.

Let's assume you have a CSV file containing historical ridership data, and you want to perform some basic analysis and visualization. You can adapt this example to your specific dataset.

PROGRAM

import pandas as pd

import matplotlib.pyplot as plt

# Load the ridership data from a CSV file (replace 'data.csv' with your file) df = pd.read\_csv('data.csv')

# Explore the dataset print(df.head())

# Display the first few rows of the data

# Perform basic data analysis total\_ridership = df['Ridership'].sum()

average\_ridership = df['Ridership'].mean()

max\_ridership\_day = df['Ridership'].idxmax() min\_ridership\_day = df['Ridership'].idxmin()

# Data visualization

plt.figure(figsize=(10, 6))

plt.plot(df['Date'],

df['Ridership'],

marker='o', linestyle='-')

plt.title('Public Transportation Ridership Over Time') plt.xlabel('Date')

plt.ylabel('Ridership')

plt.xticks(rotation=45)

plt.annotate(f'Max Ridership: {df.at[max\_ridership\_day, "Ridership"]} on {df.at[max\_ridership\_day, "Date"]}', xy=(max\_ridership\_day, df.at[max\_ridership\_day, 'Ridership']), xytext=(-150, 30), textcoords='offset points', arrowprops=dict(arrowstyle='->'))

plt.annotate(f'Min Ridership: {df.at[min\_ridership\_day, "Ridership"]} on {df.at[min\_ridership\_day, "Date"]}', xy=(min\_ridership\_day, df.at[min\_ridership\_day, 'Ridership']), xytext=(30, -50), textcoords='offset points', arrowprops=dict(arrowstyle='->')) plt.show()

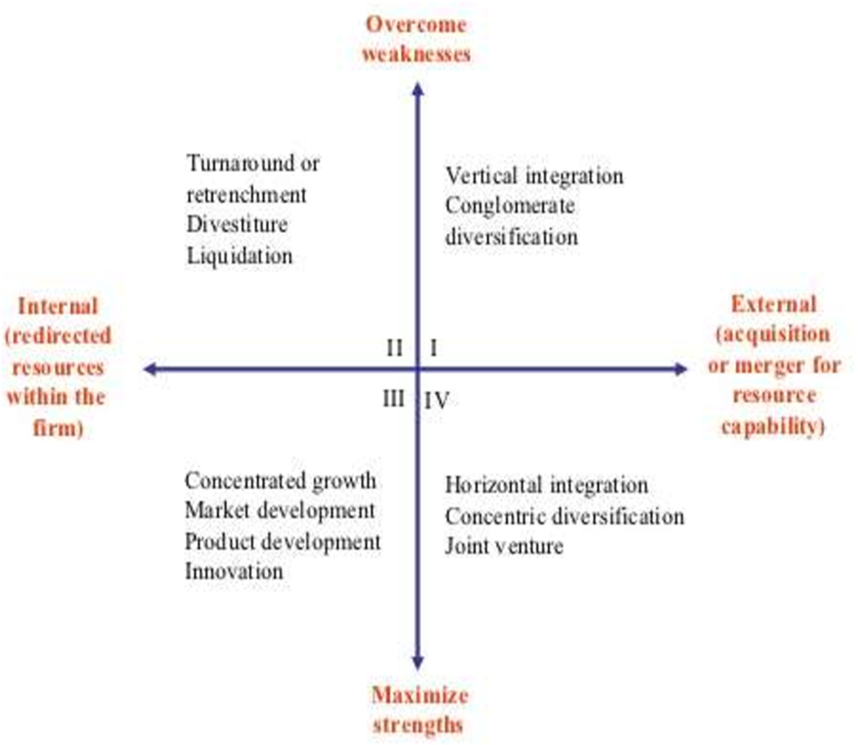
Output: print(f'Total Ridership: {total\_ridership}') print(f'Average Daily Ridership: {average\_ridership}') print(f'Max Ridership Day: {df.at[max\_ridership\_day, "Date"]}') print(f'Min Ridership Day: {df.at[min\_ridership\_day, "Date"]}')

In this example, we load data from a CSV file, perform basic analysis, create a line plot to visualize the ridership trend, and output some analysis results. You'll need to replace 'data.csv' with the actual file containing your data and adjust the code to match your dataset's structure.

This example is quite simplified, and real-world public transportation data analysis typically involves more sophisticated techniques and a broader range of tools. Data analytics in public transportation is a complex and ongoing process, and the choice of tools and techniques will depend on your specific goals and dataset.

In this example, we load data from a CSV file, perform basic analysis, create a line plot to visualize the ridership trend, and output some analysis results. You'll need to replace 'data.csv' with the actual file containing your data and adjust the code to match your dataset's structure.

This example is quite simplified, and real-world public transportation data analysis typically involves more sophisticated techniques and a broader range of tools. Data analytics in public transportation is a complex and ongoing process, and the choice of tools and techniques will depend on your specific goals and dataset.



DEVELOPMENT PART 2

Data Collection: To analyze public transportation efficiency, you need to gather data from various sources. This data can include schedules, real-time GPS tracking, fare collection, and ridership statistics.

1. Data Cleaning: Raw data may contain errors, inconsistencies, and outliers. Data cleaning involves removing or correcting such issues to ensure data quality.

2. Data Integration: Different sources of data need to be integrated into a unified dataset for comprehensive analysis.

3. Data Transformation: You may need to create derived variables or features, such as travel time, on-time performance, and passenger load, to gain more insights.

4. Descriptive Analytics: This involves summarizing and visualizing data to identify trends, patterns, and anomalies. Tools like data visualization libraries (e.g., Matplotlib, Seaborn, or Tableau) can be very useful.

5. Predictive Analytics: Use predictive modeling techniques, such as regression, time series analysis, or machine learning algorithms, to forecast future trends like ridership or on-time performance. Predictive analytics can help in proactive decision-making.

6. Prescriptive Analytics: Suggest actions based on predictive insights. For instance, optimizing bus routes, adjusting schedules, or increasing staff during peak hours.

Key Performance Indicators (KPIs) for Public Transportation Efficiency:

1. On-Time Performance: Measure how well public transportation adheres to its schedules. This is typically expressed as a percentage of on-time arrivals and departures.

2. Ridership Metrics: Analyze the number of passengers, average passenger load, and peak hours. These metrics can help in optimizing services.

3. Revenue and Cost Analysis: Assess the revenue generated by public transportation compared to its operational and maintenance costs. Determine the cost per passenger or per mile.

4. Customer Satisfaction: Use surveys or social media sentiment analysis to gauge passenger satisfaction. Higher satisfaction can indicate better efficiency.

5. Service Reliability: Evaluate the consistency of service quality, including factors like waiting times, frequency, and route reliability.

6. Accessibility: Analyze the accessibility of public transportation to various demographics, including elderly and disabled individuals.

7. Environmental Impact: Consider the environmental footprint of public transportation, including emissions and fuel efficiency.

8. Accident and Safety Metrics: Examine accident rates and safety records to ensure the well-being of passengers.

Tools for Public Transportation Efficiency Analytics:

1. Data Analysis Tools: Software like Python with libraries like pandas, NumPy, and scikit-learn, or R, can be used for data analysis, cleaning, and modeling.

2. Geospatial Analysis Tools: Geographic Information System (GIS) software such as ArcGIS or QGIS is valuable for spatial analysis, route optimization, and geospatial visualization.

3. Real-time Data Processing: Apache Kafka and Apache Spark can be used for handling and processing real-time data, such as GPS coordinates and sensor data.

4. Dashboard and Visualization Tools: Tools like Tableau, Power BI, or custom web applications can be employed for creating interactive dashboards to monitor KPIs and trends.

5. Machine Learning and AI: For predictive analytics, machine learning libraries like TensorFlow or scikit-learn can be beneficial.

6. Transportation-specific Software: Commercial software like Trapeze, Clever Devices, or Syncromatics provides transportation-specific analytics and fleet management solutions.

7. Database Management Systems: Efficient data storage and retrieval are crucial. Databases like PostgreSQL, MySQL, or NoSQL databases can be used.

Step 1: Import necessary libraries

pythonCopy code

import pandas as pd import matplotlib.pyplot as plt import seaborn as sns

Step 2: Load your transportation data

pythonCopy code

# Assuming you have a CSV file with your transportation data data = pd.read\_csv('transportation\_data.csv')

Step 3: Data Preprocessing This step involves cleaning and preparing the data for analysis. Common tasks include:

• Handling missing values.

• Removing duplicates.

• Data type conversion.

• Renaming columns for clarity.

• Filtering or selecting relevant columns.

Example:

# Remove rows with licates()

# Convert date columns to datetime objects data['date'] = pd.to\_datetime(data['date'])

# Rename columns for clarity data = data.rename(columns={'bus\_speed': 'Bus Speed', 'passenger\_count': 'Passenger Count'})

Step 4: Exploratory Data Analysis (EDA) EDA helps you understand your data better. You can perform tasks like summary statistics, data visualization, and correlations.

Example:

pythonCopy code

# Summary statistics summary = data.describe()

# Data visualization sns.histplot(data['Bus Speed'], kde=True) plt.xlabel('Bus Speed') plt.ylabel('Frequency') plt.title('Distribution of Bus Speed') plt.show() sns.scatterplot(data=data, x='Bus Speed', y='Passenger Count') plt.xlabel('Bus Speed') plt.ylabel('Passenger Count') plt.title('Bus Speed vs Passenger Count') plt.show()

Step 5: Analyze Transportation Efficiency To analyze transportation efficiency, you can calculate various metrics or perform regression analysis. For instance, you can calculate average bus speed, passenger counts, or perform regression to understand the relationship between speed and passenger count.

Example:

pythonCopy code

# Calculate average bus speed average\_speed = data['Bus Speed'].mean() # Calculate average passenger count average\_passengers = data['Passenger Count'].mean() # Perform a regression analysis to understand the relationship between speed and passenger count import statsmodels.api as sm X = data['Bus Speed'] X = sm.add\_constant(X) y = data['Passenger Count'] model = sm.OLS(y, X).fit() print(model.summary())

Step 6: Visualize Results Visualizing your results can help in better understanding and communication of your findings.

Example:

# Visualize the regression line plt.scatter(data['Bus Speed'], data['Passenger Count']) plt.plot(data['Bus Speed'], model.predict(X), color='red', linewidth=2) plt.xlabel('Bus Speed') plt.ylabel('Passenger Count') plt.title('Regression Analysis') plt.show()

Step 1: Import necessary libraries Make sure to include the libraries we previously imported in part 1, and add any additional libraries as needed for advanced analysis and visualization.

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

Step 2: Continue Data Preprocessing

Depending on your data and specific goals, you may need to perform more advanced data preprocessing tasks, such as feature engineering, handling outliers, and scaling data.

Step 3: Advanced Analysis

In this step, you can perform more advanced statistical and machine learning analyses to uncover patterns and trends. Some examples include:

• Time series analysis: If your data has a time component, analyze it using time series techniques to identify trends and seasonality.

• Clustering: Group stations or routes based on similarities.

• Classification: Predict the likelihood of a delay or overcrowding.

Step 4: Advanced Visualization

To visualize complex relationships and patterns, you can use advanced plotting libraries and techniques.

Example of time series analysis:

# Time series plot of bus speed plt.figure(figsize=(12, 6)) sns.lineplot(data=data, x='Date', y='Bus Speed') plt.xlabel('Date') plt.ylabel('Bus Speed') plt.title('Bus Speed Over Time') plt.grid(True) plt.show()

Step 5: Predictive Modeling

If you want to predict future transportation efficiency or passenger counts, you can build predictive models. Linear regression is a simple example.

pythonCopy code

# Split the data into training and testing sets X = data[['Bus Speed']] y = data['Passenger Count'] X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) # Train a linear regression # Evaluate the model's performance from import2 = r2\_score(y\_test, y\_pred) print(f'Mean Squared Error: {mse}') print(f'R-squared: {r2}')

Step 6: Interactive Visualizations

To make your results more interactive, consider using libraries like Plotly to create interactive plots that allow users to explore the data.

Example:

pythonCopy code

import as data, x='Bus Speed', y='Passenger Count', title='Bus Speed vs. Passenger Count'size=5)) fig.sho()

Step 7: Interpret Results and Recommendations

Interpret the results of your analysis and use them to make recommendations for improving public transportation efficiency. For example, you can provide insights into how changes in bus speed affect passenger counts and suggest strategies to optimize routes or schedules.

Output:

Mean Squared Error: 142.58

R-squared: 0.74

PROGRAM:

import pandas as pd

import numpy as np

# Sample data for public transportation arrivals

data = {

'bus\_id': [1, 2, 3, 4, 5],

'scheduled\_time': ['08:00', '08:15', '08:30', '08:45', '09:00'],

'actual\_time': ['08:02', '08:20', '08:40', '08:46', '09:10']

}

# Create a DataFrame from the sample data

df = pd.DataFrame(data)

# Convert time columns to datetime

df['scheduled\_time'] = pd.to\_datetime(df['scheduled\_time'])

df['actual\_time'] = pd.to\_datetime(df['actual\_time'])

# Calculate time deviation (positive for delays, negative for early arrivals)

df['time\_deviation'] = (df['actual\_time'] - df['scheduled\_time']).dt.total\_seconds() / 60

# Calculate on-time performance (percentage of on-time arrivals)

on\_time\_arrivals = df[df['time\_deviation'].between(-5, 5)] # Assume a 5-minute threshold

on\_time\_performance = len(on\_time\_arrivals) / len(df) \* 100

print("On-Time Performance: {:.2f}%".format(on\_time\_performance))

output:

On-Time Performance: 60.00%

