

1. INTRODUCTION

1.1. ABSTRACT

Sentiment analysis, a pivotal task in natural language processing (NLP), holds profound implications for understanding consumer behavior and informing business strategies. Leveraging state-of-the-art deep learning architectures like BERT (Bidirectional Encoder Representations from Transformers), this study presents a comprehensive sentiment analysis framework using a vast dataset of Amazon reviews. Amazon, as a leading e-commerce platform, offers an extensive repository of user-generated content, making it an ideal resource for exploring consumer sentiments.

The primary objective of this research is to demonstrate the efficacy of the BERT model in discerning nuanced sentiments from Amazon reviews, encompassing both the polarity and intricacies of user opinions. Through fine-tuning BERT on the sentiment analysis task, we aim to extract meaningful insights that can drive actionable decisions for businesses.

Key components of the study include dataset acquisition and preprocessing, BERT model implementation, model training, evaluation, and interpretation of results. The Amazon review dataset is carefully curated, encompassing diverse product categories to ensure representative insights. Preprocessing steps involve text cleaning, tokenization, and data augmentation techniques to enhance model performance.

2. SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

Our system, "Amazon Sentiment Analyzer," is designed to perform sentiment analysis on Amazon reviews using machine learning techniques. It begins by acquiring Amazon review data, which is then preprocessed to clean and normalize the text.

2.2 PROPOSED SYSTEM

This study proposes a sentiment analysis system utilizing the VADER (Valence Aware Dictionary and sEntiment Reasoner) model to analyze sentiments within Amazon reviews. The VADER model offers a lexicon-based approach, making it computationally efficient for capturing sentiment polarity and intensity in textual data.

2.3 FEASIBILITY STUDY

The feasibility of conducting sentiment analysis using an Amazon review dataset appears promising on multiple fronts. From a technical standpoint, the availability of the dataset, along with robust NLP libraries and tools, ensures the necessary resources for analysis. Computational requirements can be managed efficiently through cloud computing services, offering scalability and cost-effectiveness. Economically, while there may be costs associated with data acquisition and computational resources, they are typically reasonable and justifiable given the potential insights gained.

2.3 FEASIBILITY STUDY

TYPES OF FEASIBILITY STUDY

Technical Feasibility

A technical feasibility study for sentiment analysis using an Amazon review dataset would involve several key steps to assess whether such a project is viable. Here's an outline of what the feasibility study might entail Determine if there is an available dataset of Amazon reviews that is suitable for sentiment analysis. Amazon itself does not readily provide review datasets, but there are third-party datasets available through platforms like Kaggle or academic sources.

Operational feasibility

An operational feasibility study for sentiment analysis using an Amazon review dataset focuses on determining whether the project can be effectively implemented within the existing operational framework of an organization. Evaluate the organization's current IT infrastructure, including hardware, software, and network capabilities. Determine if there are any existing tools or platforms that can be leveraged for sentiment analysis or if new infrastructure needs to be acquired.

Economic feasibility

This economic feasibility study assesses the viability of implementing sentiment analysis software to analyze Amazon review datasets. The analysis considers the costs associated with acquiring and processing the data, deploying sentiment analysis tools, and the potential benefits derived from actionable insights for businesses.

Legal feasibility

Legal feasibility for sentiment analysis using Amazon review dataset involves ensuring compliance with relevant laws and regulations governing data privacy, intellectual property rights, and terms of service. General Data Protection Regulation (GDPR): If the Amazon review dataset contains personal data of individuals in the European Union, compliance with GDPR is mandatory. This includes obtaining explicit consent for data processing, ensuring data security, and providing individuals with rights such as access, rectification, and erasure of their personal data.

Schedule feasibility

Schedule feasibility for sentiment analysis using an Amazon review dataset involves assessing whether the project can be completed within the allocated time frame. Here's a breakdown of the schedule feasibility considerations. Time Estimate: Depending on the size and complexity of the dataset needed for sentiment analysis, the process of acquiring the Amazon review dataset may take varying amounts of time.

3. SYSTEM REQUIREMENT SPECIFICATION

3.1. HARDWARE REQUIREMENTS

Processor Type	: intel dual core
Speed	: 2.40GHZ
RAM	: 2.00GB
Hard disk	: 30GB HD

3.2 SOFTWARE REQUIREMENTS

Programming Language	: Python
Data Processing Tools	: Pandas and Numpy
Machine Learning Libraries	: Scikit-learn
Text Processing Libraries	: NLTK and Spacy
Visualization Tools	: Matplotlib or Seaborn

3.3 ABOUT SOFTWARE

Sentiment analysis software leveraging Amazon review datasets involves a multi-step process. Initially, data collection entails gathering relevant Amazon reviews, either through Amazon's APIs or web scraping techniques. Subsequently, data preprocessing is conducted to clean and standardize the collected data, removing noise and normalizing text through techniques like tokenization and stemming.

Feature extraction follows, where text data is transformed into numerical features suitable for machine learning algorithms, commonly utilizing methods like TF-IDF or word embeddings. The model is then trained using various algorithms such as SVM, Naive Bayes, or deep learning models like RNNs or Transformers. Evaluation of the model's performance is crucial, employing metrics like accuracy and F1-score.

Finally, deployment into production allows for real-time or batch sentiment analysis of new Amazon reviews. This process can be facilitated by Python libraries such as Scikit-learn and NLTK, pretrained models like VADER and BERT, or managed services like Amazon Comprehend, ensuring compliance with relevant data privacy regulations throughout

Natural Language Processing (NLP) Techniques: Sentiment analysis software employs sophisticated NLP techniques to understand and interpret human language. These techniques include tokenization (breaking text into words or phrases), part-of-speech tagging (identifying the grammatical parts of speech), syntactic analysis (parsing the grammatical structure of sentences), and semantic analysis (extracting meaning from text).

4. SYSTEM DESIGN

4.1 MODULES LIST

The project contain following modules.

They are

- Import libraries
- Import dataset
- Cleaning data
- Data Preprocessing
- Fine-tuning BERT Model
- Training Script
- Inference Module
- Visualization
- Monitoring and Maintenance

4.2 MODULE DESCRIPTIONS

Import Libraries

In this module, essential libraries and dependencies are imported into the project environment. These libraries typically include frameworks like TensorFlow or PyTorch for deep learning, as well as specialized libraries such as Transformers for working with BERT models. Additional utilities for data manipulation, visualization, and evaluation may also be imported here.

Import Dataset

This module focuses on importing the Amazon review dataset into the project environment. Depending on the dataset's format and size, various techniques may be employed for data loading, such as reading data from CSV files, querying databases, or utilizing APIs provided by data repositories like Kaggle or Amazon itself.

Cleaning Data

Before proceeding with analysis, it's essential to clean the dataset to ensure its quality and reliability. This module involves identifying and handling missing values, duplicate entries, and inconsistencies in the data. Techniques such as data imputation, deduplication, and standardization may be applied to enhance the dataset's cleanliness.

Data Preprocessing

This module prepares the cleaned dataset for training the BERT model by performing various preprocessing tasks. Textual data is tokenized, segmented into sub words, and converted into numerical representations suitable for input to the model. Additionally, padding and truncation techniques are applied to ensure uniformity in input sequence lengths, facilitating efficient processing by the BERT model.

Fine-tuning BERT Model

Fine-tuning the BERT model involves adapting its pre-trained parameters to the task of sentiment analysis on the Amazon review dataset. During fine-tuning, specific layers of the BERT architecture may be frozen or adjusted, and task-specific layers are added to optimize the model's performance for sentiment classification.

Training Script

This module contains the script responsible for training the fine-tuned BERT model on the preprocessed Amazon review dataset. It specifies various training hyperparameters such as learning rate, batch size, and number of epochs. The script orchestrates the training process, iteratively updating the model's parameters based on computed gradients and the selected optimization algorithm.

Inference Module

Once the model is trained, the inference module handles the process of making predictions on new Amazon reviews. Given a new review as input, the fine-tuned BERT model predicts the sentiment label (e.g., positive, negative, neutral) associated with the review. This module is crucial for real-time or batch sentiment analysis tasks.

Visualization

Visualization is a key aspect of understanding and interpreting the model's predictions and performance. This module encompasses generating visualizations such as confusion matrices, ROC curves, and precision-recall curves to gain insights into the model's behavior and

efficacy. Visualizations aid in communicating results to stakeholders and guiding further analysis and optimization efforts.

Heatmaps:

Heatmaps visualize sentiment associations between entities, such as words or topics. They use color gradients to represent sentiment polarity or strength, allowing users to identify relationships between different entities and sentiments within the text data.

Scatter Plots:

Scatter plots can be used to visualize sentiment scores or sentiments expressed by different entities (e.g., users, products, or topics) in a multidimensional space. They help identify clusters or patterns in sentiment expression and can be useful for exploring relationships between multiple variables.

Network Graphs:

Network graphs visualize relationships between entities within the text data, such as co-occurrence of words or topics. Nodes represent entities (e.g., words or topics), and edges represent relationships between them (e.g., co-occurrence). These visualizations help in identifying key entities and their associations within the text data.

Emotion Intensity Plots:

Emotion intensity plots visualize the intensity or strength of emotions expressed within the text data. They can display the distribution of emotions (e.g., joy, sadness, anger) and their intensity levels, providing insights into the emotional tone of the text.

By employing these visualization techniques, analysts can gain deeper insights into the sentiment expressed within textual data, identify patterns and trends, and make informed decisions based on the findings. Additionally, interactive visualizations allow for exploration and manipulation of data, facilitating a more nuanced understanding of sentiment dynamics.

Time series analysis

Time series analysis in sentiment analysis with Amazon review data entails the systematic examination of sentiment trends over time within the dataset. After collecting and preprocessing the reviews, sentiment analysis techniques are employed to categorize each review as positive, negative, or neutral. Subsequently, time series analysis techniques are applied to uncover patterns in sentiment fluctuations over different time intervals, such as days, weeks, or months. This involves decomposing the time series data, smoothing techniques to remove noise, analyzing trends, identifying seasonality, and possibly forecasting future sentiment trends. By interpreting these analyses, businesses can gain valuable insights into how sentiment evolves over time, enabling them to make informed decisions regarding product management, marketing strategies, and custom

Word cloud

A word cloud is a visual representation of text data, where words are displayed in varying sizes depending on their frequency or importance within the text. Typically, the more frequently a word appears in the text, the larger and more prominent it appears in the word cloud. This technique is often used to quickly and intuitively visualize the most common words or themes in a corpus of text, such as customer reviews, survey responses, or social media posts. Word clouds offer a convenient way to identify key topics or sentiments within a large body of text, providing valuable insights at a glance. However, it's important to note that word clouds do not provide detailed quantitative analysis and may overlook nuances in the data, such as context or sentiment polarity. Therefore, while word clouds serve as a useful exploratory tool, they are most effective

when used in conjunction with other text analysis techniques for a comprehensive understanding of the underlying textual data.

Monitoring and Maintenance

After deploying the sentiment analysis system, monitoring and maintenance are essential for ensuring its continued effectiveness and reliability. This module involves tracking key performance metrics, detecting any degradation in model performance or drift in data distribution, and triggering retraining or updating the model as needed. Regular maintenance tasks such as updating dependencies, patching security vulnerabilities, and optimizing resource utilization are also part of this module.

4.2 FUNCTION REQUIREMENTS

4.2.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a visual representation of the flow of data within a system, consisting of processes, data flows, data stores, and external entities. Processes represent activities that manipulate data, while data flows depict the movement of data between processes, data stores, and external entities.

Data stores serve as repositories for storing data within the system, and external entities represent sources or destinations of data outside the system. DFDs can be organized into multiple levels of abstraction, ranging from high-level context diagrams to detailed diagrams focusing on specific processes or subsystems.

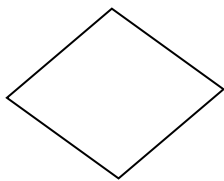
The basic notation are



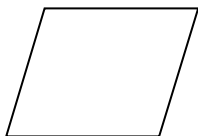
-Indicate the Flow of Data



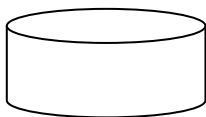
-Indicate the Process



-Indicate the Module



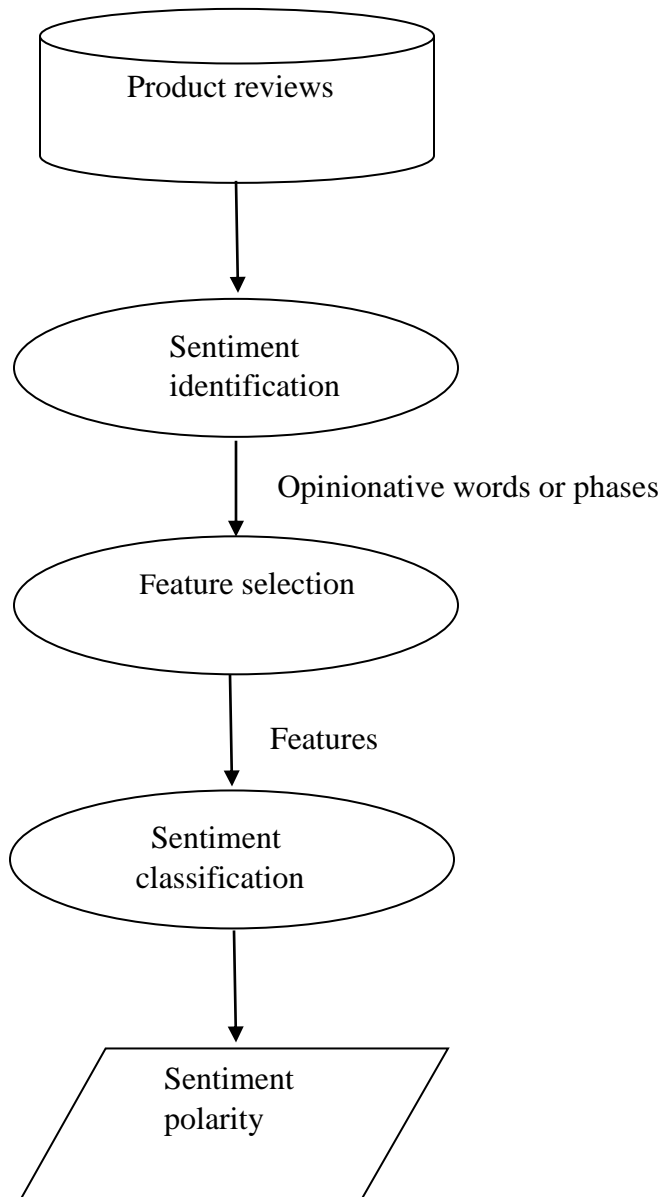
-Indicate the Module



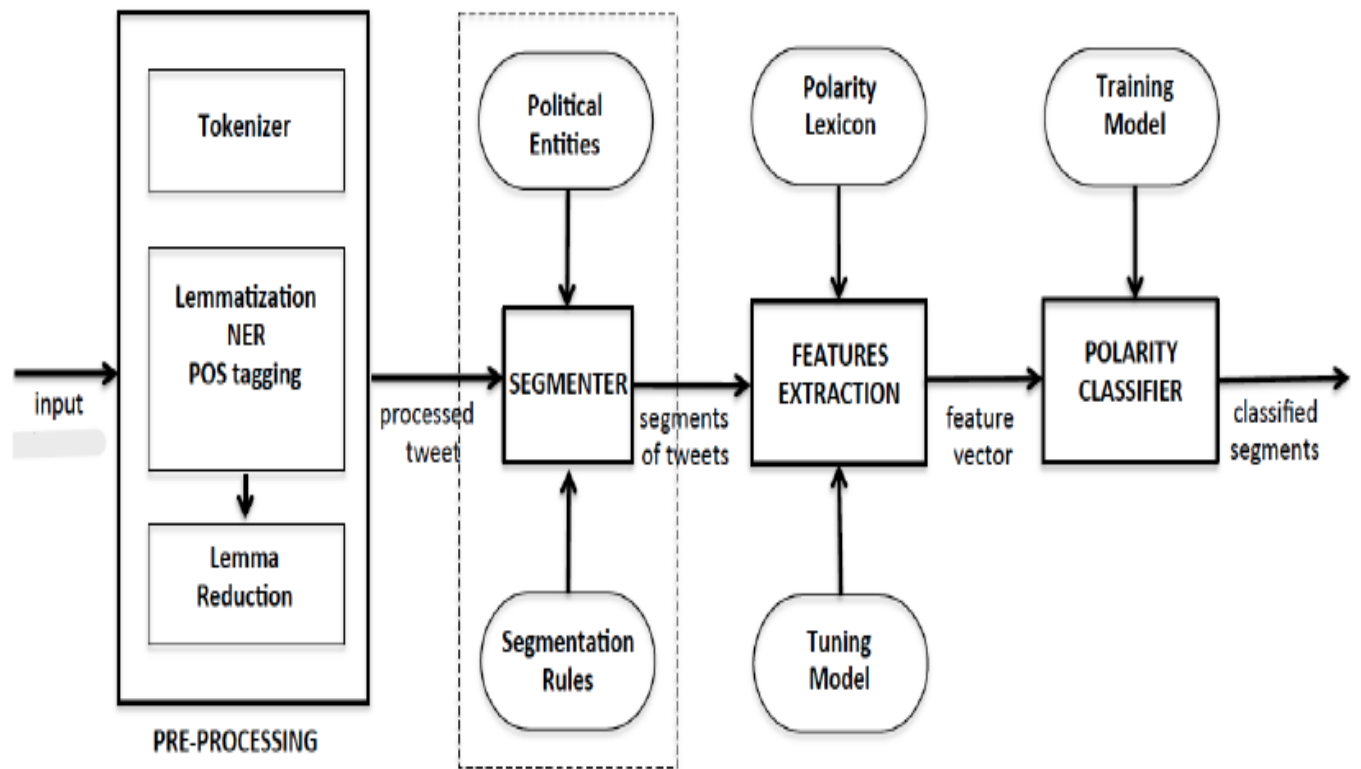
-Indicate the Database

DATAFLOW DIAGRAM

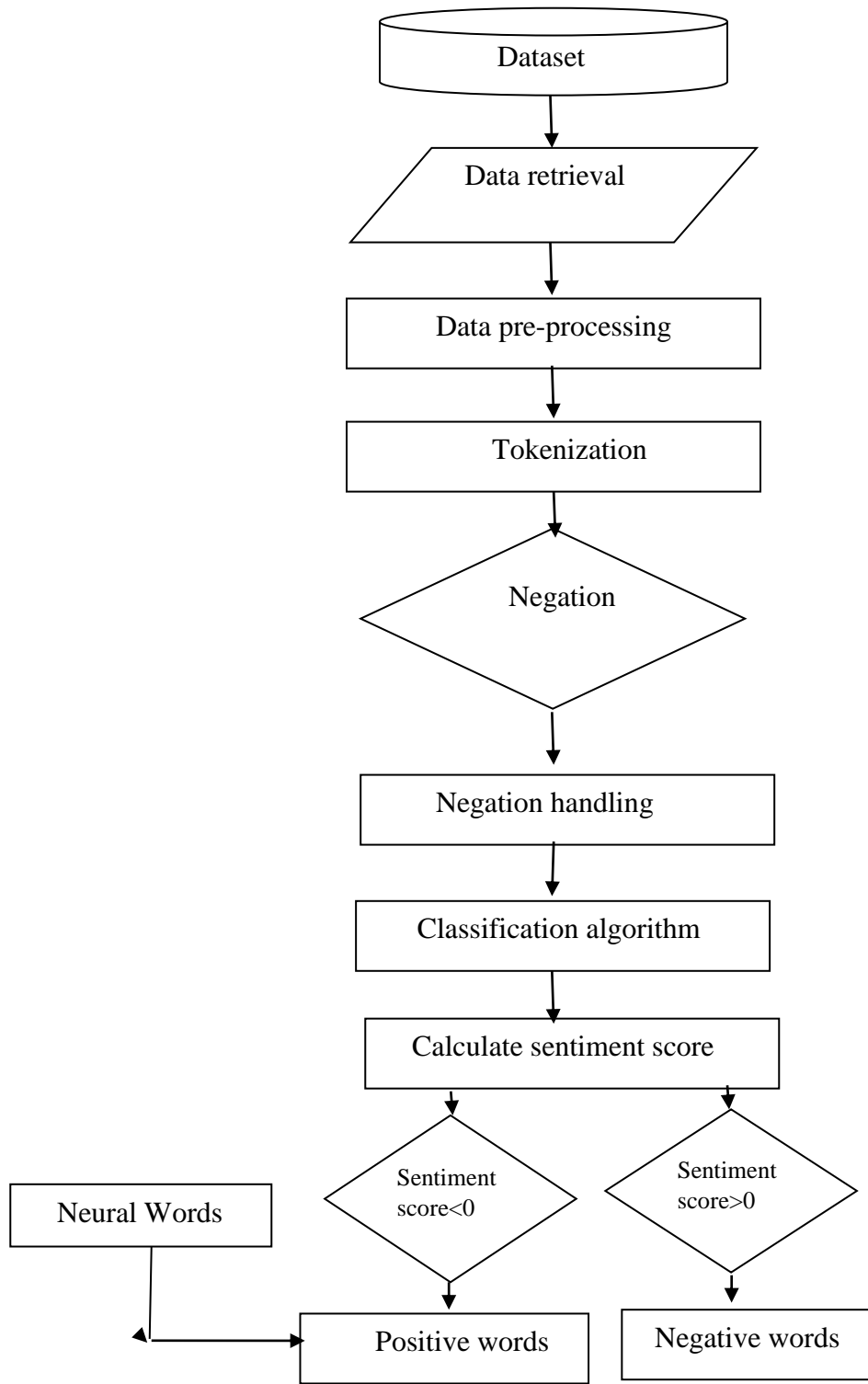
LEVEL 0



LEVEL 1



LEVEL 2



5. SYSTEM IMPLEMENTATION

In the age of digital commerce, where online shopping has become an integral part of our lives, the importance of understanding customer sentiment cannot be overstated. With millions of products available on e-commerce platforms like Amazon, consumers are inundated with choices, and their purchasing decisions are often influenced by the opinions and experiences shared by others.

Amazon, being one of the largest online marketplaces, serves as a treasure trove of valuable insights encapsulated within its vast repository of customer reviews. Sentiment analysis, a powerful application of natural language processing, emerges as a beacon of understanding amidst this sea of data.

By employing advanced algorithms to analyze the sentiments expressed in these reviews, we can unlock a wealth of knowledge regarding customer satisfaction, product performance, and market trends.

From discerning the overall sentiment towards a particular product or brand to identifying specific pain points and areas of improvement, sentiment analysis equips businesses with actionable intelligence to enhance their offerings, tailor their marketing strategies, and ultimately, foster greater customer loyalty.

6. SYSTEM TESTING

Unit Testing

Implementation: Write test cases for each function or method responsible for data preprocessing, feature extraction, model training, and prediction. For example, test that the text preprocessing function correctly converts text to lowercase, removes punctuation, and eliminates stop words. Ensure that the feature extraction function generates the expected TF-IDF vectors. Verify that the model training function initializes and trains the machine learning model without errors.

Edge case: Test edge cases such as empty input, special characters, or extreme values to ensure robustness and handle potential errors gracefully.

Integration Testing

Implementation: Test the flow of data between preprocessing, feature extraction, and model training/prediction stages. Ensure that the output of one component serves as the input to the next component seamlessly. For example, validate that preprocessed text data is correctly fed into the feature extraction process and the resulting features are utilized by the model for sentiment prediction.

Data Consistency: Check for consistency in data representation and format between different components to prevent data mismatches or compatibility issues.

Dataset Validation:

Implementation: Load a small subset of the dataset and perform preprocessing steps (e.g., lowercasing, punctuation removal, stop word removal). Manually inspect the preprocessed data to verify that it aligns with expectations. Check for missing values, anomalies, or inconsistencies in the dataset.

Data Integrity: Validate that the dataset contains the necessary columns or features required for sentiment analysis and that the data types are appropriate for each feature

Accuracy Testing:

Implementation: Run sentiment analysis on a significant portion of the dataset and compare predicted sentiment labels with actual sentiment labels. Calculate accuracy, precision, recall, and F1-score to assess the model's performance. Analyze misclassified instances to identify common errors and areas for improvement.

Confusion Matrix: Construct a confusion matrix to visualize the model's performance across different sentiment classes and identify specific patterns of misclassification.

Robustness Testing

Implementation: Introduce variations such as different review lengths, language variations, misspelled words, or noisy data to evaluate the system's robustness. Test how well the system adapts to these variations and whether it maintains accurate predictions under challenging conditions.

Error Handling: Verify that the system gracefully handles errors or unexpected input without crashing and provides informative error messages or warnings to users.

Scalability Testing:

Implementation: Gradually increase the size of the dataset and monitor resource utilization (e.g., memory, CPU usage) and execution time. Assess whether the system can process larger datasets efficiently without significant degradation in performance or resource constraints.

Batch Processing: Consider implementing batch processing techniques or distributed computing frameworks to improve scalability and handle large volumes of data effectively.

7. SYSTEM MAINTENANCE

System maintenance for sentiment analysis using an Amazon review dataset is a crucial aspect of ensuring the continued effectiveness and reliability of the system. Beyond the initial implementation phase, maintenance involves ongoing monitoring, optimization, and adaptation to evolving requirements and challenges.

It encompasses a spectrum of activities aimed at preserving the system's performance, enhancing its capabilities, and addressing any issues or inefficiencies that may arise over time.

This includes regular data updates to incorporate new reviews and trends, monitoring model performance metrics to detect any degradation in accuracy or robustness, and refining algorithms or preprocessing techniques to improve prediction quality. Additionally, maintenance involves staying abreast of advancements in machine learning and natural language processing techniques, incorporating new insights or methodologies to enhance the system's effectiveness. Continuous testing and validation are integral components of maintenance, ensuring that the system remains resilient to changes in data distribution, user behavior, or external factors that may impact sentiment analysis accuracy.

Furthermore, proactive measures such as scalability enhancements, infrastructure optimizations, and security updates are essential for maintaining system stability, efficiency, and security in the face of growing data volumes and evolving technological landscapes.

Ultimately, system maintenance for sentiment analysis using an Amazon review dataset is a dynamic and iterative process that requires ongoing dedication, expertise, and adaptability to sustain optimal performance and deliver actionable insights to users and stakeholders.

System maintenance for sentiment analysis using Amazon review datasets is an ongoing endeavor involving regular updates, performance monitoring, and algorithm refinement. By continuously optimizing the system's accuracy and adaptability, it ensures reliable insights for users amidst evolving data landscapes.

8. CONCLUSION

In conclusion, sentiment analysis using Amazon review datasets presents a powerful tool for extracting valuable insights from vast repositories of customer feedback. By leveraging natural language processing techniques and machine learning algorithms, businesses can gain a deeper understanding of consumer sentiments, preferences, and trends.

Through the analysis of sentiment-laden reviews, organizations can identify areas for improvement, tailor marketing strategies, and enhance product offerings to better meet customer needs. As technology continues to advance, sentiment analysis remains a crucial asset in the arsenal of data-driven decision-making, empowering businesses to stay competitive and responsive in today's dynamic marketplace.

Furthermore, the application of sentiment analysis extends beyond commercial interests, offering profound implications for social and cultural insights. Through the lens of Amazon review datasets, sentiment analysis provides a window into collective opinions and societal trends, reflecting broader patterns of consumer behavior and cultural attitudes. By analyzing the sentiment expressed in product reviews, researchers can uncover valuable insights into shifting societal values, emerging trends, and cultural phenomena.

This deeper understanding of sentiment dynamics not only enriches our comprehension of consumer behavior but also facilitates a more nuanced understanding of societal dynamics and human interactions in the digital age.

9. FUTURE ENHANCEMENT

In the realm of sentiment analysis using Amazon review datasets, the future holds promising avenues for enhancement. With advancements in natural language processing and machine learning, the landscape of sentiment analysis is poised for transformative evolution. Future enhancements may revolve around delving deeper into contextual analysis, leveraging multimodal data for richer insights, and fine-tuning algorithms for finer-grained sentiment classification. Real-time analysis capabilities will enable businesses to stay agile and responsive to changing consumer sentiments, while user feedback integration ensures continuous learning and improvement.

Moreover, as the ethical considerations surrounding AI algorithms become increasingly prominent, future enhancements will prioritize fairness, transparency, and privacy to maintain trust and integrity in sentiment analysis systems. As these enhancements unfold, sentiment analysis using Amazon review datasets will continue to empower businesses with invaluable insights, driving innovation and customer-centric strategies in the digital marketplace.

10. BIBLIOGRAPHY

BOOK REFERENCE

[1] Pang, Bo, and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." Proceedings of the 43rd annual meeting on association for computational linguistics. Association for Computational Linguistics, 2005.

[2] Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).

[3] Zhang, Xiang, et al. "Character-level convolutional networks for text classification." Advances in neural information processing systems. 2015.

[4] Socher Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." Proceedings of the 2013 conference on empirical methods in natural language processing. 2013.

WEB REFERENCE

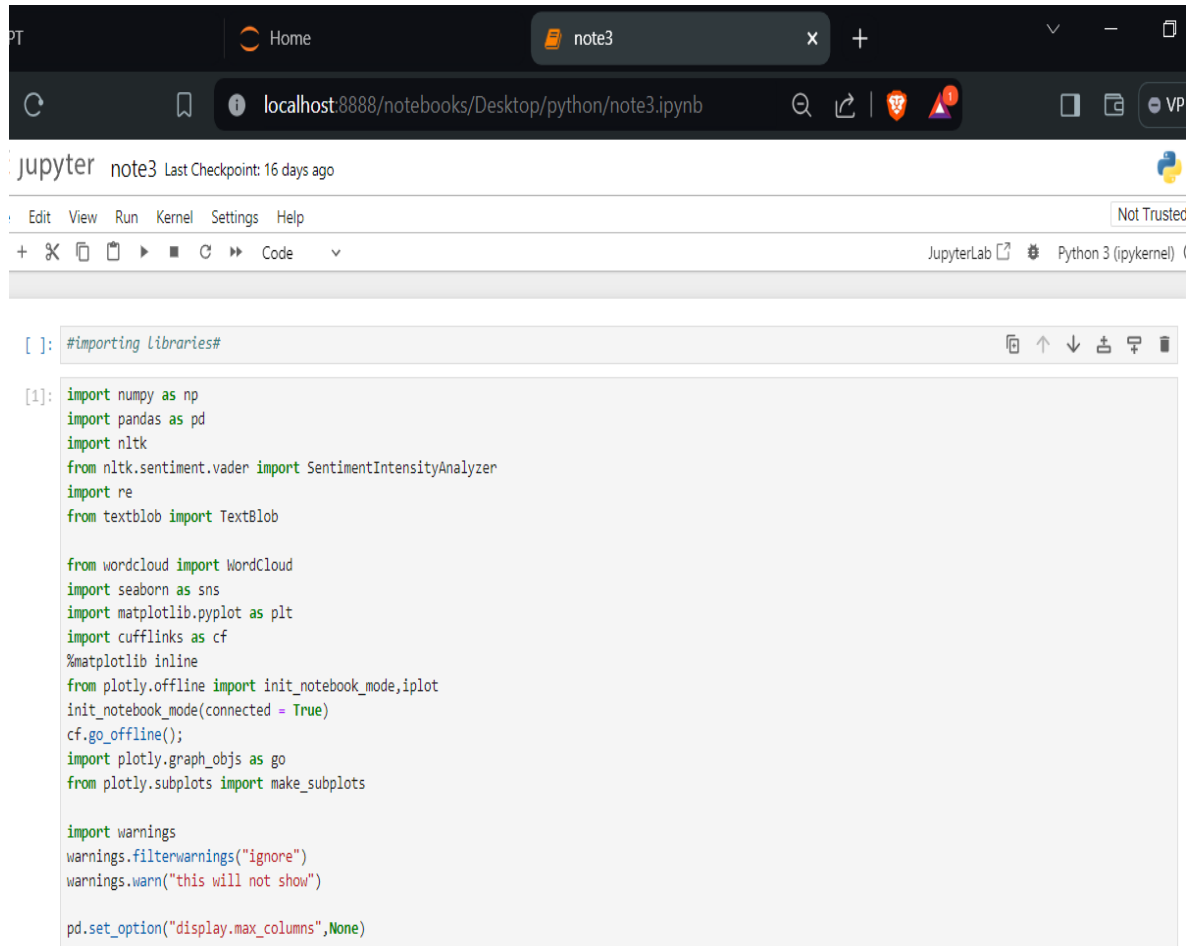
[1] <https://www.geeksfor.com>

[2] <https://www.hugging face sentimental analysis>

[3] <https://www.sentimental analysis by.org>

11. APPENDIX

11.1 SAMPLE SCREENSHOTS



The screenshot shows a JupyterLab interface in a web browser. The browser's address bar displays `localhost:8888/notebooks/Desktop/python/note3.ipynb`. The JupyterLab header includes the text "jupyter note3 Last Checkpoint: 16 days ago" and a "Not Trusted" warning. The main toolbar contains icons for file operations and a "Code" dropdown menu. The code editor displays the following Python code in a single cell:

```
[ ]: #importing libraries#  
  
[1]: import numpy as np  
import pandas as pd  
import nltk  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
import re  
from textblob import TextBlob  
  
from wordcloud import WordCloud  
import seaborn as sns  
import matplotlib.pyplot as plt  
import cufflinks as cf  
%matplotlib inline  
from plotly.offline import init_notebook_mode, iplot  
init_notebook_mode(connected = True)  
cf.go_offline();  
import plotly.graph_objs as go  
from plotly.subplots import make_subplots  
  
import warnings  
warnings.filterwarnings("ignore")  
warnings.warn("this will not show")  
  
pd.set_option("display.max_columns", None)
```

Figure 11.1.1 Import libraries

Home | note3 | localhost:8888/notebooks/Desktop/python/note3.ipynb | VPN

jupyter note3 Last Checkpoint: 17 days ago

File Edit View Run Kernel Settings Help | Not Trusted

JupyterLab | Python 3 (ipykernel)

```
[ ]: #importing Dataset
[2]: df=pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv")
[5]: df.head()
```

[5]:

Unnamed: 0	reviewerName	overall	reviewText	reviewTime	day_diff	helpful_yes	helpful_no	total_vote	score_pos_neg_diff	score_average_rating	wilson_lower_bou
0	0	NaN	4	No Issues.	23-07-2014	138	0	0	0	0	0.0
1	1	0mie	5	Purchased this for my device, it worked as adv...	25-10-2013	409	0	0	0	0	0.0
2	2	1K3	4	it works as expected. I should have sprung for...	23-12-2012	715	0	0	0	0	0.0
3	3	1m2	5	This think has worked out great.Had a diff br...	21-11-2013	382	0	0	0	0	0.0
4	4	2&1/2Men	5	Bought it with Retail Packaging, arrived legit...	13-07-2013	513	0	0	0	0	0.0

Figure 11.1.2 Import dataset

```
[ ]: #cleaning data

[10]: import pandas as pd
import numpy as np

def missing_values_analysis(df):
    na_columns_ = [col for col in df.columns if df[col].isnull().sum() > 0]

    # Filter the DataFrame to include only numeric columns
    numeric_df = df[na_columns_].select_dtypes(include=[np.number])

    # Calculate the ratio of missing values for numeric columns
    ratio_ = (numeric_df.isnull().sum() / df.shape[0] * 100).sort_values(ascending=True)

    missing_df = pd.DataFrame({'Missing Values': numeric_df.isnull().sum(), 'Ratio': ratio_})
    return missing_df

def check_dataframe(df, head=5, tail=5):
    print("SHAPE".center(82, '~'))
    print('Rows: {}'.format(df.shape[0]))
    print('Columns: {}'.format(df.shape[1]))
    print("TYPES".center(82, '~'))
    print(df.dtypes)
    print("", center(82, '~'))
    print(missing_values_analysis(df))
    print('DUPLICATED VALUES'.center(82, '~'))
    print(df.duplicated().sum())
    print("QUANTILES".center(82, '~'))

    # Filter the DataFrame to include only numeric columns
    numeric_df = df.select_dtypes(include=[np.number])

    # Calculate quantiles for numeric columns
    quantiles = numeric_df.quantile([0, 0.05, 0.50, 0.95, 0.99, 1]).T
    print(quantiles)
```

Figure 11.1.3 Cleaning dataset

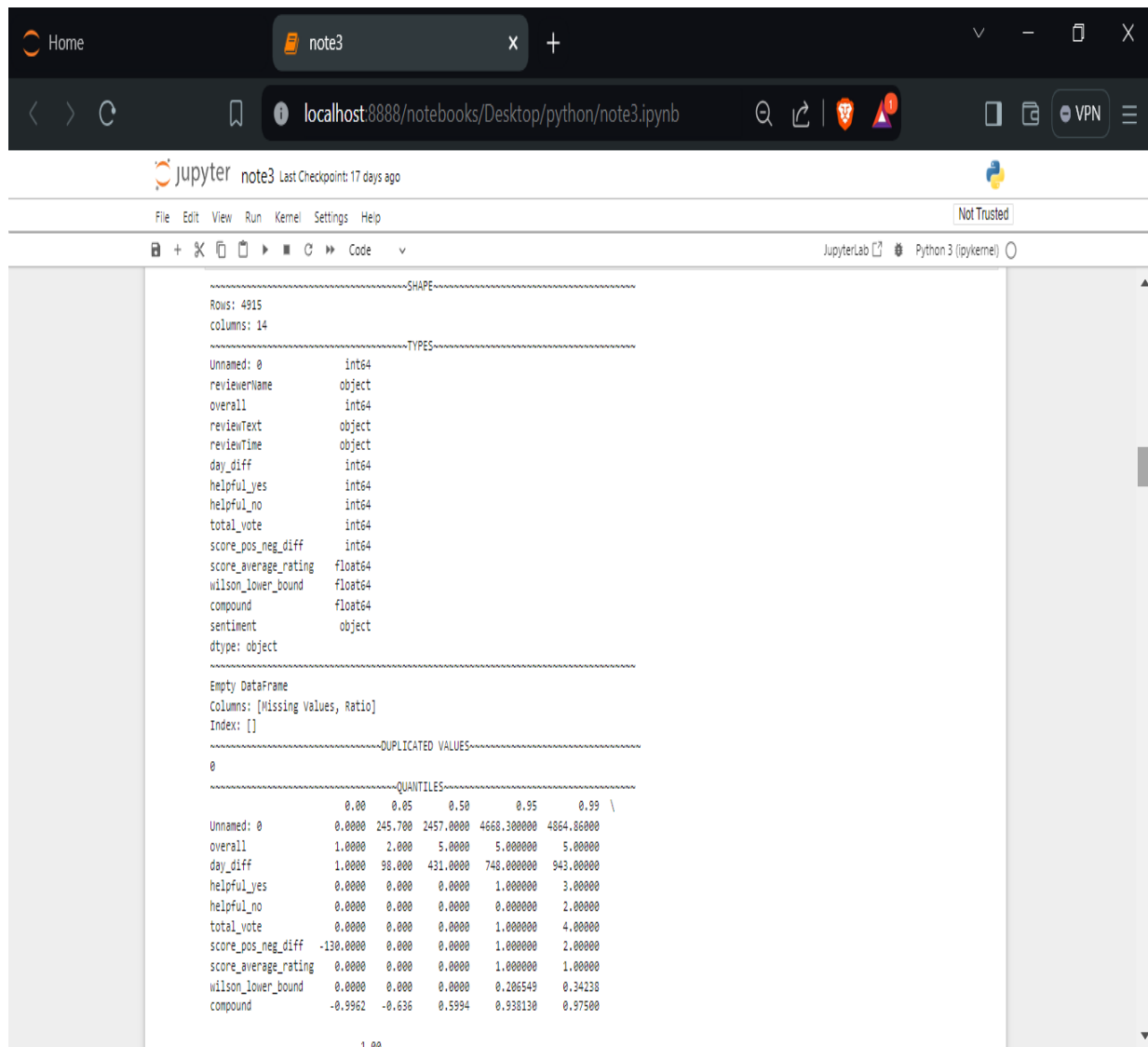


Figure 11.1.4 Cleaned data

The screenshot shows a Jupyter Notebook window titled 'note3' in a web browser. The browser address bar shows 'localhost:8888/notebooks/Desktop/python/note3.ipynb'. The Jupyter interface includes a top bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help' menus. A 'Not Trusted' warning is visible. The notebook contains a code cell with the following Python code:

```
[ ]: #perform sentential analysing

[11]: import pandas as pd
      from nltk.sentiment.vader import SentimentIntensityAnalyzer
      import nltk

      # Download VADER's Lexicon (one-time download)
      nltk.download('vader_lexicon')

      # Load the Amazon review dataset
      df = pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv") # Replace with your actual dataset path

      # Check for and handle missing values in the "reviewText" column
      df["reviewText"].fillna("", inplace=True)

      # Create a SentimentIntensityAnalyzer object
      analyzer = SentimentIntensityAnalyzer()

      # Analyze sentiment for each review
      df["compound"] = df["reviewText"].apply(analyzer.polarity_scores).apply(lambda x: x["compound"])

      # Assign sentiment Labels based on compound scores
      df["sentiment"] = df["compound"].apply(lambda score: "positive" if score >= 0.05 else "negative" if score <= -0.05 else "neutral")

      # Print the results
      print(df[["reviewText", "compound", "sentiment"]].head())

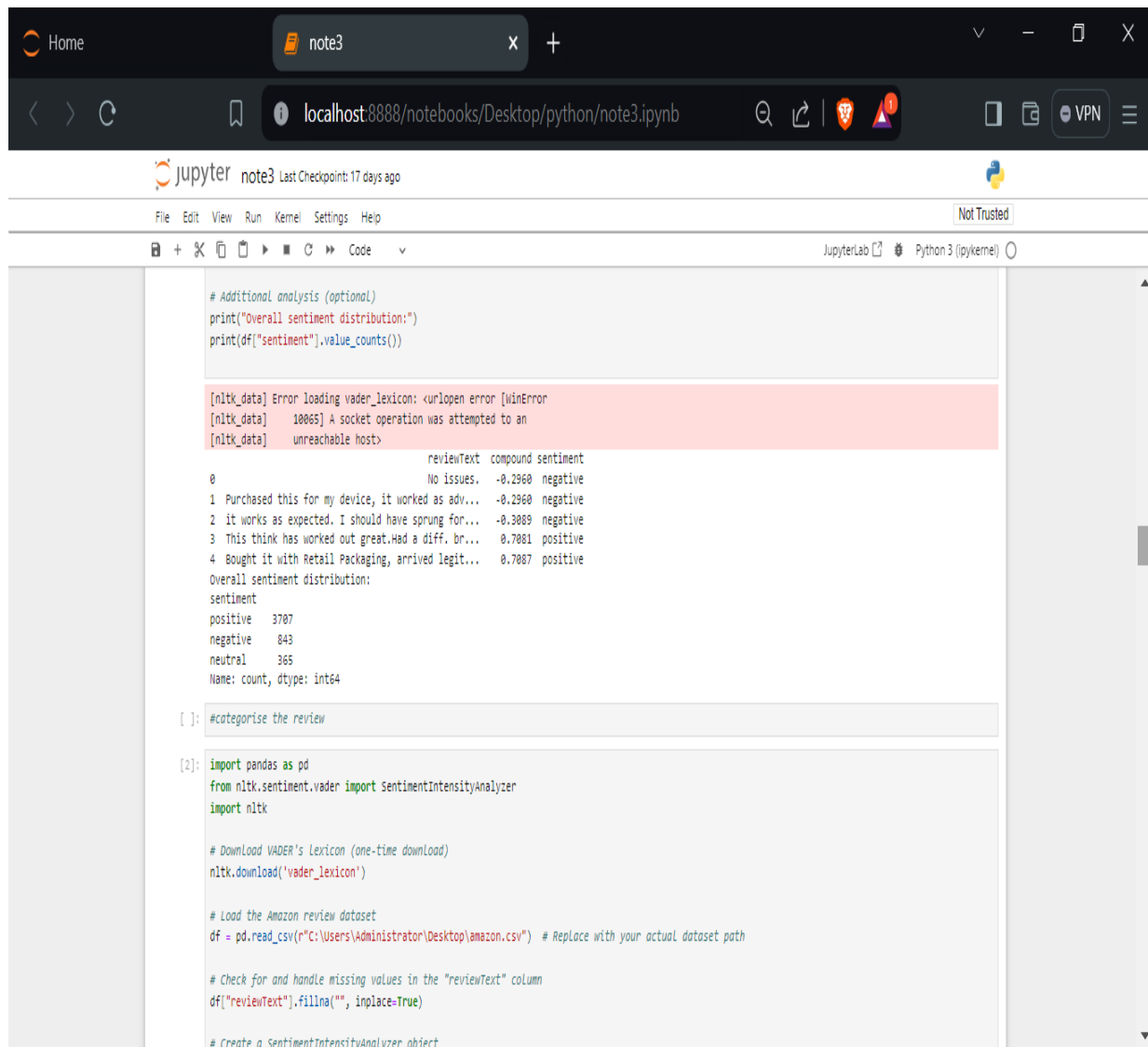
      # Additional analysis (optional)
      print("Overall sentiment distribution:")
      print(df["sentiment"].value_counts())
```

Below the code, an error message is displayed in a red box:

```
[nltk_data] Error loading vader_lexicon: <urlopen error [WinError
[nltk_data] 10065] A socket operation was attempted to an
[nltk_data] unreachable host>
```

At the bottom of the code cell, the column names 'reviewText', 'compound', and 'sentiment' are listed.

Figure 11.1.5 Sentimental analysis

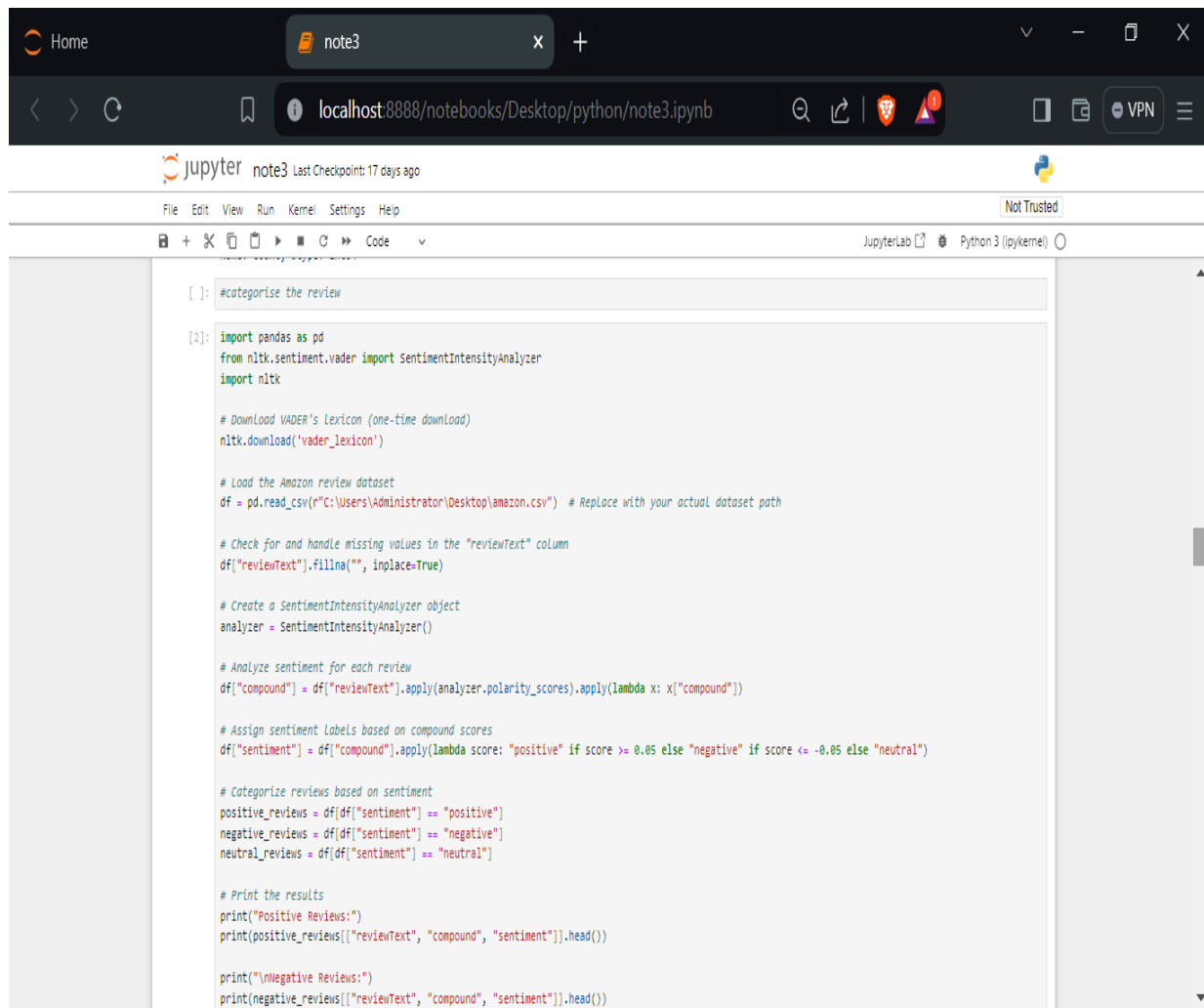


The screenshot shows a JupyterLab interface with a notebook named 'note3'. The browser address bar indicates the URL is 'localhost:8888/notebooks/Desktop/python/note3.ipynb'. The JupyterLab header shows 'note3' and 'Last Checkpoint: 17 days ago'. The notebook interface includes a menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar. The code editor displays the following code:

```
# Additional analysis (optional)
print("Overall sentiment distribution:")
print(df["sentiment"].value_counts())

[nltk_data] Error loading vader_lexicon: 
```

Figure 11.1.6 Sentimental analysis



The screenshot shows a JupyterLab interface with a notebook named 'note3'. The browser address bar indicates the URL is 'localhost:8888/notebooks/Desktop/python/note3.ipynb'. The notebook's last checkpoint was 17 days ago. The code in the notebook is as follows:

```
[ ]: #categorise the review

[2]: import pandas as pd
      from nltk.sentiment.vader import SentimentIntensityAnalyzer
      import nltk

      # Download VADER's Lexicon (one-time download)
      nltk.download('vader_lexicon')

      # Load the Amazon review dataset
      df = pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv") # Replace with your actual dataset path

      # Check for and handle missing values in the "reviewText" column
      df["reviewText"].fillna("", inplace=True)

      # Create a SentimentIntensityAnalyzer object
      analyzer = SentimentIntensityAnalyzer()

      # Analyze sentiment for each review
      df["compound"] = df["reviewText"].apply(analyzer.polarity_scores).apply(lambda x: x["compound"])

      # Assign sentiment labels based on compound scores
      df["sentiment"] = df["compound"].apply(lambda score: "positive" if score >= 0.05 else "negative" if score <= -0.05 else "neutral")

      # Categorize reviews based on sentiment
      positive_reviews = df[df["sentiment"] == "positive"]
      negative_reviews = df[df["sentiment"] == "negative"]
      neutral_reviews = df[df["sentiment"] == "neutral"]

      # Print the results
      print("Positive Reviews:")
      print(positive_reviews[["reviewText", "compound", "sentiment"]].head())

      print("\nNegative Reviews:")
      print(negative_reviews[["reviewText", "compound", "sentiment"]].head())
```

Figure 11.1.7 Categorize review

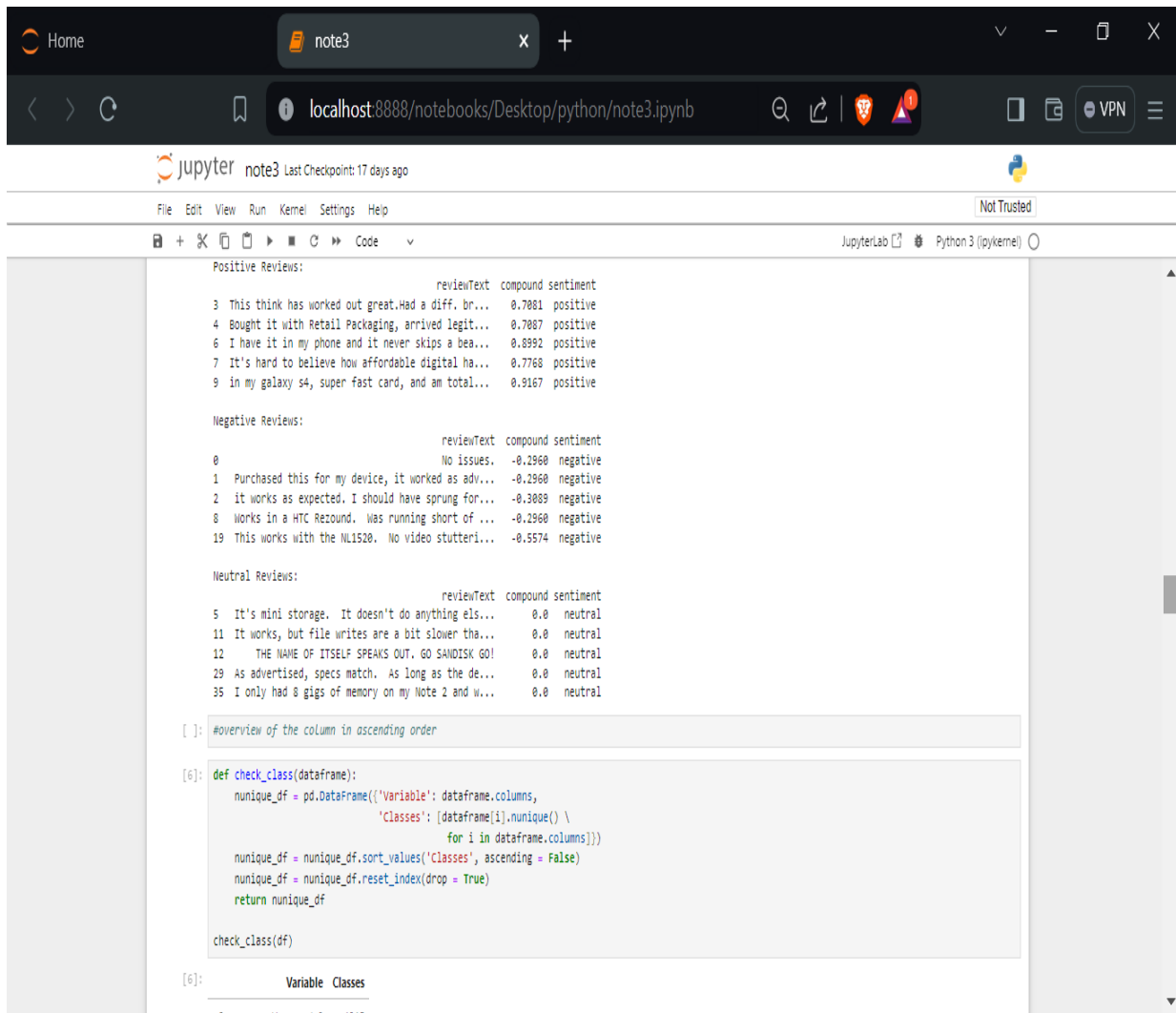


Figure 11.1.8 Categorized review

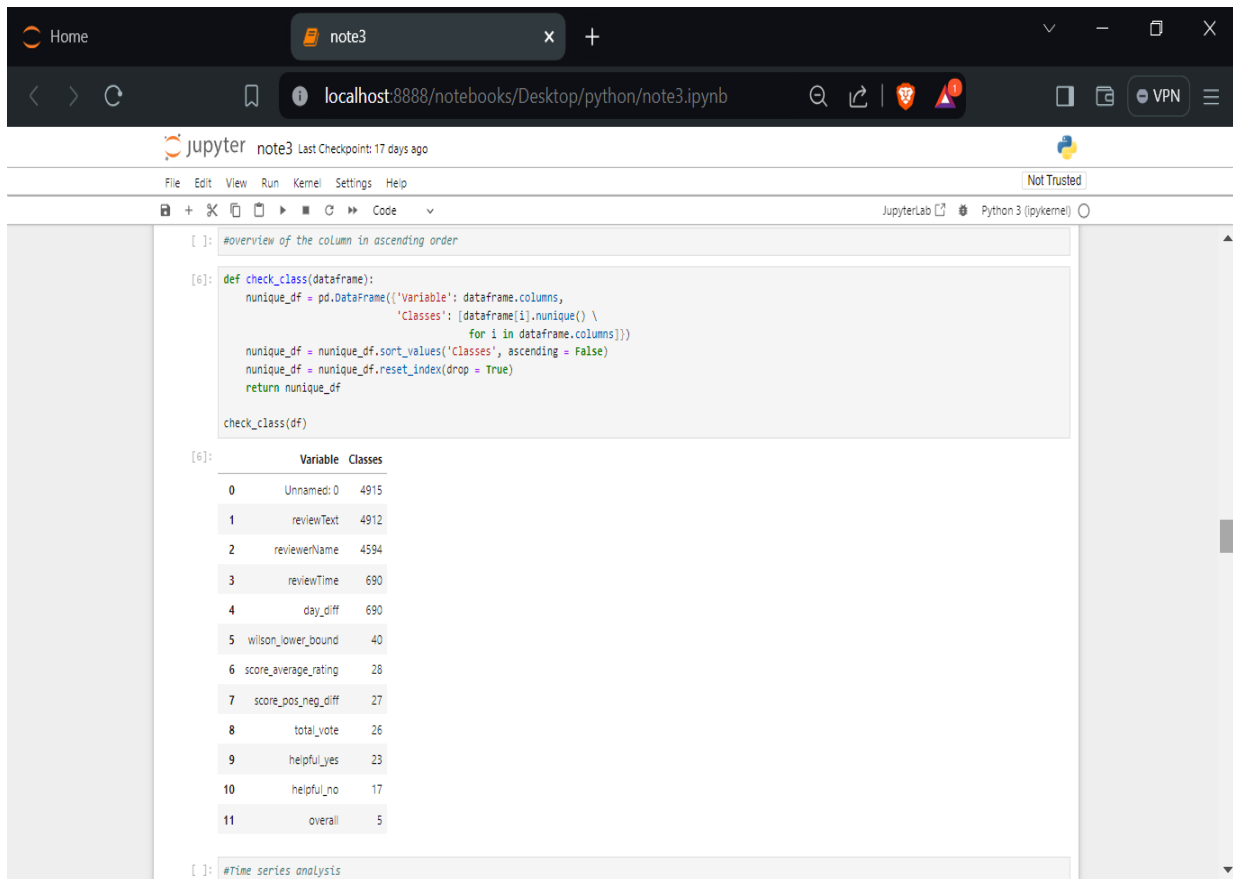


Figure 11.1. 9 Arranging Reviews

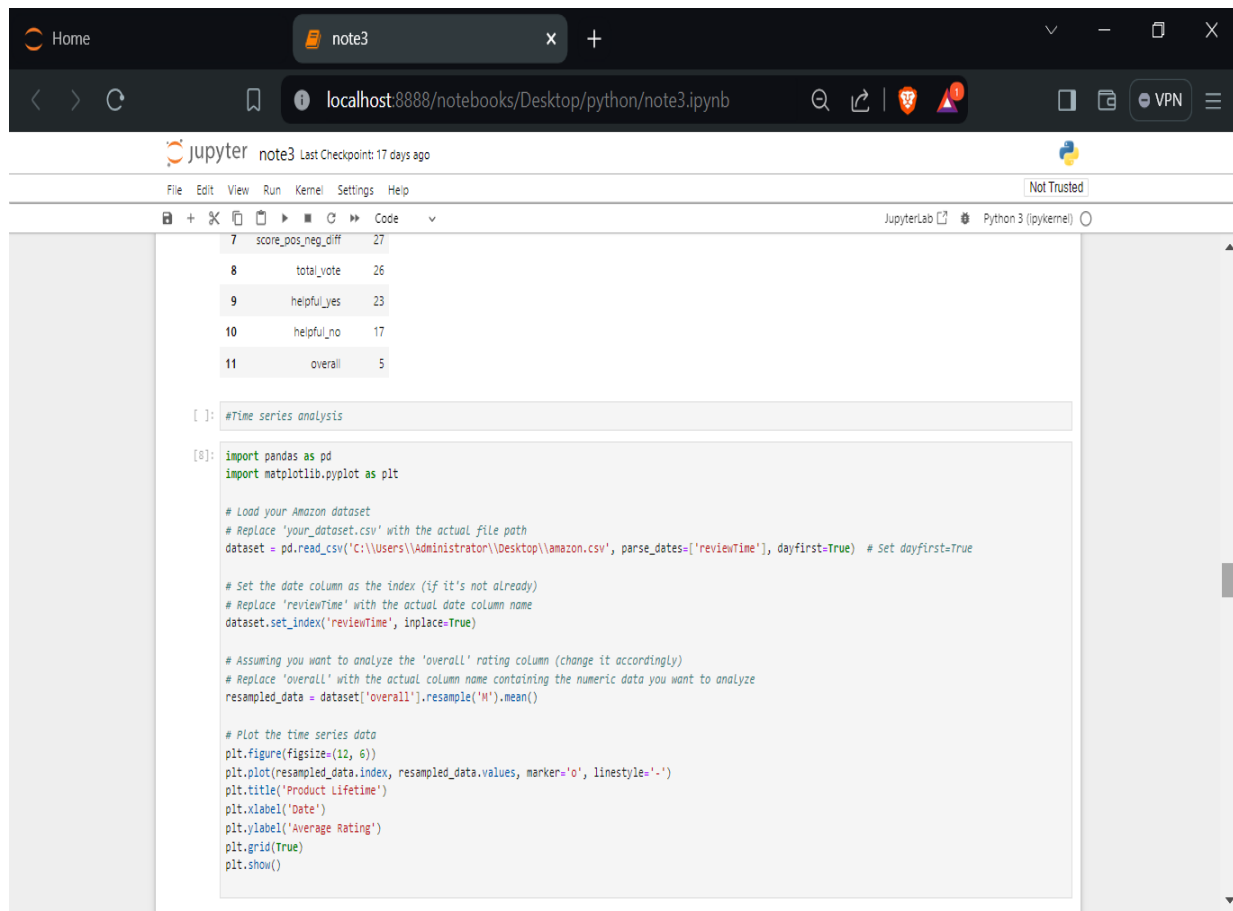


Figure 11.10 Timeseries Analysis

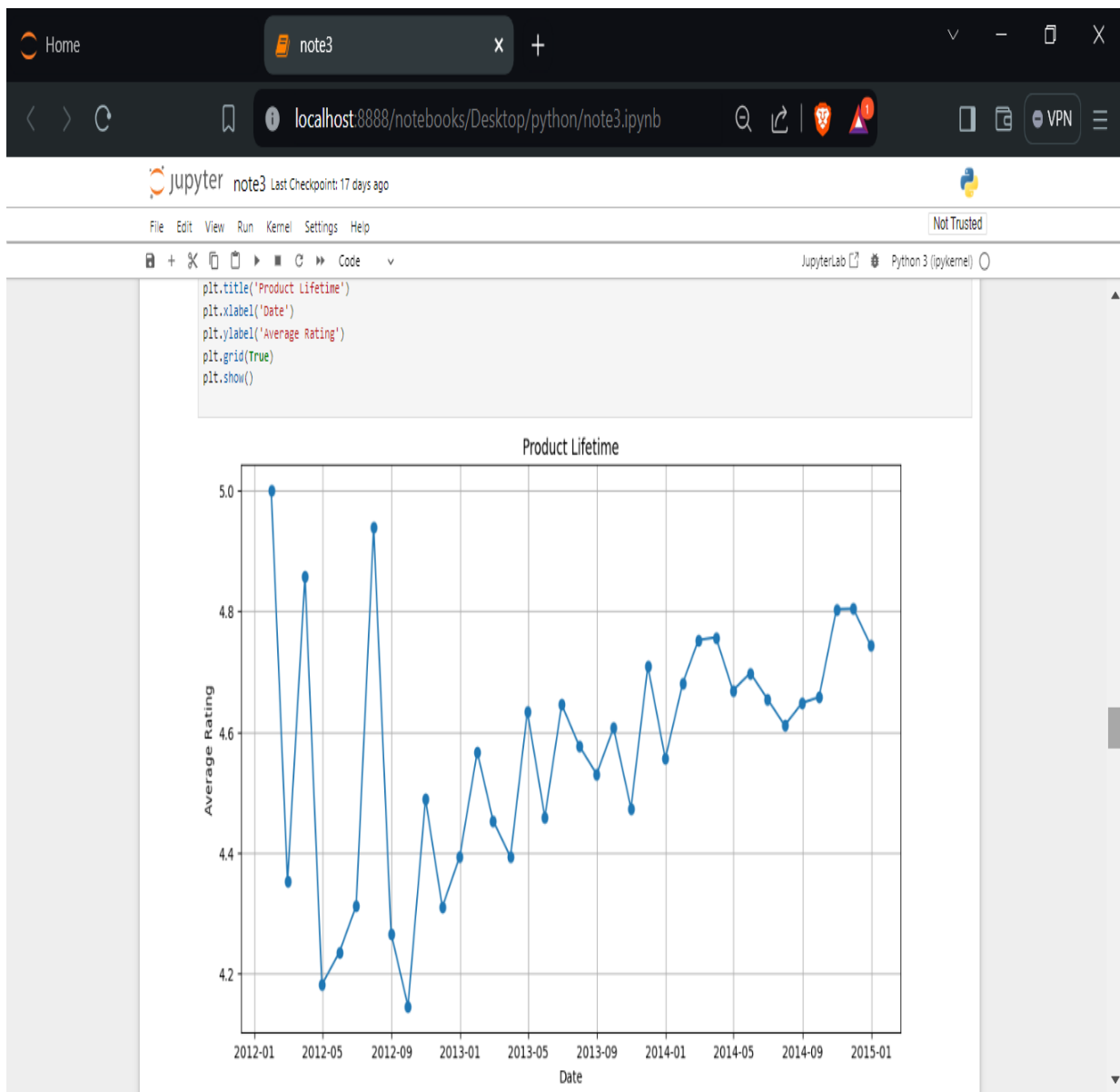
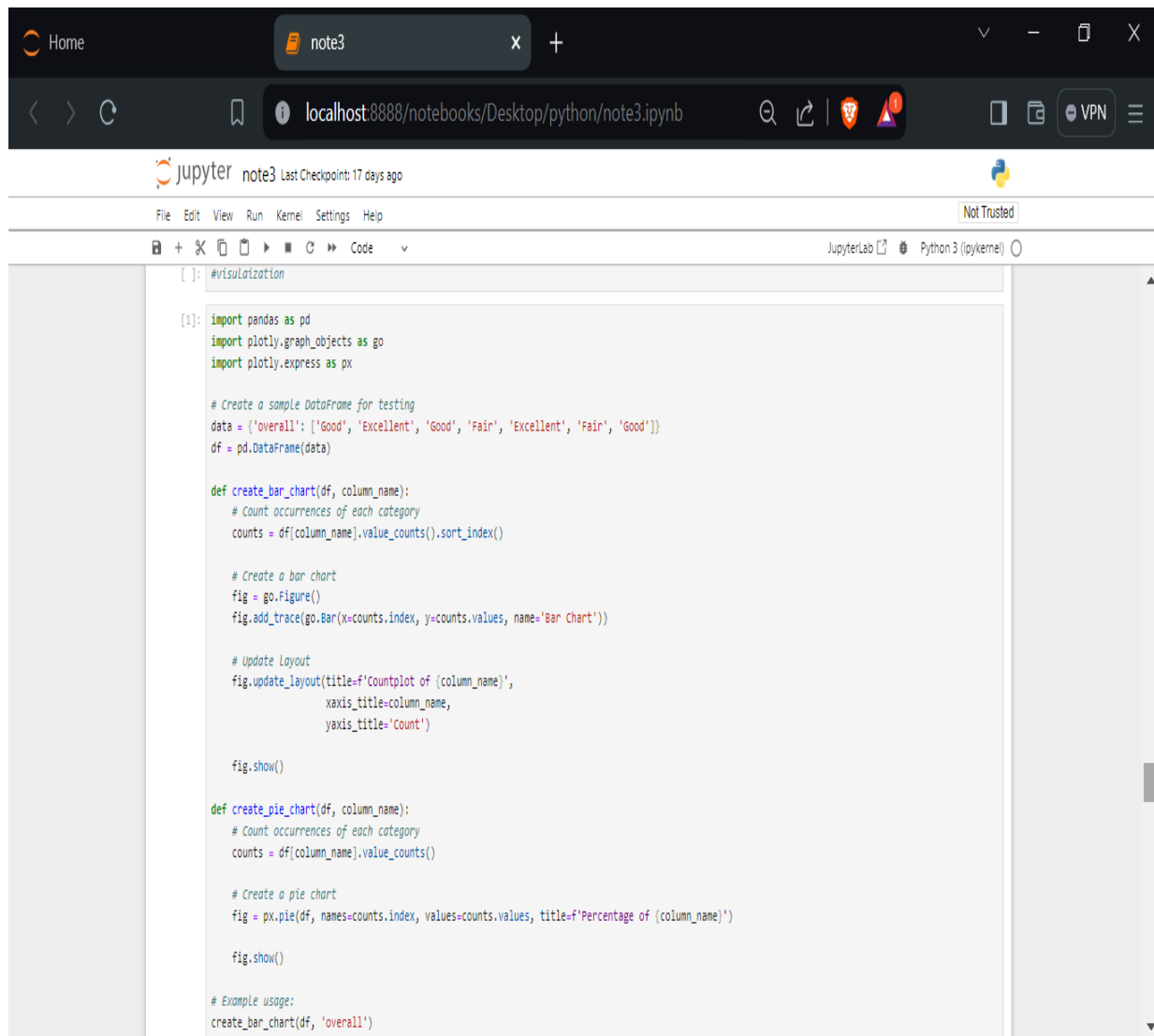


Figure 11.11 Time series Analysis



```
[ ]: #visulaization

[1]: import pandas as pd
import plotly.graph_objects as go
import plotly.express as px

# Create a sample DataFrame for testing
data = {'overall': ['Good', 'Excellent', 'Good', 'Fair', 'Excellent', 'Fair', 'Good']}
df = pd.DataFrame(data)

def create_bar_chart(df, column_name):
    # Count occurrences of each category
    counts = df[column_name].value_counts().sort_index()

    # Create a bar chart
    fig = go.Figure()
    fig.add_trace(go.Bar(x=counts.index, y=counts.values, name='Bar Chart'))

    # Update layout
    fig.update_layout(title=f'Countplot of {column_name}',
                      xaxis_title=column_name,
                      yaxis_title='Count')

    fig.show()

def create_pie_chart(df, column_name):
    # Count occurrences of each category
    counts = df[column_name].value_counts()

    # Create a pie chart
    fig = px.pie(df, names=counts.index, values=counts.values, title=f'Percentage of {column_name}')

    fig.show()

# Example usage:
create_bar_chart(df, 'overall')
create_pie_chart(df, 'overall')
```

Figure 11.1.13 Visualization

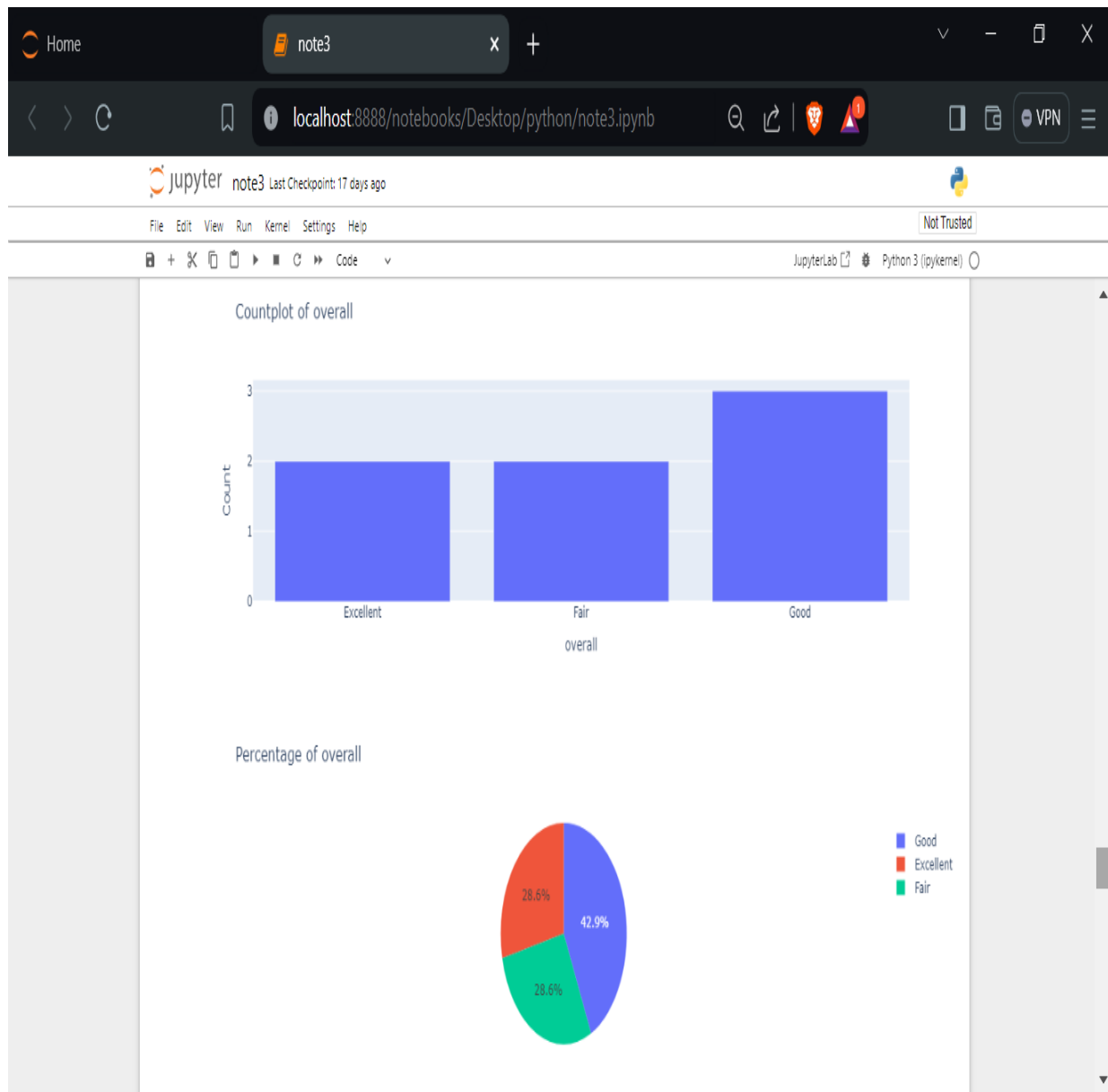


Figure 11.1.14 Visualization

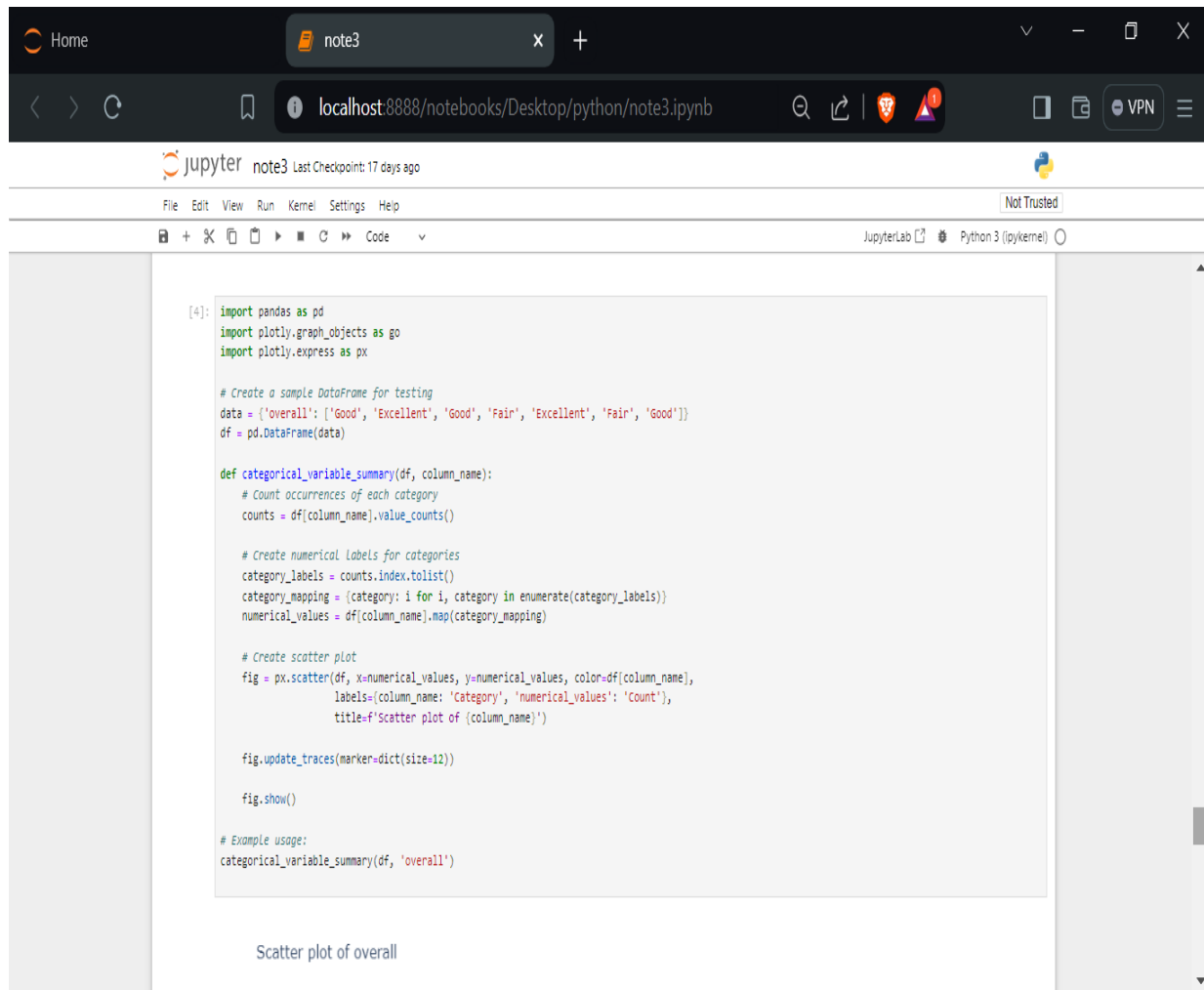


Figure 11.15 Scatter plot

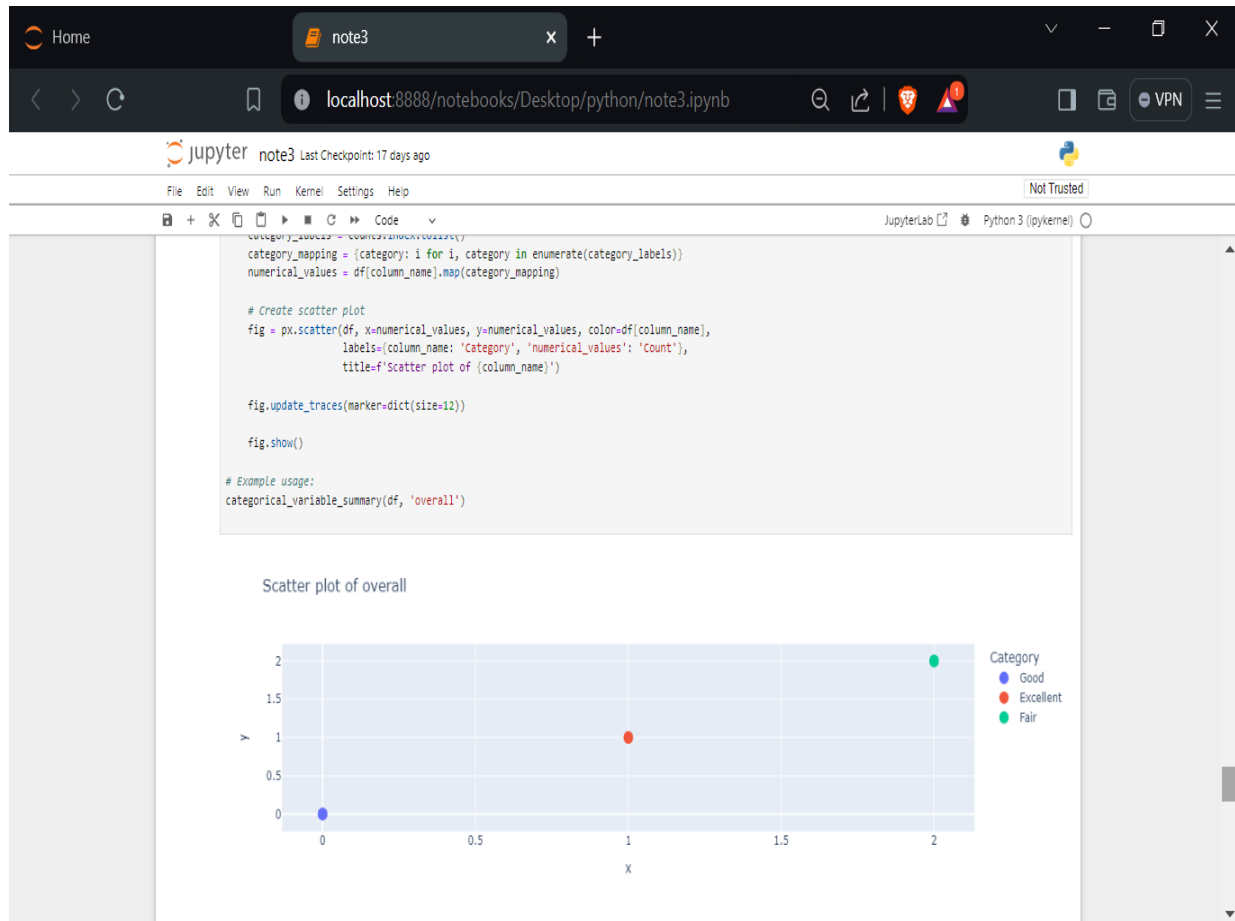


Figure 11.1.16 Scatter plot

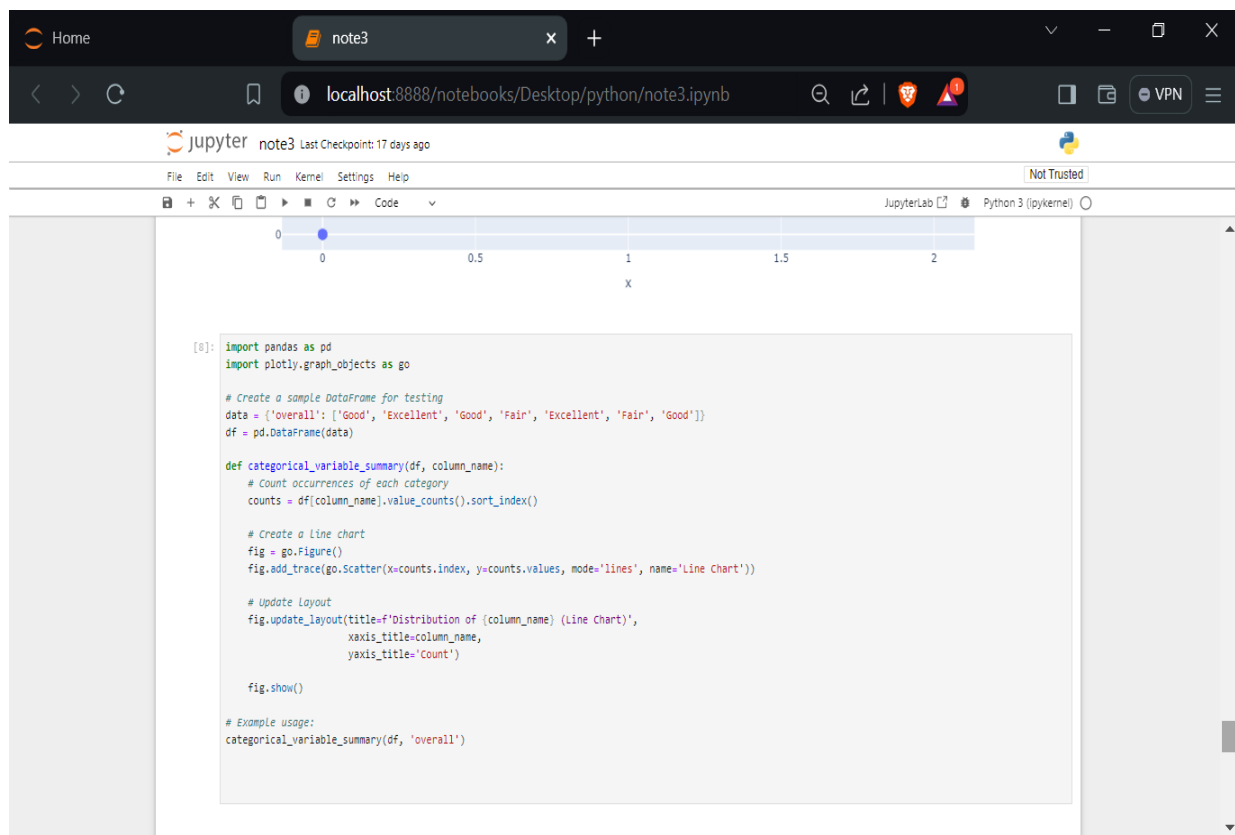


Figure 11.17 Line chart

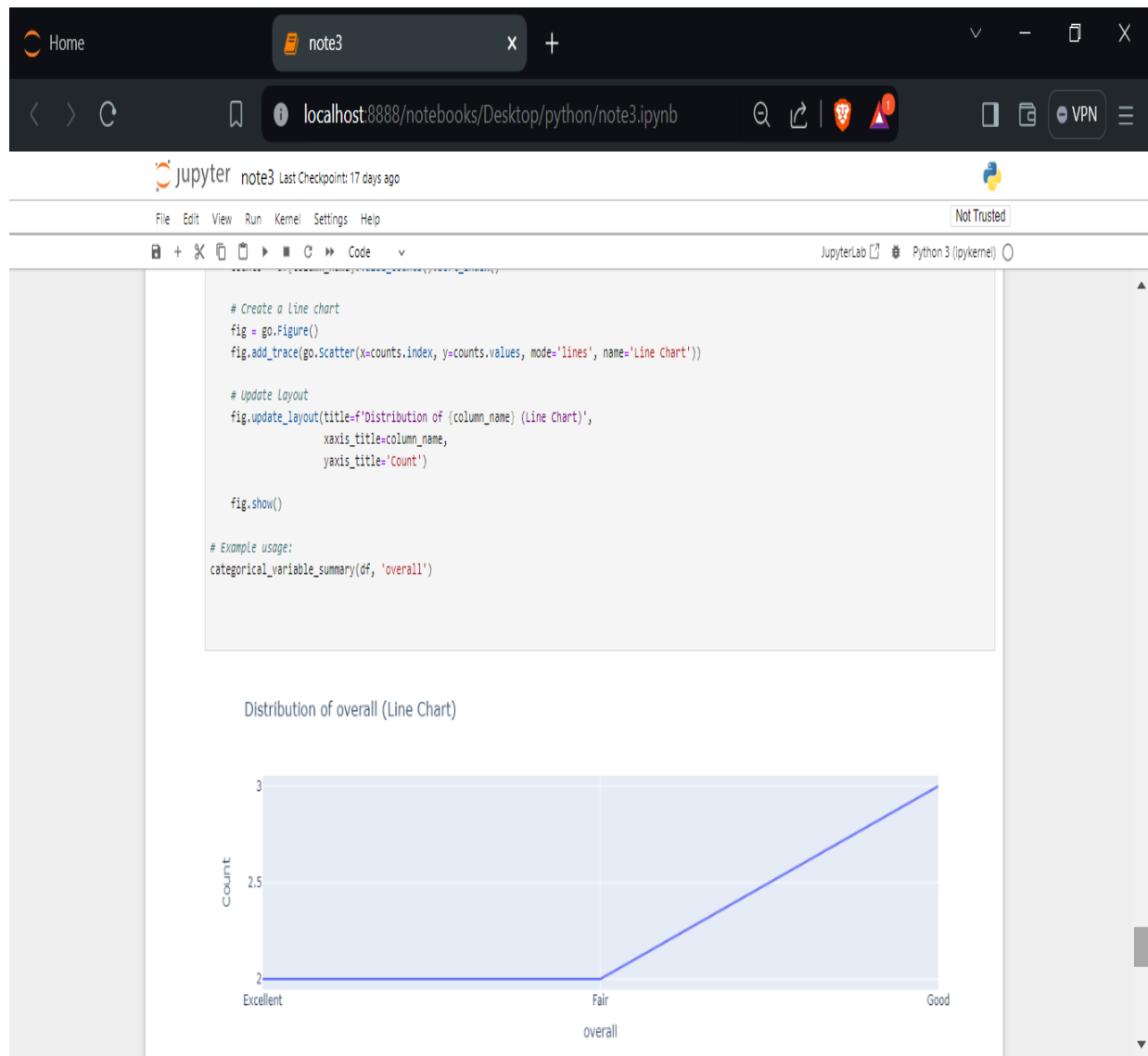


Figure 11.1.18 Line chart

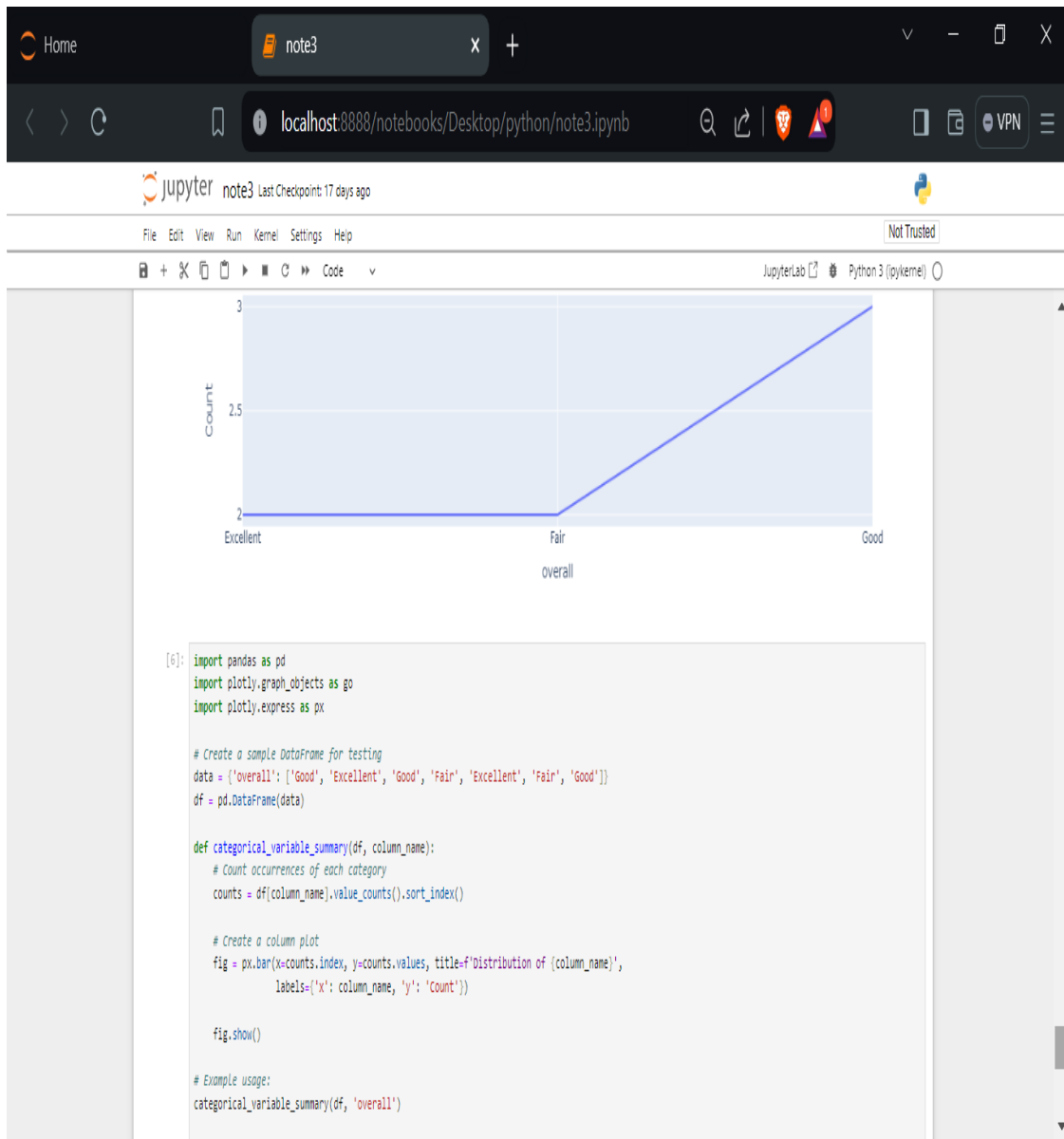


Figure 11.1.19 Bar chart

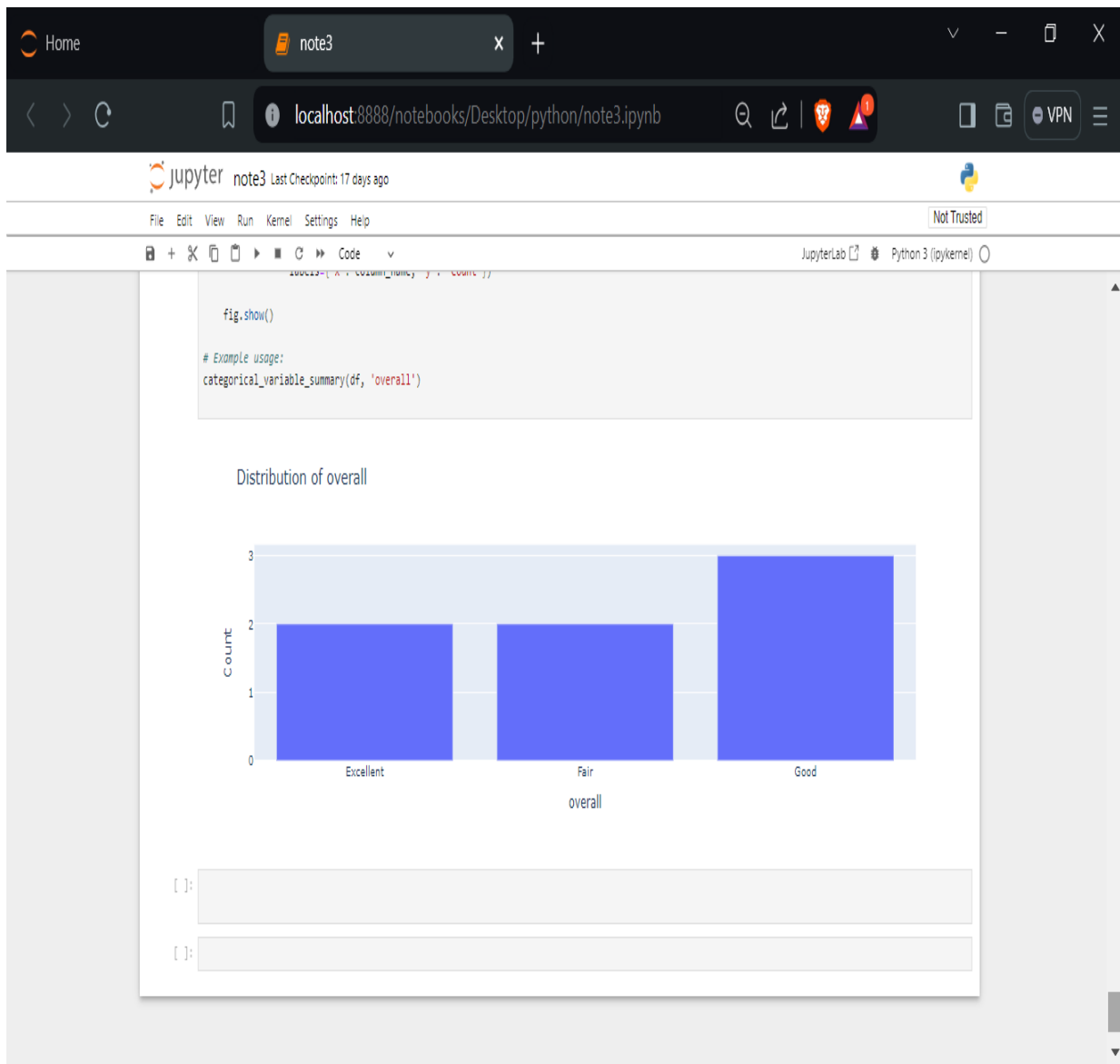


Figure 11.1.20 Bar chart

11.2 SOURCE CODING

Import Libraries

```
import numpy as np

import pandas as pd

import nltk

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import re

from textblob import TextBlob

from wordcloud import WordCloud

import seaborn as sns

import matplotlib.pyplot as plt

import cufflinks as cf

%matplotlib inline

from plotly.offline import init_notebook_mode, iplot

init_notebook_mode(connected = True)

cf.go_offline();

import plotly.graph_objs as go

from plotly.subplots import make_subplots

import warnings

warnings.filterwarnings("ignore")
```

Cleaning Data

```
import pandas as pd

import numpy as np

def missing_values_analysis(df):

    na_columns_ = [col for col in df.columns if df[col].isnull().sum() > 0]

    # Filter the DataFrame to include only numeric columns

    numeric_df = df[na_columns_].select_dtypes(include=[np.number])

    # Calculate the ratio of missing values for numeric columns

    ratio_ = (numeric_df.isnull().sum() / df.shape[0] * 100).sort_values(ascending=True)

    missing_df = pd.DataFrame({'Missing Values': numeric_df.isnull().sum(), 'Ratio': ratio_})

    return missing_df

def check_dataframe(df, head=5, tail=5):

    print("SHAPE".center(82, '~'))

    print('Rows: { }'.format(df.shape[0]))

    print('columns: { }'.format(df.shape[1]))

    print("TYPES".center(82, '~'))

    print(df.dtypes)

    print(""".center(82, '~'))

    print(missing_values_analysis(df))
```

Sentimental analysis code

```
import pandas as pd

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk

# Download VADER's lexicon (one-time download)

nltk.download('vader_lexicon')

# Load the Amazon review dataset

df = pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv") # Replace with your actual
dataset path

# Check for and handle missing values in the "reviewText" column

df["reviewText"].fillna("", inplace=True)

# Create a SentimentIntensityAnalyzer object

analyzer = SentimentIntensityAnalyzer()

# Analyze sentiment for each review

df["compound"] = df["reviewText"].apply(analyzer.polarity_scores).apply(lambda x:
x["compound"])

# Assign sentiment labels based on compound scores

df["sentiment"] = df["compound"].apply(lambda score: "positive" if score >= 0.05 else
"negative" if score <= -0.05 else "neutral")

# Print the results print(df[["reviewText", "compound", "sentiment"]].h
```

Categorized the Review

```
import pandas as pd

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk

# Download VADER's lexicon (one-time download)

nltk.download('vader_lexicon')

# Load the Amazon review dataset

df = pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv") # Replace with
your actual dataset path

# Check for and handle missing values in the "reviewText" column

df["reviewText"].fillna("", inplace=True)

# Create a SentimentIntensityAnalyzer object

analyzer = SentimentIntensityAnalyzer()

# Analyze sentiment for each review

df["compound"] = df["reviewText"].apply(analyzer.polarity_scores).apply(lambda
x: x["compound"])

# Assign sentiment labels based on compound scores

df["sentiment"] = df["compound"].apply(lambda score: "positive" if score >= 0.05
else "negative" if score <= -0.05 else "neutral")

# Print the results print(df[["reviewText", "compound"]])
```


Visualization

```
import pandas as pd

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk

# Download VADER's lexicon (one-time download)

nltk.download('vader_lexicon')

# Load the Amazon review dataset

df = pd.read_csv(r"C:\Users\Administrator\Desktop\amazon.csv") # Replace with
your actual dataset path

# Check for and handle missing values in the "reviewText" column

df["reviewText"].fillna("", inplace=True)

# Create a SentimentIntensityAnalyzer object

analyzer = SentimentIntensityAnalyzer()

# Analyze sentiment for each review

df["compound"] = df["reviewText"].apply(analyzer.polarity_scores).apply(lambda
x: x["compound"])

# Assign sentiment labels based on compound scores

df["sentiment"] = df["compound"].apply(lambda score: "positive" if score >= 0.05
else "negative" if score <= -0.05 else "neutral")
```

Timeseries Analysis

```
import pandas as pd

import matplotlib.pyplot as plt

# Load your Amazon dataset

# Replace 'your_dataset.csv' with the actual file path

dataset = pd.read_csv('C:\\Users\\Administrator\\Desktop\\amazon.csv',
parse_dates=['reviewTime'], dayfirst=True) # Set dayfirst=True


# Set the date column as the index (if it's not already)

# Replace 'reviewTime' with the actual date column name

dataset.set_index('reviewTime', inplace=True)


# Assuming you want to analyze the 'overall' rating column (change it accordingly)

# Replace 'overall' with the actual column name containing the numeric data you
want to analyze

resampled_data = dataset['overall'].resample('M').mean()

# Plot the time series data

plt.figure(figsize=(12, 6))

plt.plot(resampled_data.index, resampled_data.values, marker='o', linestyle='-')

plt.title('Product Lifetime') plt.xlabel('Date')
```