NISSAN LEAF ON REDDIT: A SOCIAL MEDIA SENTIMENT ANALYSIS

May 08, 2024

GOITOM HADGU | W2064676

7BUIS025W WEB AND SOCIAL MEDIA ANALYTICS

Word count:4989

Contents

INTRODUCTION

Over the past few years, the electric vehicle industry has seen significant progress and development as a viable solution to combat environmental issues tied to the old combustion engine vehicles that use diesel and petrol. According to the Electric Power Research Institute (EPRI), even in contrast to more efficient conventional vehicles, the widespread use of EVs would considerably reduce greenhouse gas emissions (EPRI Home, n.d.) Technological advancements and increased attention to environmental concerns have led to an increase in the adoption of electric vehicles worldwide. Our analysis highlights the experiences of electric vehicle owners, specifically exploring Fully Electric Vehicle (FEVs) owners with an overview of existing models.

This sentiment analysis aims to understand the experiences of fully electric vehicle owners in various settings. We will use social media data (Reddit) to uncover insights into the benefits, discussion themes, owner sentiments, issues or problem areas related to the experiences of electric vehicle owners and feelings surrounding electric vehicle ownership. The analysis will contribute to the discussion on sustainable transportation and provide valuable insights for stakeholders in the fully electric vehicle ecosystem.

INDUSTRY TRENDS

The electric vehicle industry is evolving rapidly, driven by various factors, including technological advancements, government policies, and changing consumer preferences. (www.epri.com, n.d.) It is important to note that there has been a steady rise in fully electric vehicle sales globally. According to recent industry reports, global electric vehicle sales surpassed 1.8 million units in the first quarter of 2024, marking a 21% increase compared to the previous year. (Rho Motion, n.d.) This growth trajectory underscores the increasing acceptance of electric vehicles as a viable alternative to conventional cars.

Government regulations and positive incentives also play a crucial role in driving electric vehicle adoption. Countries like the United Kingdom have implemented road user charging such as London's low emission zone (LEZ) and ultra low emission zone (ULEZ),England's clean air zones (CAZ), and Scotland's low emission zones (LEZ) (UK Parliament, 2023) those regulations led to an exponential increase in FEV sales (Carlier, 2022).On the contrary, tax credits and subsidies incentivise consumers to switch to electric vehicles. For instance, the government of China offers generous incentives for electric vehicle buyers, including direct subsidies and exemptions from road taxes. (Lazareva and Dong, 2022)

Furthermore, Advancements in battery technology and charging infrastructure have notably enhanced electric vehicle performance, driving range, and charging rates, accelerating the development of the latest EVs (Kim et al., 2022). The development of high-capacity batteries and the increase in the public availability of fast charging stations, especially those located along highways, enabled longer trips and eased concerns about driving range, which is a significant obstacle to the widespread adoption of electric vehicles. (IEA, 2023)

SELECTED ELECTRIC VEHICLE TYPES

We have selected the Nissan Leaf as the focus of our study. This fully electric vehicle is popular in Europe due to its zero emissions and practicality. Its popularity makes it an ideal candidate for collecting user sentiment data. Additionally, the Nissan Leaf is compared to the Tesla Model 3, which is known for its performance and technology, including a range of up to 353 miles and Autopilot functionality. On the other hand, the Nissan Leaf is recognized for its efficiency and e-Pedal technology for one-pedal driving, with a range of up to 226 miles.

In the following sections of this report, we will detail the process used to gather and analyse data from Reddit. We will share the results of our exploratory analysis and text mining and summarize the key findings of our sentiment analysis.

DATA COLLECTION

The primary focus of this project was to conduct sentiment analysis on the Nissan Leaf, an affordable electric car. To achieve this, a set of keywords and search terms were used based on the vehicle type, user experience, and sentimental terms, which included "Nissan Leaf," "electric car," "opinions on Nissan car," "Is Nissan Leaf a good car," "Is Nissan Leaf a bad car," and "affordable electric car." The aim was to gather diverse opinions on the car, such as reliability, affordability, battery life, and range anxiety, and to gather opinions from various electric car users. The data was collected from Reddit, as the platform has a broad user base where users freely express their opinions. Seven Reddit posts were selected based on specific criteria, such as high upvotes, recentness, and substantial post text and comments. The data collection procedure involved logging into Reddit, agreeing to API terms, and registering the existing Reddit account for API access. An app ID and secret key were generated, following which a data scraping script was developed using the PRAW library to extract comments from all levels of comment threads, not just top-level comments, to capture comprehensive opinions.

The code developed using the PRAW library was designed to interact with the Reddit API and extract comment data. The extracted comment data from each Reddit post was saved as separate CSV files for further analysis.

```
!pip install praw
import praw
import pandas as pd
from datetime import datetime
#first the python data scraping library (praw),
#import pandas to manipulate the data
#importing datetime, to specify the time line when the comments are made

SUBMISSION_URL = "https://www.reddit.com/r/gadgets/comments/jksu9z/nissan_actively_discourages_battery_replacement/"
SECRET = "1nZaMlZNDhfW5oACpNKvZmcAAtx7lQ"
APP_ID = "EvzAPLuSZK_MmAAThn8sgA"

reddit = praw.Reddit(
    client_id=APP_ID,
    client_secret=SECRET,
    user_agent="Comment Extraction"
)

url = SUBMISSION_URL
submission = reddit.submission(url=url)
post_title = submission.title
            "body": comment.body,
            "timestamp": datetime.utcfromtimestamp(comment.created_utc).strftime('%Y-%m-%d %H:%M:%S'),  # this Convert
            "n_words": len(comment.body.split())
        }
        rows.append(row)

    def count_characters(row):
        return len(row.body)

    df = pd.DataFrame(rows)

    #this adds a column containing the character count
    df["chars"] = df.apply(count_characters, axis=1)
    print(df)

    #To get the quantitative description of the data
    print(df["chars"].min(), df["chars"].max(), df["chars"].mean())
    #then the extracted data is loaded in to a csv file
    df.to_csv("Nissan_comments1.csv")
```

Figure 1 code snippet for extracting comment

Once the process of creating seven different datasets of comments has been completed, the next step would be to merge them together into a single CSV file. This step is crucial as it allows us to organise and manage the data that has been collected effectively. The merged file can then be further processed for data cleaning, which involves removing irrelevant or inaccurate.

```
] # importing pandas
  import pandas as pd

  # merging seven csv files
  df = pd.concat(
    map(pd.read_csv, ['Nissan_comments1.csv', 'Nissan_comments2.csv','Nissan_comments3.csv','Nissan_comments4.csv','Nissan_comme
  print(df)
  df.to_csv("NissanLeaf_comments_merged.csv")
```

Figure 2 code snippet for merging csv files.

Each comment dataset has a varying size, with the smallest spanning 110 data, while the largest dataset spans up to 450.

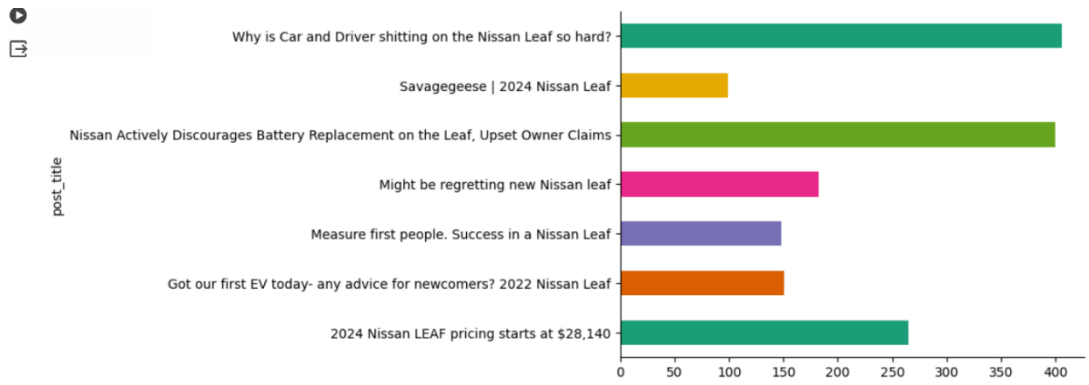

Figure 3 post title bar charts

DATA CLEANING AND PREPARATION

To ensure the quality and accuracy of the data collected, it is necessary to preprocess the data before further analysis. Data cleaning involves several steps, outlined below with a detailed explanation of the provided code (import re / def clean_text(text): ...) that implements them.

To implement this, the combined dataset CSV file was placed into the Pandas dataframe for further preprocessing.

```
# Read the CSV file into a DataFrame
Nissan_Comments = pd.read_csv("NissanLeaf_comments_merged.csv")

Nissan_Comments
```

author	body	timestamp	n_words	chars
stortag	I know a guy who made a business out of this v...	2020-10-30 11:17:54	46	268
Car-face	Nissan Leaf sales in Canada peaked in 2018 at ...	2020-10-30 10:39:58	69	435
Idiot_Savant_Tinker	I looked at a leaf at a local nissan dealer. $...	2020-10-30 11:20:54	30	144
wewewawa	“I love the car,” he explains. “Honestly, in t...	2020-10-30 08:16:16	42	243
generaljimdave	A competent high voltage technician could refu...	2020-10-30 11:59:43	125	676
...	...	...	...	...

Figure 4 unfiltered scraped data

The first step was cleaning the data, which involved removing any extra spaces, converting the text to lowercase, and removing special characters and HTML tags using regular expressions. For instance, re.sub('\[.*?\]', '', text) was used to remove text within square brackets.

The second step entailed normalisation, which included removing usernames, URLs, and punctuation. This step also kept only letters, numbers, spaces, and tabs and split the text into a list of words, removing characters using another regular expression and then rejoining the list into a single string with spaces.

The code also incorporated additional steps to convert the input to a string type, remove extra spaces, convert all characters to lowercase, remove text enclosed in square brackets, remove usernames, remove URLs, remove punctuation and normalise text, remove HTML tags, remove newline characters, remove words containing numbers, and finally, return the

cleaned text. The image below showed the snippet code used for the data-cleaning pipeline.

```python
def clean_text(text):

    # Convert text to lowercase
    text = text.lower()
    # This will remove links (URLs)
    text = re.sub(r'https?://\S+|www\.\S+', '', text)
    # This will remove text in brackets
    text = re.sub(r'\[.*?\]', '', text)
    # This will remove usernames (e.g., '@username')
    text = re.sub(r'@[^\s]+', '', text)
    # This will remove special characters and punctuation
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # This will remove words containing numbers
    text = re.sub(r'\w*\d\w*', '', text)
    # This will remove HTML tags
    text = re.sub(r'<.*?>', '', text)
    # This will remove newline characters
    text = re.sub(r'\n', '', text)

    return text
```

Figure 5 data cleaning pipe line

After creating the pandas data frame named Nissan_comments, the defined function clean text is applied.

```python
# the the data cleaning pipleline is applied to the body of the panda dataframe
Nissan_Comments['body'] = Nissan_Comments['body'].apply(lambda x:clean_text(x))
Nissan_Comments
```

body	timestamp	n_words	chars
i know a guy who made a business out of this v...	2020-10-30 11:17:54	46	268
nissan leaf sales in canada peaked in at sal...	2020-10-30 10:39:58	69	435
i looked at a leaf at a local nissan dealer ...	2020-10-30 11:20:54	30	144
love the car he explains honestly in three y...	2020-10-30 08:16:16	42	243
a competent high voltage technician could refu...	2020-10-30 11:59:43	125	676

Figure 6 output of the cleaned data

DATA PREPROCESSING

The cleaned data is passed to NLTK (Natural Language Toolkit) library for text preprocessing tasks. It begins by importing the necessary modules: NLTK itself, pandas for data manipulation, and specifically, the word_tokenize function from NLTK's tokenise module, and the stopwords corpus from NLTK's corpus.

```python
# importing the natural language processing tool kit
import nltk
nltk.download('punkt')
import pandas as pd
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
```

Figure 7 Code snippet for tokenisation and stopword removal

The primary purpose of the code is to define a function called tokenize text, which serves to tokenise and clean text data by removing stopwords and converting the text to lowercase. Within the function, it first initialises a set of English stopwords using NLTK's stopwords.words('english') function. Stopwords are common words in a language, such as "a", "its", "and" that are often removed from text data during preprocessing as they do not typically contribute significant meaning to the text. This is followed by tokenising the input text using the word_tokenize function from NLTK, which splits the text into individual words or tokens. After tokenisation, it filters out stopwords and converts the remaining tokens to lowercase using a list comprehension. This results in a list of tokens containing only meaningfulwords

```
# Define a function for tokenization and removing stopwords
def tokenize_text(text):
  """
The function removes stopwords, converts text to lowercase and returns a list of tokens.
  """
  stop_words = set(stopwords.words('english'))  # Download stopwords for English
  tokens = word_tokenize(str(text))  # This Tokenizes the text into words
  filtered_tokens = [word.lower() for word in tokens if word.lower() not in stop_words]  # Lowercase and remove stopwords
  return filtered_tokens
```

Figure 8 code to display the output

Finally, using the apply method, we applied the tokenize_text function to the 'body' column of the Nissan Comments DataFrame. This effectively tokenises the text data in the 'body' column and creates a new column called 'tokenized_text' in the DataFrame to store the tokenised representations of the original text.

```
#Apply tokenization to the cleaned text column in Nissan_Comments DataFrame
Nissan_Comments['tokenized_text'] = Nissan_Comments['body'].apply(tokenize_text)
Nissan_Comments
```

Evidence of final dataset

author	body	timestamp	n_words	chars	tokenized_text
stortag	know guy made business problem upgrades older ...	2020-10-30 11:17:54	46	268	[know, guy, made, business, problem, upgrades,...
Car-face	nissan leaf sales canada peaked sales next clo...	2020-10-30 10:39:58	69	435	[nissan, leaf, sales, canada, peaked, sales, n...
diot_Savant_Tinker	looked leaf local nissan dealer wanted get tes...	2020-10-30 11:20:54	30	144	[looked, leaf, local, nissan, dealer, wanted, ...
wewewawa	love car explains honestly three years km repl...	2020-10-30 08:16:16	42	243	[love, car, explains, honestly, three, years, ...
generaljimdave	competent high voltage technician could refurb...	2020-10-30 11:59:43	125	676	[competent, high, voltage, technician, could, ...
...	...	...	...	...	...

Figure 9 cleaned dataset

EXPLORATORY ANALYSIS

The final dataset consists mainly of English content (1585), with a few subreddits in Somali (10) and some entries (8) where the language could not be determined. The dataset also includes texts in other languages, such as Dutch, Finnish, Danish, Welsh, and German. However, the presence of these languages is relatively low compared to English, which shows the Redditors used English to express their sentiment.

```
df["lang"].value_counts().head(10)
```

```
lang
en         1585
so           10
Unknown       8
ca            5
nl            4
fi            4
da            4
cy            4
de            3
id            2
Name: count, dtype: int64
```

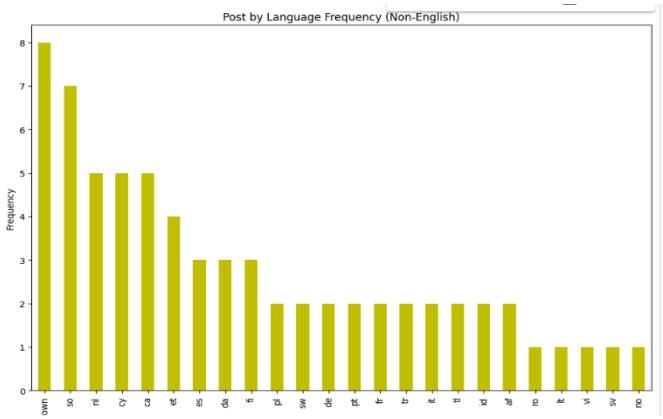

Figure 10 language frequency

The dataset contains popular words and phrases and their frequencies. The most used individual terms are "car" and "leaf," with frequencies of 450 and 389, respectively. This suggests a significant focus on cars, particularly the Nissan Leaf. Other commonly used terms include "like," "get," and "battery," which point to recurring themes and discussions within the dataset. The mentions of specific brands like "Nissan" and "Tesla" suggest the prominence of certain automotive manufacturers in the discussions.

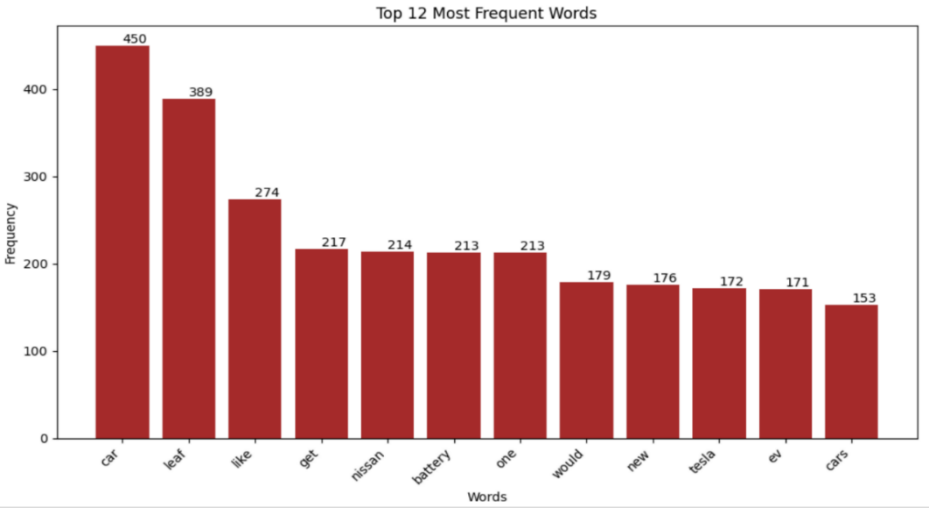

Figure 11 top 10 language frequency

In the list of phrases on our bigram analysis, some notable combinations include "miles range," "fast charge," and "tax credit." These provide insights into specific features or interests related to electric vehicles. Frequent bigrams "Road trip" and "road trips" imply discussions about the practicality and suitability of electric vehicles for long-distance travel. The recurring phrases "new leaf" and "new battery" suggest ongoing product updates or enhancements discussions. Additionally, mentions of competitor models such as the "Chevy Bolt" alongside generic references to "electric car" indicate a comparative analysis or broader discussions encompassing the electric vehicle market landscape.

	term	frequency
0	new leaf	13
1	miles range	9
2	new car	8
3	road trip	8
4	tax credit	8
5	fast charge	8
6	road trips	8
7	nissan leaf	7
8	base model	7
9	new battery	6

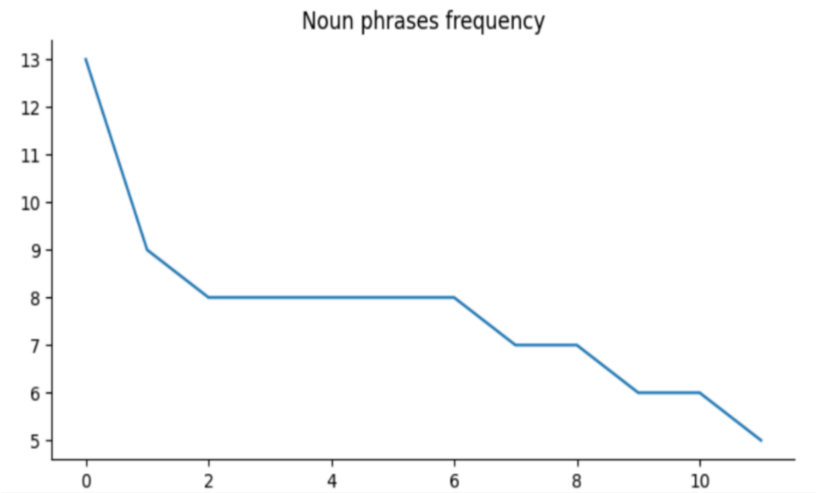

Figure 12 Noun_phrases frequencies

To further visualise and analyse the patterns and themes of the dataset's most frequently used words and phrases. Word Cloud, a popular tool, is used, as seen down below. Each word in the cloud is represented in varying font sizes, with the most common words

Figure 13 word cloud

Through the bag of word model, the correlation between "good," "bad," and "life" target words is examined by analysing a collection of text documents and converting them into a token count matrix. The model then calculates correlations between the target word and other words in the dataset, highlighting those that exhibit a strong positive correlation. For instance, the first set of correlations for "good" reveals highly correlated words such as "player," "lawn," and "violation," indicating frequent co-occurrence with "good" in the dataset. Similarly, the second set of correlations for "bad" shows words like "overreact," "takeaway," and "guest" having high positive correlations with "bad," suggesting a strong association. Lastly, for "life," words such as "indefinite," "block," and "watch" demonstrate significant positive correlations, implying a relationship within the dataset. These findings provide valuable insights into contextual associations and help identify related terms that may convey similar sentiments or themes.

```
Words with high correlations 'positive':
life          1.000000
indefinit     0.392139
block         0.323294
watch         0.291534
pz            0.275338
incorrectli   0.275338
ez            0.275338
smallest      0.275338
studi         0.273514
licens        0.266270
Name: life, dtype: float64
```

```
Words with high correlations 'positive':
bad           1.000000
overreact     0.277004
takeaway      0.277004
guest         0.277004
borderlin     0.277004
emphas        0.277004
remount       0.277004
gas           0.277004
frunk         0.277004
bless         0.277004
Name: bad, dtype: float64
```

```
Words with high correlations 'positive':
good          1.000000
player        0.261663
lawn          0.261663
violat        0.261663
mow           0.261663
asphalt       0.261663
hypothesi     0.261663
provabl       0.261663
valley        0.261663
concret       0.261663
Name: good, dtype: float64
```

Figure 14 word correlation output.

Based on a time-series analysis of comments and posts on Reddit, it is evident that the highest engagement was observed in March 2022, with February 2022 and July 2023 following closely. The analysis also indicates that there was significant discussion about the Nissan Leaf car between 2022 and 2023. These comments and posts were made between the end of 2020 and the beginning of 2024.

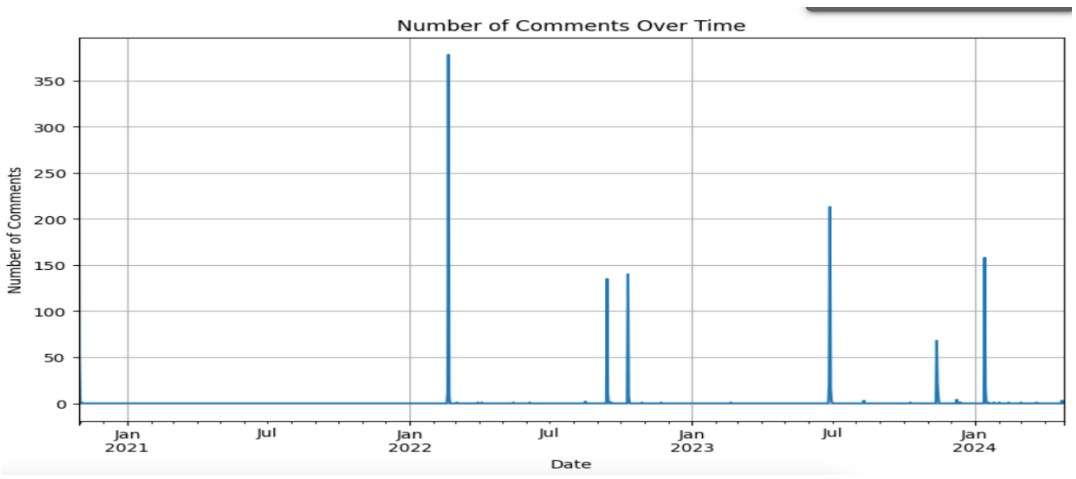

Figure 15 time series graph

Overall, this analysis provides valuable insights into the community's prevailing discussions, interests, and concerns about Nissan Leaf.

STEPS AND CODES

To begin the analysis of languages used, we utilised the langdetect library to detect the language present in each entry. Next, a custom function called "language_code" takes in a data row as input and applies the detect function to identify the language code for each text entry within the "body" column. To ensure accuracy, we included a try-except block within the function to handle any potential errors.

```
!pip install langdetect
from langdetect import detect
# installed langdetect then created a column to specify the language
def language_code(row):
  try:
    return detect(row["body"])
  except:
    return "Unknown" #this returns unknown if launguage is not recognised

df["lang"] = df.apply(language_code, axis=1)
```

Figure 16 code snippet for language fequency

Once the language identification function was applied to each row, a new column was generated that specified the language of each text entry. This enabled us to examine the distribution of languages within the dataset closely. To visualise the language used by non-English speakers, the code below was used to create a bar chart.

```
# visualizing the frequency of language
df['lang'].value_counts().iloc[1:].plot(kind="bar",
                                        figsize=(12,8), color="y",
                                        title="Post by Language Frequency (Non-Eng
                                        xlabel="Language Code",
                                        ylabel="Frequency")
plt.show()  # Display the plot
```

We utilised a code snippet for term frequency analysis that analyses the frequencies of words in the 'body' column of the Nissan_Comments DataFrame. We began by initialising a Counter object to count word occurrences accurately. Then, we iterated over each row to tokenise the text and update the Counter accordingly. After completing the analysis, we generated a visually informative DataFrame that highlights the top 12 most commonly used words and their respective frequencies. Finally, we created a bar chart that provides a helpful visualisation of the distribution of word occurrences.

```
[ ]  from collections import Counter

     word_counter = Counter()
     for row in Nissan_Comments.to_dict("records"): #converts_to_dictionary
       word_counter.update(row["body"].split()) #tokenization
     df_tf = pd.DataFrame(word_counter.most_common(12))
     df_tf.columns = ["term", "frequency"]
     df_tf
```

Figure 17 term frequency

For the bigram analysis of noun phrases in the 'body' column of the Nissan_Comments DataFrame, we used a Counter object to count the frequency of each noun phrase. We utilised the TextBlob library for noun phrase extraction. After extracting noun phrases, we generated a bar chart to display the top 10 most common phrases and created a DataFrame for tabular representation. This analysis provides insight into the most prevalent noun phrases within the dataset.

```
import nltk
nltk.download("brown")
nltk.download('punkt')
from collections import Counter
from textblob import TextBlob
import pandas as pd
# Initialize a Counter object to count the frequency of each noun phrase
noun_phrase_counter = Counter()
# Define a function to extract noun phrases from text
def extract_noun_phrases(text):
    blob = TextBlob(text)
    return blob.noun_phrases
# Apply the function to the 'body' column and update the Counter object
for text in Nissan_Comments['body']:
    noun_phrases = extract_noun_phrases(text)
    noun_phrase_counter.update(noun_phrases)
# Create a DataFrame from the Counter object containing the most common noun phrases
df_tf = pd.DataFrame(noun_phrase_counter.most_common(12), columns=["term", "frequency"])
```

```
from matplotlib import pyplot as plt
df_tf['frequency'].plot(kind='line', figsize=(8, 4), title='Noun phrases frequency')
plt.gca().spines[['top', 'right']].set_visible(False)
```

Figure 18 code snippet for bar chart

To perform word association analysis on the provided dataset, we started by importing pandas and CountVectorizer. CountVectorizer transforms text data into numerical values. Next, we calculated word frequencies and constructed a correlation matrix to identify associations between words. Finally, we selected a target word, 'good', and output the words highly correlated with it. We used the same process for target words like 'life' and 'bad'. This

process helped us explore the relationships between words within the dataset.

```
import pandas as pd
#Converts a collection of text documents to a matrix of token counts. It's useful for creating
from sklearn.feature_extraction.text import CountVectorizer

corpus = Nissan_Comments['tokenized_text'].tolist()
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(Nissan_Comments['tokenized_text'])
word_freq = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())

# Fit and transform the corpus to create the document-term matrix
dtm = vectorizer.fit_transform(corpus)
# Calculate correlation matrix
correlation_matrix = word_freq.corr()

# Analyze associations
target_word = 'life'
Highly_correlated_words = correlation_matrix[target_word].sort_values(ascending=False).head(10
print("Words with high correlations 'positive':")
print(Highly_correlated_words)
```

Figure 19 code snippet for word correlations

Below is a code snippet that aimed to visually represent the most commonly used words or phrases within a given text dataset. The text data was stored in a DataFrame named 'Nissan_Comments' with a column named 'body'. The code first concatenated all the text from the 'body' column into a single string variable called 'text'. Next, the WordCloud library created a Word Cloud object with specific parameters like width, height, and background colour. Finally, the Matplotlib library was used to display the resulting visualisation. By examining the resulting word cloud, insights into the most common terms used within the text corpus were gained, with larger font sizes indicating higher usage frequencies.

```
# To Analyse popular words or phrases (1) -
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Assuming df is my DataFrame with a column named 'body'

#BY Combining all text from the 'body' column into a single string
text = ' '.join(Nissan_Comments['body'].astype(str))

# Generate the word cloud
wordcloud = WordCloud(width=850, height=450, background_color='white').generate(text)

# To Display the word cloud using matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Figure 20code snippet for wordcloud

For the time series analysis, we started by importing the necessary Pandas and Matplotlib.pyplot for visualisation. Then, we used the "Nissan_Comments" data frame, which converts the "timestamp" column to datetime format using pd.to_datetime(). Next, we set the "timestamp" column as the index of the dataframe using the set_index() method to enable time-based analysis. Afterwards, we grouped the data by day with the resample() function,

specifying a frequency of "D" (day), and calculated the total number of comments per day using the size() method. Finally, we created a time series plot that displays the trend in the number of comments over time, complete with appropriate labels and grid lines to enhance readability. This process shows the patterns in comment activity, providing valuable insights into user engagement dynamics over time.

```
import pandas as pd
import matplotlib.pyplot as plt

# based on my dataframe Nissan_Comments

# convert the "timestamp" column to datetime format.
Nissan_Comments["timestamp"] = pd.to_datetime(Nissan_Comments["timestamp"])

# Set the "timestamp" column as the index
Nissan_Comments.set_index("timestamp", inplace=True)

# group the data by day and calculate the total number of comments per day.
comments_per_day = Nissan_Comments.resample("D").size()

# Plot the time series
plt.figure(figsize=(10, 6))
comments_per_day.plot()
plt.title("Number of Comments Over Time")
plt.xlabel("Date")
plt.ylabel("Number of Comments")
plt.grid(True)
plt.show()
```

Figure 21 code snippet for time series

TEXT MINING

The Latent Dirichlet Allocation (LDA) model is an effective tool that can be used to discover latent topics within a collection of textual data. In this particular analysis, the LDA model was applied to a dataset consisting of comments or posts related to electric vehicles (EVs), specifically focusing on discussions about the Nissan Leaf. The result of the LDA model produced ten distinct topics, each represented by a set of related words or phrases. These topics can offer insights into the primary themes present in the dataset and can aid in understanding the underlying structure of the text.

```
Topic 0:
car like want tv fit deal model drive set need
Topic 1:
car like make money sale time dealership think shit dealer
Topic 2:
credit tax bolt model mach chevi vehicl make tire look
Topic 3:
leaf batteri ev charg nissan rang car use cool like
Topic 4:
car leaf buy use good yeah model love know month
Topic 5:
car nissan chademo drive use year batteri make new got
Topic 6:
peopl replac vehicl tesla batteri buy new transport flat compani
Topic 7:
car leas tesla year want state leaf box buy incent
Topic 8:
car mile year cost price chang electr new like low
Topic 9:
charg trip road time leaf like tesla drive charger ev
```

Figure 22 tope 10 topics

Several key themes emerged after analysing the topics generated by the LDA model. For instance, Topic 0 explores cars' various features and specifications, including fitting, driving, and model types. Conversely, Topic 1 centres around the financial aspects of car ownership, covering sales, dealership experiences, and monetary considerations. Topic 2 encompasses various discussions related to financial and technical aspects, such as credit, taxes, vehicle models like Model 3 and Bolt, and tire-related issues. Additionally, Topic 4 covers general discussions about car ownership, including purchasing decisions, satisfaction levels, and usage patterns. In Topic 7, discussions revolve around leasing and incentives related to electric vehicles, including mentions of Tesla, Leaf, and state-specific incentives. Meanwhile, Topics 5 and 8 focus on driving experiences with electric vehicles, with mentions of features like charging, Nissan Leaf, and new vehicle acquisitions, and discussions related to the cost and price considerations of owning electric vehicles, including factors like mileage, electrification, and price changes, respectively.

Other topics, such as Topic 3, about electric vehicles' battery and charging aspects, with a particular focus on the Nissan Leaf and its battery range and usability. Similarly, Topic 9 focuses on charging infrastructure and related experiences during road trips, mentioning EV charging, Tesla, and Leaf drives.

...	topic_0	topic_1	topic_2	topic_3	topic_4	topic_5	topic_6	topic_7	topic_8	topic_9
...	0.004763	0.354790	0.362512	0.125705	0.004762	0.004763	0.004763	0.128417	0.004763	0.004762
...	0.855408	0.002857	0.121729	0.002858	0.002858	0.002858	0.002858	0.002857	0.002858	0.002858
...	0.367280	0.007694	0.007695	0.007693	0.007695	0.007694	0.007695	0.442605	0.136254	0.007695
...	0.963991	0.004001	0.004000	0.004001	0.004002	0.004001	0.004001	0.004001	0.004001	0.004001
...	0.149157	0.002000	0.711353	0.002001	0.002000	0.002000	0.002000	0.002000	0.125487	0.002000

Figure 23 topic vs cleaned relation

Moreover, the developed LDA model produces topic weights for each document within the dataset. These weights illustrate the extent to which a document is related to a particular topic. The accompanying pie chart exhibits the distribution of Reddit comments across the ten topics. The largest segment, comprising 12% of the chart, pertains to Topic 6, which explores replacing vehicle components - with notable mentions of Tesla and company transport. Conversely, the smaller segments represent less frequently discussed topics. Notably, the least discussed topic, Topic 3, accounts for 7% of the chart and revolves around electric vehicles' battery and charging aspects, with a specific focus on the Nissan Leaf and its battery range and usability. Nissan Leaf and its battery range and usability.

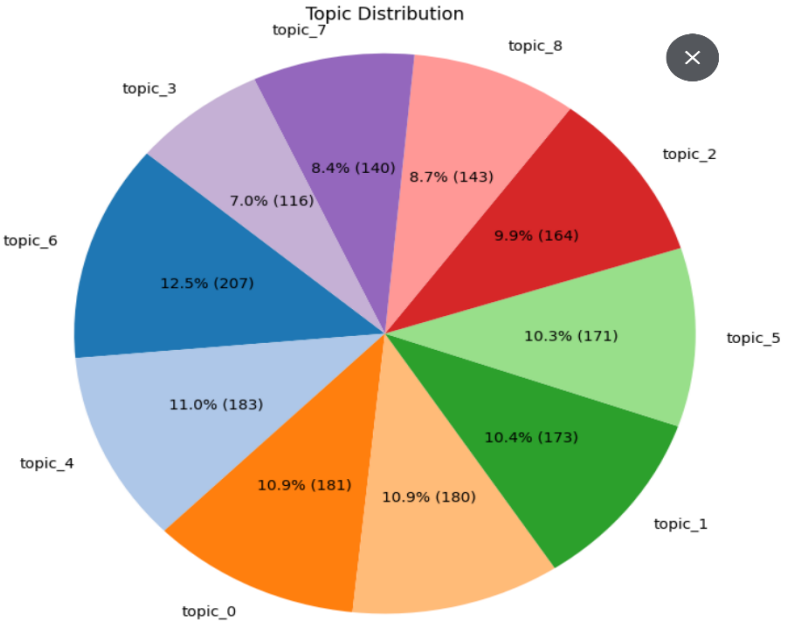

Figure 24 pie chart

By implementing the Vader lexicon model, we could gauge users' sentiment based on the text data they provided. The model effectively categorised the text into various sentiment categories based on their compound scores. Negative compound scores denote negative sentiment, whereas positive compound scores indicate positive sentiment.Texts with compound scores near zero are classified as neutral, revealing the overall sentiment polarity.

	n_words	chars	positive	negative	neutral	compound	sentiment
0	46	268	0.132	0.780	0.088	-0.0772	Negative
1	69	435	0.136	0.809	0.055	-0.3612	Negative
2	30	144	0.000	0.704	0.296	0.4939	Positive
3	42	243	0.000	0.840	0.160	0.6369	Positive
4	125	676	0.119	0.788	0.093	-0.4215	Negative
...	...	...	...	...	...	...	...

Figure 25 sentiment score

After analysing the data, we learned that most users had positive feedback towards the Nissan Leaf, as illustrated by the bar chart. We also observed that approximately 300 negative and 200 neutral sentiments were expressed. The dispersion of sentiment categories within the dataset gave us a better comprehension of the overall sentiment conveyed in the text data. This knowledge can be used to delve deeper into sentiment trends and patterns.

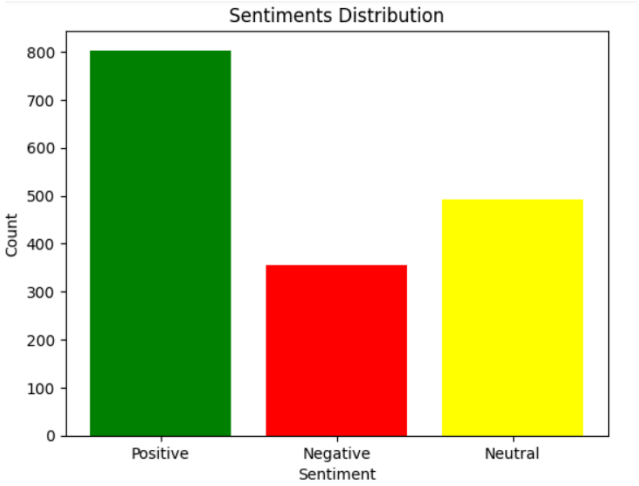

Figure 26 sentiment distributions

To assess the performance of our model, we utilised the Silhouette Score metrics. This measures how cohesive an object is within its own cluster compared to other clusters. A negative score indicates poor definition or overlap between clusters, suggesting that the data points may not be well-separated. Unfortunately, our model produced a Silhouette Score of -0.0747, indicating that it is not performing as expected.

We also looked at the Perplexity Score to further evaluate the topic model's performance. This measures how well the model predicts the given data, and our score of 741.64 suggests that, on average, the model is somewhat uncertain about predicting the next word in the sequence and may struggle to capture the underlying structure of the text data accurately. This could be due to the complexity or ambiguity of the dataset.

Based on these evaluation metrics, it seems that the clustering or topic modelling algorithm may not function at its best and may need additional refinement or exploration of alternative methods to represent discussion themes and owner sentiments adequately.

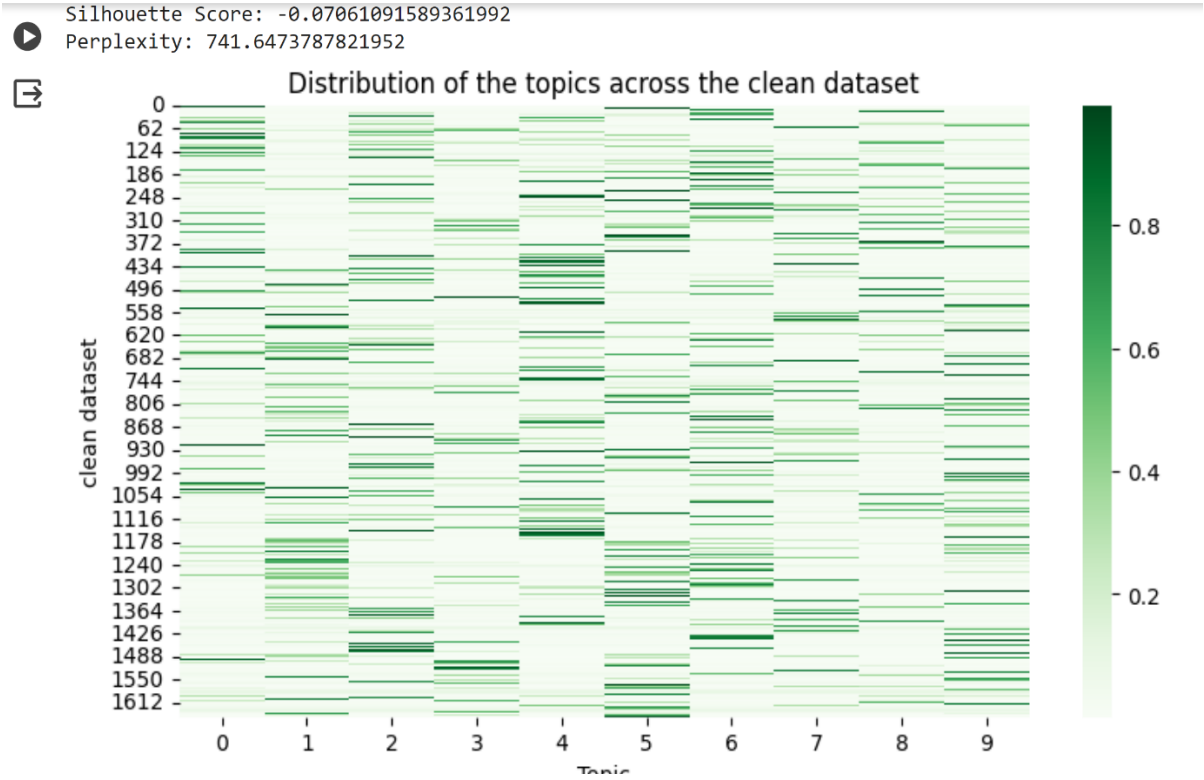

Figure 27 topic distribution

Based on our analysis of topic coherence, we found that the LDA coherence score was 0.6222. This score measures how well the topics generated by the LDA model fit together semantically. A higher score means that the topics are more coherent and easier to understand, showing a stronger connection between the words within each topic. Our score

of 0.6222 suggests a moderate level of coherence, indicating that the topics identified by the LDA model are reasonably consistent in their meaning.

STEPS AND CODES

The first step involved importing important libraries such as pandas for data manipulation ntlk for natural language processing tasks. SentimentIntensityAnalyzer, To perform sentiment analysis, was imported from the nltk.sentiment.vader module, which enabled sentiment scoring. CountVectorizer from sklearn was also used to convert text data into a numerical document-term matrix, which was necessary for further analysis. In addition, the VADER lexicon was downloaded to aid in sentiment analysis.

```
# started by importing the necessary library
import pandas as pd
import nltk
# from natuarl language took kit import sentimentintensityanalyzer used to calculate the score
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#This convert the text data to numerical data
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
#  downloads the VADER lexicon
nltk.download('vader_lexicon')
```

Figure 28 code snippet for vader lexicon

Each text snippet in the "body" column of the dataset underwent sentiment analysis, and the resulting scores were extracted into separate columns for positive, negative, neutral, and compound sentiments. A new column called "sentiment" was created to categorise each text's sentiment based on its compound score. Next, LatentDirichletAllocation (LDA) was utilised to perform topic modelling on the text data. The LDA model was fitted to the document-term.

```
# Initialize Sentiment Analyzer
sia = SentimentIntensityAnalyzer()

# Perform sentiment analysis on each text snippet in the column
sentiment_scores = Nissan_Comments['body'].apply(lambda x: sia.polarity_scores(x))

# Extract sentiment scores into separate columns
Nissan_Comments[['positive', 'negative', 'neutral', 'compound']] = pd.DataFrame(sentiment_scores.tolist())

# A new column to clearly state the sentiment of the text based on the compound score
Nissan_Comments['sentiment'] = ''
Nissan_Comments.loc[Nissan_Comments['compound'] > 0, 'sentiment'] = 'Positive'
Nissan_Comments.loc[Nissan_Comments['compound'] == 0, 'sentiment'] = 'Neutral'
Nissan_Comments.loc[Nissan_Comments['compound'] < 0, 'sentiment'] = 'Negative'

# we use "body" column of  cleaned nissan_comments text data for LDA
text_data = Nissan_Comments['body']
# Convert text data to  numerical document-term  matrix
vectorizer = CountVectorizer(max_features=1000, lowercase=True, stop_words='english')
doc_term_matrix = vectorizer.fit_transform(text_data)
```

Figure 29 code snippet for vader lexicon II

The top words for each topic were then printed, and topic probabilities were assigned to each document. These probabilities were added to the DataFrame to facilitate further analysis and interpretation.

```
# Print the top words for each topic
feature_names = vectorizer.get_feature_names_out()
for idx, topic in enumerate(lda_model.components_):
    print("Topic %d:" % (idx))
    print(" ".join([feature_names[i] for i in topic.argsort()[:-10 - 1:-1]]))

# Assign topic probabilities to each document
topic_probabilities = lda_model.transform(doc_term_matrix)

# Add topic probabilities to DataFrame
for i in range(num_topics):
    Nissan_Comments['topic_{}'.format(i)] = topic_probabilities[:, i]

# Display the DataFrame with added topic probabilities
print(Nissan_Comments)
Nissan_Comments.to_csv('Nissan_Comments_with_topics.csv', index=False)
```

Figure 30 code snippet for top words for each topic

This code snippet computes the topic distribution of the Nissan_Comments dataset by importing essential libraries. The resulting pie chart displays the proportion of each topic, with the corresponding comment count listed underneath. This visualisation depicts the prevalence of different conversation topics in the dataset.

```
# import math library
import math
topic_cols = [col for col in Nissan_Comments.columns if col.startswith('topic_')]

topic_distribution = Nissan_Comments[topic_cols].sum().sort_values(ascending=False)

# Plot a pie chart for topic distribution with number of comments underneath percentage

plt.figure(figsize=(8, 8))

wedges, texts, autotexts = plt.pie(topic_distribution, labels=topic_distribution.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.tab20.colors

# Add number of comments rounded up underneath percentage

for i, (text, autotext) in enumerate(zip(texts, autotexts)):

    rounded_count = math.ceil(topic_distribution[i])

    autotext.set_text(f"{autotext.get_text()} ({rounded_count})")

plt.title('Topic Distribution')

plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

Figure 31 code snippet for pie charts

To display the sentiment distribution in the Nissan_Comments dataset, the matplotlib library is utilised for graphical representation. The first step involved filtering the dataset based on the sentiment label and counting the number of positive, negative, and neutral occurrences. Next, two lists containing the sentiment labels ('Positive', 'Negative', and 'Neutral') and the corresponding sentiment category counts were created. These lists are then utilised as inputs to create a bar plot. Each sentiment category is represented by a unique colour (green for positive, red for negative, and yellow for neutral). The resulting plot provides a visual representation of the sentiment distribution in the dataset, which can help in understanding the overall sentiment composition

```
import matplotlib.pyplot as plt

# Calculate the counts of positive, negative, and neutral sentiments
positive_count = (Nissan_Comments['sentiment'] == 'Positive').sum()
negative_count = (Nissan_Comments['sentiment'] == 'Negative').sum()
neutral_count = (Nissan_Comments['sentiment'] == 'Neutral').sum()

# Create labels and counts for each sentiment category
sentiments = ['Positive', 'Negative', 'Neutral']
counts = [positive_count, negative_count, neutral_count]

# Create a bar plot
plt.bar(sentiments, counts, color=['green', 'red', 'yellow'])

# Add labels and title
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title(' Sentiments Distribution ')

# Show plot
plt.show()
```

Figure 32 code snippet for graph

The following code demonstrated a comprehensive method for conducting topic modelling with Latent Dirichlet Allocation (LDA) to analyse text data. To begin, necessary libraries such as pandas, sklearn, matplotlib, and seaborn were imported to facilitate data manipulation, topic modelling, and visualisation. Then, 'Nissan_Comments', the DataFrame containing text data and 'body' was the column storing the text entries, the text data was extracted. Next, the text data was transformed into a document-term matrix using CountVectorizer, where parameters were set to limit vocabulary size, convert text to lowercase, and remove English stopwords to enhance modelling efficiency.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns

text_data = Nissan_Comments['body']

# Convert text data to document-term matrix
vectorizer = CountVectorizer(max_features=1000, lowercase=True, stop_words='english')
doc_term_matrix = vectorizer.fit_transform(text_data)
```

Figure 33 code snippet for topic modelling

The LDA model was then applied to the document-term matrix, specifying the number of topics to extract. This step identified latent topics within the text data, providing insights into the underlying thematic structure. Once the LDA model was fit, topic probabilities for each document were computed, indicating the likelihood of each document belonging to different topics

```
# Convert text data to document-term matrix
vectorizer = CountVectorizer(max_features=1000, lowercase=True, stop_words='english')
doc_term_matrix = vectorizer.fit_transform(text_data)

# Apply LDA
num_topics = 10  # Specify the number of topics
lda_model = LatentDirichletAllocation(n_components=num_topics, random_state=42)
lda_model.fit(doc_term_matrix)

# Assign topic probabilities to each document
topic_probabilities = lda_model.transform(doc_term_matrix)
```

Figure 34 code snippet for topic modelling II

The LDA model's goodness of fit was evaluated using the silhouette score, a metric assessing the cohesion and separation of the identified clusters. Additionally, perplexity, a measure of how well the model predicted the sample text, as calculated to gauge the

model's performance. These evaluations aided in assessing the quality of the topic modelling process and provided insights into potential improvements.

```
# Evaluate goodness of fit using silhouette score
silhouette_avg = silhouette_score(doc_term_matrix, topic_probabilities.argmax(axis=1))
print("Silhouette Score:", silhouette_avg)
# Evaluate perplexity
perplexity = lda_model.perplexity(doc_term_matrix)
print("Perplexity:", perplexity)
# Visualize topic distribution using heatmap
```

Figure 35 code snippet for heatmap

A heatmap was generated using Seaborn to visualise the distribution of topics across the dataset. The heatmap provided a graphical representation of the topic probabilities assigned to each document, facilitating the interpretation of topic distributions and their relative prominence within the dataset.

```
plt.figure(figsize=(9, 5))
sns.heatmap(topic_probabilities, cmap='Greens')
plt.title('Distribution of the topics across the clean dataset')
plt.xlabel('Topic')
plt.ylabel('clean dataset')
plt.show()
```

Figure 36 creates plot

For the Coherence score, we imported necessary libraries from the Gensim package, including LdaModel, CoherenceModel, and Dictionary, which are essential for topic modeling. Next, we tokenised the text data from the Nissan_Comments dataset, splitting each document into individual tokens. This step prepared the data for further processing. We then created a dictionary to map each token to a unique ID, facilitating the transformation of the text data into a format suitable for topic modelling. Using the created dictionary and tokenised text data, we constructed a document-term matrix that represented the frequency of each token in the corpus of documents.

```
from gensim.models import LdaModel
from gensim.models import CoherenceModel
from gensim.corpora import Dictionary

# Convert text data to a list of tokenized documents
tokenized_text_data = [doc.split() for doc in Nissan_Comments['body']]

# Create a dictionary mapping tokens to IDs
dictionary = Dictionary(tokenized_text_data)
```

Figure 37 code snippet for cohorence score I

We trained the LDA (Latent Dirichlet Allocation) model on the document-term matrix to identify underlying topics within the text data. The model was configured to identify ten topics

and was initialized with a random seed for reproducibility. Finally, we computed a coherence score to evaluate the quality of the generated topics. This score measured the semantic similarity between words within each topic and provided insight into the interpretability and coherence of the identified topics. We assessed the effectiveness of the LDA model in capturing meaningful topics within the text corpus by printing the coherence score.

```
# Create a document-term matrix
corpus = [dictionary.doc2bow(doc) for doc in tokenized_text_data]

# Train LDA model
lda_model = LdaModel(corpus=corpus, id2word=dictionary, num_topics=10, random_state=42)

# Compute coherence score
coherence_model = CoherenceModel(model=lda_model, texts=tokenized_text_data, dictionary=dictionary, coherence='c_v')
coherence_score = coherence_model.get_coherence()
print("Coherence Score:", coherence_score)
```

CONCLUSION

In summary, analysing the electric vehicle industry with a focus on Fully Electric Vehicle (FEV) owners provides valuable insights into the community's prevailing trends, interests, and concerns. The industry's rapid evolution is driven by technological advancements, government regulations, and changing consumer preferences, as evidenced by the significant increase in global electric vehicle sales. Government initiatives, such as road user charging and tax incentives, promote electric vehicle adoption and mitigate environmental impact. Advancements in battery technology and charging infrastructure further enhance electric vehicle performance and practicality, addressing driving range and convenience concerns.

Our analysis examines the Nissan Leaf, a popular choice in the EV market, to comprehensively understand user sentiments and experiences. We scraped Reddit posts and comments to gather diverse opinions on the Nissan Leaf, covering reliability, affordability, and battery life. Preprocessing steps, including text cleaning and tokenisation, ensured the quality and accuracy of the dataset for further analysis. The exploratory analysis revealed insights into popular words and phrases, word associations, and sentiment trends among Nissan Leaf owners.

The Latent Dirichlet Allocation (LDA) model facilitated the discovery of latent topics within the text corpus, uncovering primary themes and discussions related to electric vehicles. We evaluated topic coherence and sentiment analysis to assess the LDA model's performance and understand the dataset's underlying structure. While the model exhibited moderate

coherence, further refinement and exploration of alternative methods may be necessary to improve clustering and topic modelling accuracy.

This comprehensive analysis provides valuable insights into the current discourse on sustainable transportation. It also offers practical recommendations for stakeholders in the electric vehicle ecosystem. By thoroughly examining user sentiments and addressing significant concerns, policymakers, manufacturers, and consumers can work collaboratively to promote the adoption of electric vehicles and accelerate the transition towards a cleaner and more sustainable future.

References

IEA (2023), Global EV Outlook 2023, IEA, Paris https://www.iea.org/reports/global-ev-outlook-2023, Licence: CC BY 4.0)

Rho Motion. (n.d.). *Q1 2024: Over 3 Million Electric Vehicles Sold Globally*. [online] Available at: https://rhomotion.com/news/q1-2024-over-3-million-electric-vehicles-sold-globally/ [Accessed 15 Apr. 2024].

UK Parliament (2023). *Research Briefings Archive*. [online] House of Commons Library. Available at: https://commonslibrary.parliament.uk/research-briefings.

Kim, S., Tanim, T.R., Dufek, E.J., Scoffield, D., Pennington, T.D., Gering, K.L., Colclasure, A.M., Mai, W., Meintz, A. and Bennett, J. (2022). Projecting Recent Advancements in Battery Technology to Next-Generation Electric Vehicles. *Energy Technology*, p.2200303. doi:https://doi.org/10.1002/ente.202200303.

Beren, D. (2023). *Nissan Leaf vs. Tesla Model 3: Which One Wins?* [online] History-Computer. Available at: https://history-computer.com/nissan-leaf-vs-tesla-model-3-which-one-wins/ [Accessed 15 Apr. 2024].

Carlier, M. (2022). *Topic: Electric Vehicle Market in the United Kingdom*. [online] Statista. Available at: https://www.statista.com/topics/2298/the-uk-electric-vehicle-industry/#topicOverview.

Lazareva, E.I. and Dong, Y. (2022). Features of Chinese Government Policy to Stimulate Demand for Electric Vehicles: The Willingness of Car Owners. *Innovative Trends in International Business and Sustainable Management*, pp.529–541. doi:https://doi.org/10.1007/978-981-19-4005-7_57.

EPRI Home. (no date). EPRI Home. Available from https://www.epri.com/research/programs/053122/overview [Accessed 15 April 2024].