AN INTERNSHIP REPORT

on

Data Analytics Using Python

Submitted in partial fulfilment of the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BANDARU DEVA SRIRAMA SAI GANESH

(21021A0548)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA-533003, ANDHRA PRADESH, INDIA

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA-533003, ANDHRA PRADESH, INDIA

CERTIFICATE

This is to certify that this project report entitled "**DATA ANALYTICS USING PYTHON**" is a bonafide record of the work being submitted by **BANDARU DEVA SRIRAMA SAI GANESH** bearing the roll number **21021A0548**, in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** to the **UCEK(A)**, **JNTUK**, Kakinada, Andhra Pradesh, India. It has been found satisfactory and hereby approved for submission.

Signature of Head of the Department

Dr. N. Rama Krishnaiah

Professor & HOD

Department of CSE

UCEK(A)

JNTU KAKINADA

ACKNOWLEDGEMENT

It is with immense pleasure that I take this opportunity to express my heartfelt gratitude to everyone who supported me in successfully completing my internship at Main Flow Services and Technologies. I extend my sincere thanks to Dr. Ramakrishnaiah, Professor and Head of the Department of Computer Science and Engineering, for his guidance and encouragement. I am also deeply grateful to our Principal, Dr. Mohan Rao, University College of Engineering Kakinada, for his constant support and motivation throughout this internship.

I would like to extend my profound gratitude to Dr. Ravindranath and Dr. Deekshithulu for their valuable encouragement and support during this journey. My heartfelt appreciation also goes to my family, friends, and companions, whose unwavering support was instrumental in the completion of this report.

This internship has provided me with invaluable practical exposure and an opportunity to bridge the gap between theoretical knowledge and real-world applications. It is a privilege and an honor to present this report, and I hope it serves as a meaningful contribution to bridging the divide between academic concepts and practical implementation.

INTERNSHIP OFFER LETTER

Main Flow Services and Technologies

15 April 2024

Dear, DEVA SRIRAMA SAI GANESH ID –11199

Congratulations on being selected for the "**Data Analysis With Python Internship"** internship position with **"Main Flow"**. We at **"Main Flow"** are excited that you will join our team.

The internship will be 8 weeks, from 15 April 2024 to 15 June 2024.
The internship is an educational opportunity for you hence the primary focus is on learning and developing new skills and gaining hands-on knowledge. We believe that you will perform all your tasks/projects.

As an intern, we expect you to perform all assigned tasks to the best of your ability and follow any lawful and reasonable instructions.

We are confident that this internship will be a valuable experience for you, We lookforward to working with you and helping you achieve your career goals.

Sincerely,

Director - Gaurav kumar

MSME REGISTERED UP-29-XX

+919389641586
+919773699074

www.mainflow.in
career.mainflow@gmail.com

INTERNSHIP CERTIFICATE

ABSTRACT

This internship at Main Flow Services and Technologies focused on applying data analytics techniques using Python to derive insights from complex datasets. The first project, **Sentiment Analysis**, involved preprocessing textual data, implementing machine learning models such as Logistic Regression, Bernoulli Naive Bayes, and Linear SVC, and using VADER for sentiment classification. The project emphasized natural language processing (NLP) techniques, model evaluation with accuracy, confusion matrix, and ROC curves, and provided valuable insights into textual sentiment.

The second project, **Time Series Analysis and Forecasting**, analyzed trends in Disney+ content over time, using statistical methods and ARIMA modeling to predict future patterns of titles added. The project demonstrated proficiency in time series decomposition, forecasting, and visualizing content release trends, offering actionable insights for content strategy.

Throughout the internship, Python libraries such as pandas, NumPy, matplotlib, and statsmodels were extensively used for data manipulation, visualization, and predictive modeling. These projects enhanced technical skills in data preprocessing, model development, and forecasting, while showcasing the power of Python in solving real-world data analytics challenges.

Index Terms: Data Analytics, Python, Sentiment Analysis, Time Series Forecasting, Logistic Regression, Bernoulli Naive Bayes, Linear SVC, ARIMA, Disney+ Dataset, VADER, Data Preprocessing, Model Evaluation.

Table of Contents

CHAPTER I

INTRODUCTION

INTRODUCTION

Main Flow Services and Technologies is a prominent player in the data analytics and technology sector, offering solutions for businesses to harness the power of data. The company works with a variety of industries, helping clients make informed decisions based on data analysis, machine learning, and predictive analytics.

As an intern in the Data Analytics department, my primary goal was to contribute to the company's data-driven initiatives by analyzing data using Python and its associated libraries. The objective was to develop skills in data analytics, with a focus on real-world data analysis, sentiment analysis, and time series forecasting. Through hands-on experience, I aimed to improve my technical capabilities and contribute meaningfully to the team's efforts.

During the internship, I was tasked with analyzing two key datasets: one focused on text data (sentiment analysis) and the other on time-series data (forecasting trends). The challenge was to preprocess these datasets, apply the appropriate statistical methods, and build predictive models that could provide actionable insights for the business.

This report outlines my work during the internship, focusing on two main projects:

1. **Mini Project - Sentiment Analysis:**

   Analyzing customer feedback and social media data to classify sentiments and derive insights about customer opinions.

2. **Major Project - Time Series Forecasting:**

   Analyzing Disney+ title data to understand trends over time and predict future content additions.

To accomplish these projects, I employed several data analytics techniques, including natural language processing (NLP) for sentiment analysis and statistical time-series forecasting models like ARIMA. I

used Python libraries such as pandas, NumPy, statsmodels, and matplotlib for data preprocessing, analysis, and visualization.

This report details the steps taken to achieve these objectives and provides insights into the methods used to solve the problems at hand.

CHAPTER II

LITERATURE SURVEY

LITERATURE SURVEY

The field of **Data Analytics** has evolved rapidly with the increasing availability of big data and the development of advanced analytical tools. In the context of this internship, two key areas were explored: **Sentiment Analysis** and **Time Series Forecasting**. Both these domains have been extensively studied, with various methods and algorithms proposed over the years. Below is a review of the relevant literature for these two fields.

1. Sentiment Analysis

Sentiment analysis, also known as opinion mining, refers to the use of natural language processing (NLP) techniques to analyze and understand the sentiment or opinion expressed in text. It has become increasingly important in fields such as marketing, social media analysis, and customer feedback evaluation.

Several approaches have been proposed for sentiment analysis, with **supervised learning** methods being the most common. **Naive Bayes** and **Support Vector Machines (SVMs)** are frequently used algorithms for text classification tasks in sentiment analysis due to their simplicity and effectiveness (Pang et al., 2002). These methods rely on labeled datasets, where the sentiment of each text sample is pre-annotated, allowing the model to learn patterns in the data.

More recent research has explored the use of **deep learning** techniques such as **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks (Kim, 2014). These models are capable of learning complex patterns in sequential data, such as text, and have been shown to outperform traditional machine learning models on sentiment classification tasks.

In addition, techniques like **VADER (Valence Aware Dictionary and sEntiment Reasoner)** have gained popularity for their simplicity and effectiveness in analyzing social media text, where

informal language, abbreviations, and emojis are common (Hutto and Gilbert, 2014). VADER uses a lexicon of sentiment-related words and applies heuristics to assign sentiment scores to text.

2. Time Series Forecasting

Time series forecasting is a critical area of data analysis that involves predicting future values based on historical data. It has widespread applications in business, economics, and various fields that require predicting trends, such as stock market prices and demand forecasting.

A traditional approach to time series forecasting is the **Autoregressive Integrated Moving Average (ARIMA)** model, which has been widely used for its simplicity and effectiveness in modeling univariate time series data (Box and Jenkins, 1976). ARIMA combines autoregression (AR), moving averages (MA), and differencing (I) to model and forecast data with trends and seasonality.

Recent developments have incorporated more complex methods, such as **Seasonal ARIMA (SARIMA)**, which handles seasonality in time series data (Hyndman et al., 2008). Another approach is **Exponential Smoothing State Space Models (ETS)**, which applies smoothing techniques to forecast future values based on weighted averages of past observations (Hyndman and Athanasopoulos, 2018).

Another important method in time series forecasting is **Prophet**, developed by Facebook. Prophet is designed to handle missing data and large seasonal effects, making it well-suited for time series data with complex seasonal patterns and holidays (Taylor and Letham, 2017). This method has become popular due to its ease of use and flexibility.

In addition, machine learning techniques like **Random Forests** and **Gradient Boosting Machines (GBM)** have been explored for time series forecasting tasks, especially when dealing with large datasets with multiple features (Smyl, 2020).

3. Python Libraries for Data Analysis

Python has become one of the most popular languages for data analytics due to its simplicity and the powerful libraries it offers for data manipulation, statistical analysis, and machine learning. **Pandas**, **NumPy**, and **Matplotlib** are among the most widely used libraries for data preprocessing and visualization. Pandas provides efficient data structures like DataFrames, which facilitate data manipulation and cleaning, while NumPy is essential for numerical computing (McKinney, 2010; Van Der Walt et al., 2011).

For sentiment analysis and text mining, libraries such as **NLTK (Natural Language Toolkit)** and **spaCy** provide tools for tokenization, part-of-speech tagging, and named entity recognition (Bird et al., 2009). Additionally, **scikit-learn** is a widely used library for implementing machine learning models, providing functions for classification, regression, and clustering tasks (Pedregosa et al., 2011).

For time series forecasting, libraries like **Statsmodels** and **Prophet** are often used. **Statsmodels** provides statistical models and methods for time series analysis, including ARIMA and SARIMA models, while **Prophet** provides an easy-to-use interface for forecasting complex time series data (Seabold and Perktold, 2010; Taylor and Letham, 2017).

CHAPTER III

OBJECTIVES

OBJECTIVES

The primary objective of this internship at Main Flow Services and Technologies was to apply Data Analysis techniques using Python to address real-world business challenges. The internship provided an opportunity to work with large datasets, perform various forms of data analysis, and implement predictive models to extract valuable insights.

The key objectives were as follows:

1. **Understanding Business Context and Data**

   Gain insights into the operations of Main Flow Services and Technologies, and explore real-world datasets to identify relevant data for decision-making.

2. **Learning and Applying Data Analysis Techniques**

   Familiarize with data preprocessing, sentiment analysis, and time series analysis using Python.

3. **Exploring Python for Data Analytics**

   Utilize libraries such as pandas, matplotlib, seaborn, and statsmodels for data cleaning, manipulation, and visualization

4. **Implement Sentiment Analysis**

   Apply NLP techniques like VADER and machine learning algorithms (Logistic Regression, Naive Bayes, Linear SVC) for sentiment classification on text data.

5. **Conduct Time Series Analysis**

   Perform exploratory analysis on time-based datasets and forecast trends using models like ARIMA.

6. **Generate Actionable Insights**

   Extract valuable insights from data analysis and effectively communicate findings through visualizations and reports.

CHAPTER IV

OVERVIEW OF SENTIMENT ANALYSIS AND TIME SERIES METHODS

4.1 SENTIMENT ANALYSIS OVERVIEW

Sentiment analysis, also known as opinion mining or emotion AI, is the process of classifying a block of text as positive, negative, or neutral. It uses Natural Language Processing (NLP), text analysis, and computational linguistics to systematically identify and quantify emotional states and subjective information. The goal of sentiment analysis is to analyze people's opinions and feedback in a way that provides insights to businesses, helping them improve customer satisfaction, enhance product offerings, and make informed decisions.

Importance of Sentiment Analysis

Sentiment analysis plays a critical role in making sense of unstructured data, which accounts for 80% of the world’s data. Emails, texts, documents, and social media posts are examples of unstructured data that require efficient processing.

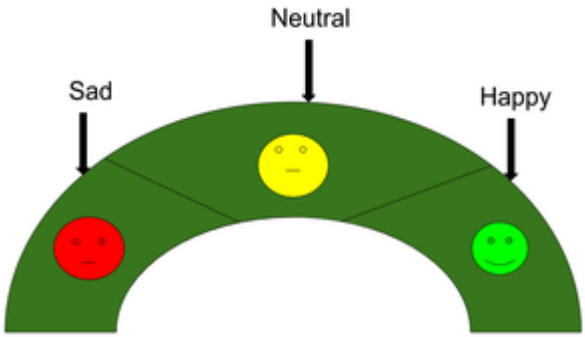
The applications of sentiment analysis are diverse, including:

1. **Customer Feedback Analysis**: Businesses can analyze reviews, surveys, and social media comments to identify customer sentiments and improve products and services.
   Example: Sentiment analysis of app reviews to improve customer service.
2. **Brand Reputation Management**: Monitoring sentiments on online platforms helps companies track public perception and respond proactively to maintain their reputation.
   Example: Monitoring customer sentiment during a new product launch.
3. **Product Development and Innovation**: Insights into customer opinions enable companies to refine existing features or introduce new ones aligned with customer preferences.
4. **Competitor Analysis**: Sentiment analysis helps compare public sentiment toward a company's products with those of competitors, guiding strategic decisions.

5. **Marketing Campaign Effectiveness**: Analyzing the sentiment of customer responses to marketing campaigns helps evaluate their success and identify areas for improvement.

   Example: Measuring consumer sentiment toward a promotional campaign.

Sentiment analysis focuses not only on the polarity of text (positive, negative, or neutral) but also on emotions such as happiness, sadness, anger, etc. It utilizes various Natural Language Processing (NLP) techniques, including rule-based, automatic, and hybrid approaches, to classify large sets of unstructured data, making it easier to analyze and structure information from sources like emails, social media posts, and customer reviews**.**



Types of Sentiment Analysis

1. **Fine-grained Sentiment Analysis**: Focuses on detailed polarity classification, such as very positive, positive, neutral, negative, or very negative, often rated on a scale (e.g., 1 to 5).
2. **Emotion Detection**: Detects emotions like happiness, sadness, anger, or surprise within text, often relying on lexicon-based methods.
3. **Aspect-based Sentiment Analysis**: Analyzes sentiment related to specific aspects or features of a product, such as battery life or camera quality in smartphones.
4. **Multilingual Sentiment Analysis**: Extends sentiment analysis across multiple languages, addressing the challenges of linguistic nuances and cultural differences.

Challenges in Sentiment Analysis

1. **Tone Detection**: Identifying whether a comment is optimistic or pessimistic can be difficult, especially in ambiguous contexts.
2. **Emoji Interpretation**: Decoding emotions from emojis requires advanced models to understand their contextual meaning.
3. **Sarcasm and Irony**: Recognizing sarcasm and irony remains one of the hardest tasks in sentiment analysis.
4. **Neutral Comparisons**: Differentiating neutral sentiments and their implications poses another significant challenge.

Sentiment Analysis and Ethical Considerations

As sentiment analysis involves analyzing personal data, it raises ethical concerns around privacy, consent, and bias. Developing ethical frameworks and adhering to data protection regulations is essential to address these issues.

4.2 TIME SERIES ANALYSIS AND FORECASTING OVERVIEW

Time series analysis and forecasting are essential for understanding historical data and predicting future trends. This technique is widely applied in domains such as finance, economics, healthcare, climate science, and resource management. By identifying trends, patterns, and anomalies in data collected over time, time series analysis aids decision-making, resource optimization, and risk mitigation.

What is a Time Series?

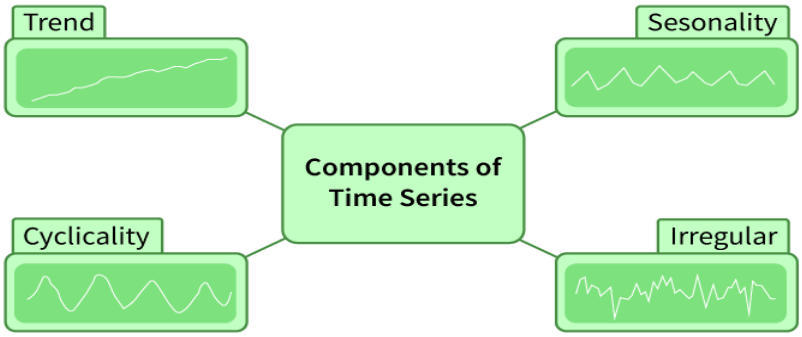
A time series is a sequence of data points collected at successive, evenly spaced time intervals.

- Examples: Stock prices, temperature readings, sales figures.
- Graphical Representation: Typically visualized as a line plot, with time on the x-axis and the variable of interest on the y-axis, facilitating trend and pattern recognition

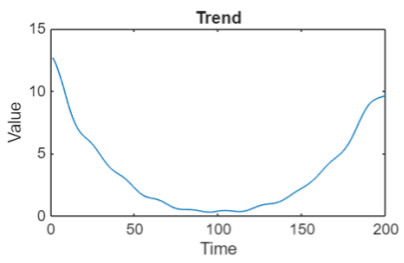
Importance of Time Series Analysis

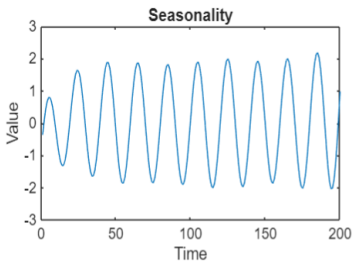
1. **Predict Future Trends**: Enables forecasting of market demand, stock prices, and sales.
2. **Detect Patterns and Anomalies**: Identifies recurring behaviours and outliers.
3. **Strategic Planning**: Supports planning across sectors such as finance and healthcare.
4. **Risk Mitigation**: Helps in early identification of potential risks.
5. **Competitive Edge**: Drives efficient decision-making and resource allocation.

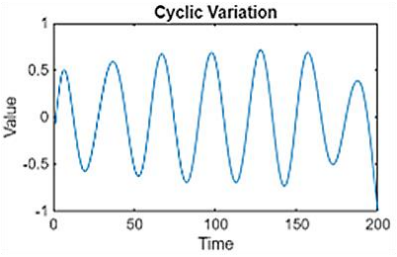
Components of Time Series Data

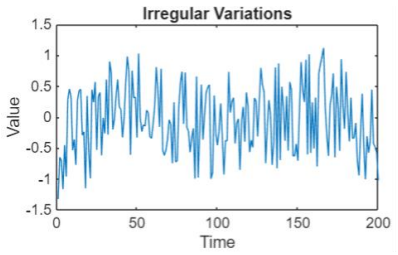


1. **Trend**: Long-term movement in the data (e.g., increasing sales over years).
2. **Seasonality**: Repeating patterns at regular intervals (e.g., peak electricity usage in summer).
3. **Cyclic Variations**: Irregular, longer-term fluctuations tied to economic or business cycles.
4. **Irregularity (Noise)**: Random, unpredictable variations caused by unforeseen factors.









Time Series Analysis Steps

Analyzing time series data involves a systematic approach that integrates various techniques for understanding, modeling, and forecasting data points collected over time.

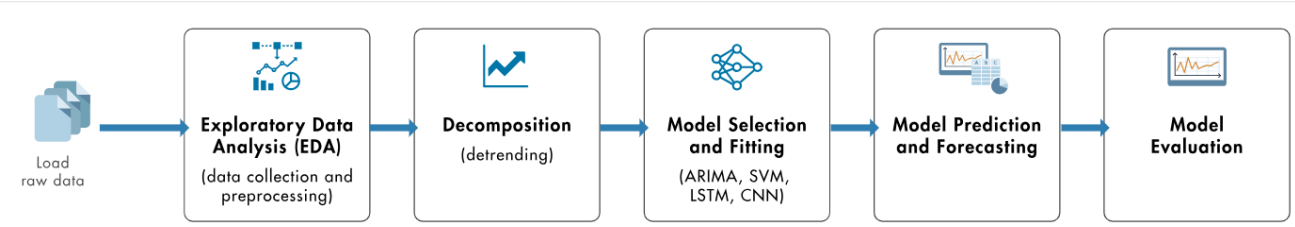


Figure: Work flow for analysing time series data

Visualization Techniques

- **Line Plots**: Shows overall trends and fluctuations.
- **Seasonal Plots**: Highlights periodic patterns.
- **Autocorrelation Plots**: Examines relationships between time lags.
- **Decomposition Plots**: Separates data into trend, seasonality, and residual components

Time Series Analysis Techniques

1) **Exploratory Analysis**:
    a) Visualize data using line plots and histograms to detect patterns.
    b) Analyze autocorrelations with the Autocorrelation Function (ACF).

2) **Decomposition**: Break down the series into its components (trend, seasonality, residuals) to understand the underlying structure.

3) **Stationarity Analysis**:

   a) Ensure the time series is stationary (mean and variance constant over time).

   b) Use statistical tests like the Augmented Dickey-Fuller (ADF) test for validation

Time Series Forecasting Techniques

1. **Statistical Methods**:

   a) **ARIMA** (AutoRegressive Integrated Moving Average): Combines autoregression (AR), differencing (I), and moving averages (MA).

   b) **SARIMA** (Seasonal ARIMA): Extends ARIMA by modeling seasonality.

   c) **Exponential Smoothing** (Holt-Winters): Smoothens data for trend and seasonal forecasting.

2. **Machine Learning Approaches**:

   a) **Linear Regression**: Uses historical data as features for prediction.

   b) **Support Vector Machines** (SVM): Fits a hyperplane to capture patterns in time series.

3. **Deep Learning Methods**:

   a) **Recurrent Neural Networks** (RNNs): Processes sequential data effectively.

   b) **Long Short-Term Memory Networks** (LSTMs): Handles long-term dependencies in data.

4. **Hybrid Methods**: Combines statistical models with machine learning to improve prediction accuracy.

Performance Metrics

1. **Mean Absolute Error (MAE)**: Average magnitude of prediction errors.
2. **Mean Squared Error (MSE)**: Squared average of prediction errors.
3. **Root Mean Squared Error (RMSE)**: Square root of MSE.
4. **Forecast Bias**: Detects over- or underestimation trends.
5. **Theil's U Statistic**: Benchmarks forecasting performance.

Libraries and Tools

- **Statsmodels**: Statistical modeling and hypothesis testing.
- **Pmdarima**: Simplified ARIMA modeling.
- **Prophet**: Scalable forecasting with built-in seasonality handling.
- **tslearn**: Advanced algorithms for classification and clustering.

CHAPTER V

PROJECTS

5.1 MINI PROJECT: SENTIMENT ANALYSIS

5.1.1 Project Aim

The aim of this project is to analyze and classify the sentiment of textual data (positive, negative, or neutral) using machine learning algorithms. The objective is to build an efficient and accurate sentiment classification system that can be applied to social media data, customer reviews, or other text-based data sources.

5.1.2 Existing Approach

The existing sentiment analysis methods are primarily rule-based, lexicon-based, or machine learning-based. Rule-based systems rely on pre-defined rules, while lexicon-based approaches use dictionaries of sentiment-related words (e.g., VADER). However, these approaches struggle with understanding contextual nuances, such as sarcasm or ambiguity.

5.1.3 Proposed Solution

The proposed solution involves a hybrid approach of leveraging VADER for quick analysis and employing supervised machine learning models like Logistic Regression, Bernoulli Naive Bayes, and Linear SVC for more accurate predictions.

5.1.4 Requirements

a) **Hardware**

A system with at least **8 GB of RAM** and a multi-core processor for handling large datasets.

b) **Software**

- **Python 3.x**
- **Libraries:** NumPy, pandas, matplotlib, scikit-learn, NLTK, wordcloud

5.1.5 Dataset Description

1. Dataset Name: **Sentiment140 Dataset**

2. Size: **1.6 million tweets** with sentiment labels (positive, negative, or neutral).
3. Attributes
    a. **tweet_id**: Unique identifier for each tweet.
    b. **text**: The content of the tweet.
    c. **label**: Sentiment of the tweet (**0 = negative, 2 = neutral, 4 = positive**).
4. Purpose: To analyze sentiment patterns in social media data.

5.1.6 Technologies and Tools Used

1. Programming Language: Python
2. Libraries:
    a) Text preprocessing: NLTK, re
    b) Machine Learning: scikit-learn
    c) Data Manipulation: pandas, NumPy
    d) Visualization: matplotlib, seaborn
3. Development Environment: **Jupyter Notebook**

5.1.7 Methodology

1. Data collection

The Sentiment140 dataset was downloaded and loaded into a pandas DataFrame.

2. Data Preprocessing

- Tokenization and lowercasing of text.
- Removal of stopwords, special characters, and URLs.
- Stemming to reduce words to their base forms.

3. Feature Extraction

Converting text data into numerical form using TF-IDF vectorization

4. Models Used

- **VADER (Valence Aware Dictionary and sEntiment Reasoner)**

a) Why?

VADER is optimized for analyzing social media text and other informal text formats. It provides sentiment polarity based on an extensive lexicon of sentiment-laden words.

b) Working

1. Assigns each word in the text a sentiment score from its lexicon
2. Combines the scores, considering punctuation, capitalization, and modifiers (e.g., "not good")
3. Outputs a compound sentiment score in the range [-1, 1]

c) Pseudocode

Input: Text

1. Tokenize the input text.

2. Assign sentiment scores to words using the VADER lexicon.

3. Adjust scores for modifiers (e.g., negations, exclamations).

4. Aggregate sentiment scores:

Compound Score = Sum(Weighted Scores) / Normalization Factor

5. Output: Positive, Negative, or Neutral sentiment.

- **Logistic Regression**

a) Why?

Logistic Regression is simple, interpretable, and efficient for binary classification problems like sentiment analysis.

b) Working

1. Computes a weighted sum of features (e.g., TF-IDF values) and applies the sigmoid function to predict the probability of belonging to a class.
2. The classification threshold determines whether a sentiment is positive or negative.

c) Pseudocode

Input: Feature vector X, Labels Y

1. Initialize weights W and bias b.

2. Compute the linear combination:

$Z = W.X + b$

3. Apply the sigmoid function:

$P(Y=1|X) = 1 / (1 + \exp(-Z))$

4. Compute the loss using binary cross-entropy:

$Loss = - (1/N) \Sigma [Y*\log(P) + (1-Y)*\log(1-P)]$

5. Optimize W and b using gradient descent.

6. Output: Classify sentiment based on $P > 0.5$

- ## Bernoulli Naive Bayes

a) Why?

Efficient for binary feature representations like word presence/absence in text.

b) Working

1. Calculates the likelihood of a sentiment label given word presence/absence using Bayes' theorem.
2. Assumes feature independence to simplify probability computation.

c) Pseudocode

Input: Feature Matrix X, Labels Y

1. Calculate prior probabilities for each class:

P(C) = Count(C) / Total Samples

2. Calculate likelihoods:

P(X|C) = Product of $P(x_i|C)$ for all features xi.

3. Compute posterior probabilities:

$P(C|X) \propto P(C) * P(X|C)$

4. Select the class with the highest posterior:

Predicted Class = argmax(P(C|X)).

- **Linear SVC (Support Vector Classifier)**

a) Why?

Effective for high-dimensional datasets like TF-IDF vectors and robust in handling text classification problems.

b) Working

1. Maximizes the margin between sentiment classes by finding a hyperplane in the feature space.
2. Classifies data points based on which side of the hyperplane they fall.

c) Pseudocode

Input: Feature matrix X, Labels Y

1. Initialize hyperplane parameters W and b.

2. For each sample, compute:

Decision Value = W.X + b

3. Maximize the margin:

Minimize ||W|| subject to $yi(W.Xi + b) \geq 1$ for all i.

4. Solve using optimization techniques like gradient descent.

5. Output: Classify sentiment based on decision value.

5. Model Evaluation

Calculated precision, recall, F1-score, and accuracy using a test set

5.1.8 Challenges Faced

- Addressing sarcastic or context-dependent tweets.
- Optimizing hyperparameters for machine learning models.

5.1.9 Code Implementation

```
# Import necessary libraries
import pandas as pd
import numpy as np
import re
import string
import nltk
import seaborn as sns
import matplotlib.pyplot as plt
from nltk.sentiment import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, PorterStemmer
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc, accuracy_score
```

```
# Download necessary NLTK resources
nltk.download('vader_lexicon')
nltk.download('stopwords')
nltk.download('wordnet')

# Load and preprocess the dataset
columns = ['target', 'id', 'date', 'flag', 'user', 'text']
dataset_path = "/content/drive/MyDrive/to_post/training.1600000.processed.noemoticon.csv"
data = pd.read_csv(dataset_path, encoding='ISO-8859-1', names=columns)[['target', 'text']]
data['target'] = data['target'].map({0: 0, 4: 1})  # Map 0 to Negative, 4 to Positive
```

```
# Text preprocessing functions
def preprocess_text(text):
    text = re.sub(r'@\w+', '', text)  # Remove mentions
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove non-alphabetic characters
    text = re.sub(r'(.)\1+', r'\1', text)  # Remove repeating characters
    text = text.lower()  # Convert to lowercase
    return text
```

```
def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    return " ".join([word for word in text.split() if word not in stop_words])
```

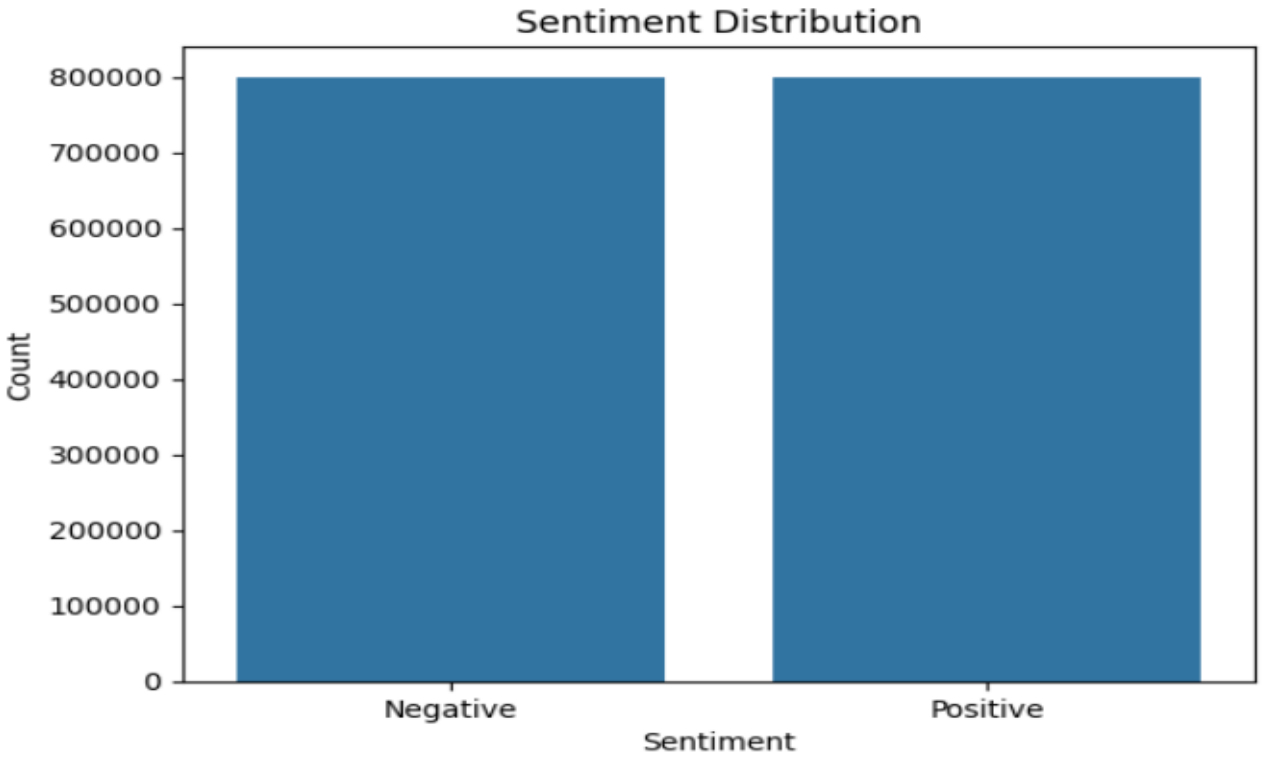
```
[4] def lemmatize_text(text):
        lemmatizer = WordNetLemmatizer()
        return " ".join([lemmatizer.lemmatize(word) for word in text.split()])
```

```
[5] # Apply preprocessing
    data['text'] = data['text'].apply(preprocess_text).apply(remove_stopwords).apply(lemmatize_text)
```

```
# Display basic dataset information
print(f"Dataset Shape: {data.shape}")
print(data.head())
```

```
Dataset Shape: (1600000, 2)
   target                                                text
0       0     aw thats bumer shoulda got david car third day
1       0  upset cant update facebok texting might cry re...
2       0     dived many time bal managed save rest go bound
3       0                    whole body fels itchy like fire
4       0                      behaving al im mad cant se al
```

```
# Visualize data distribution
sns.countplot(x='target', data=data)
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.xticks(ticks=[0, 1], labels=['Negative', 'Positive'])
plt.show()
```



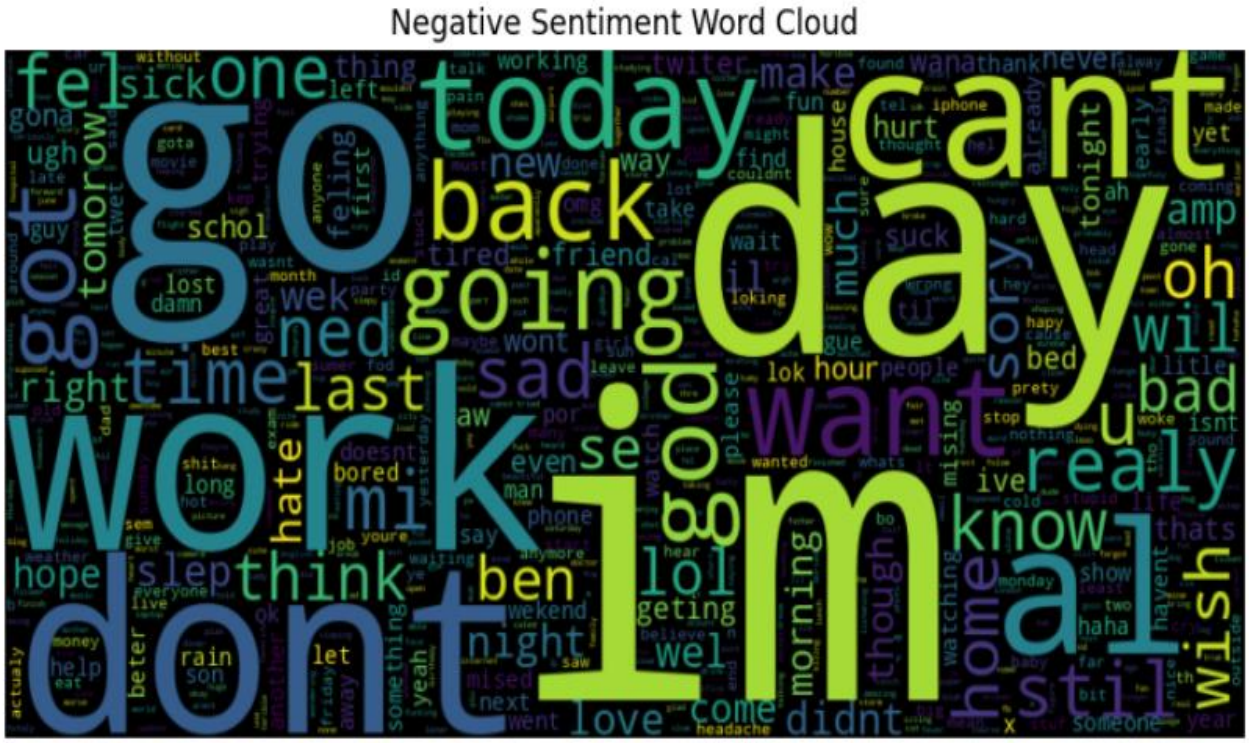
Observations:

The sentiment distribution chart reveals a balanced dataset with nearly equal instances of negative and positive sentiments, each comprising approximately 800,000 samples. This balance is crucial for training machine learning models, as it minimizes the risk of bias toward one class and ensures fair representation of both sentiment categories. The x-axis denotes the two sentiment labels, "Negative" and "Positive," while the y-axis displays their respective counts.

```
[8]  # Generate word clouds for positive and negative sentiments
     positive_text = " ".join(data[data['target'] == 1]['text'])
     negative_text = " ".join(data[data['target'] == 0]['text'])
```

```
plt.figure(figsize = (20,20))
plt.subplot(1, 2, 1)
plt.imshow(WordCloud(max_words=1000, width=800, height=400, collocations=False).generate(negative_text), interpolation='bilinear')
plt.title('Negative Sentiment Word Cloud')
plt.axis('off')
```

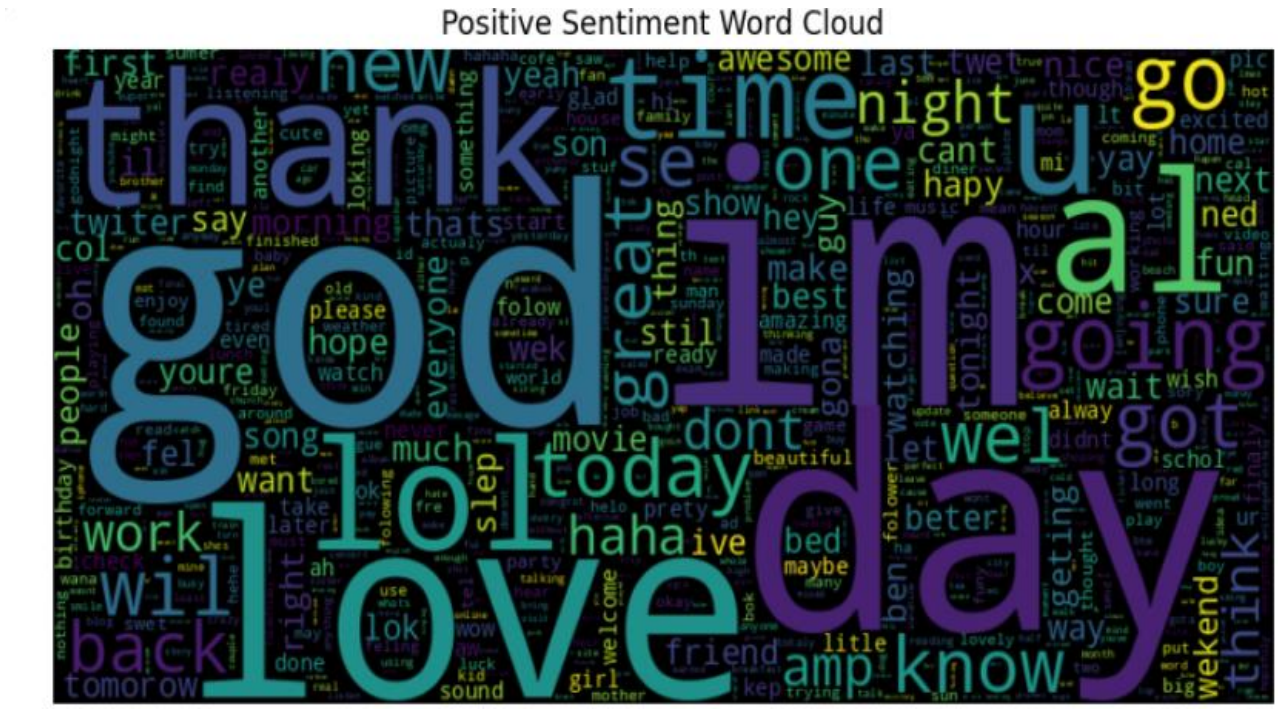
```
(-0.5, 799.5, 399.5, -0.5)
```



Observations:

The Negative Sentiment Word Cloud highlights the most frequent words associated with negative sentiments in the dataset. Prominent words such as *"day," "work," "don't,"* and *"back"* suggest common themes of dissatisfaction or struggles in daily life and work-related contexts. The size of the words indicates their frequency, with larger words being more frequent. Other words like *"bad," "sad,"* and *"hate"* emphasize expressions of frustration, sadness, or negativity.

```
plt.figure(figsize = (20,20))
plt.subplot(1, 2, 2)
plt.imshow(WordCloud(max_words=1000, width=800, height=400, collocations=False).generate(positive_text), interpolation='bilinear')
plt.title('Positive Sentiment Word Cloud')
plt.axis('off')
plt.show()
```



Observations:

The Positive Sentiment Word Cloud highlights frequently occurring words associated with positive sentiments in the dataset. Words like *"thank," "good," "love,"* and *"great"* are prominent, indicating expressions of gratitude, happiness, and appreciation. Other terms such as *"today," "day," "hope,"* and *"fun"* reflect positive emotions related to daily activities and optimistic outlooks. The presence of words like *"awesome," "nice,"* and *"best"* further emphasizes uplifting and cheerful sentiments.

```
[11] # Initialize VADER Sentiment Analyzer
     sia = SentimentIntensityAnalyzer()
     data['vader_score'] = data['text'].apply(lambda x: sia.polarity_scores(x)['compound'])
```

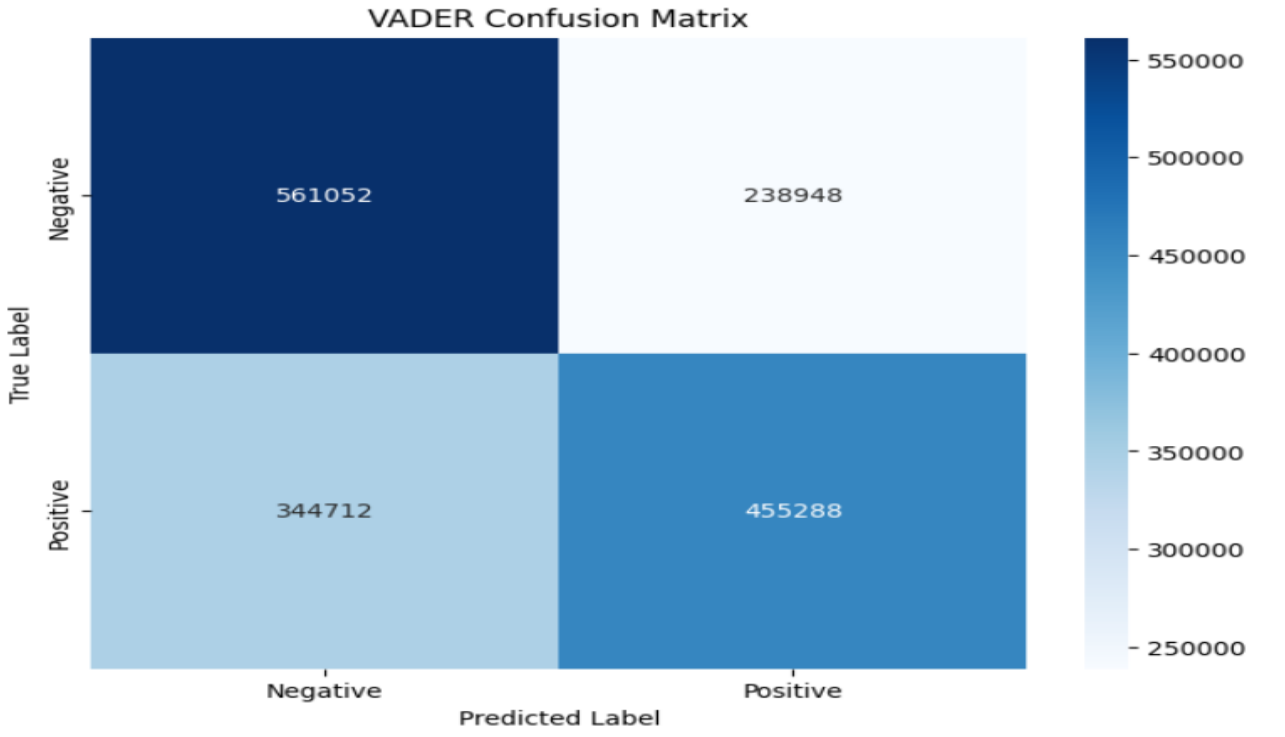
```
[12] # Apply VADER-based classification
     data['vader_sentiment'] = data['vader_score'].apply(lambda x: 1 if x > 0.1 else 0)
```

```
[13] # Evaluate VADER model
     accuracy_vader = accuracy_score(data['target'], data['vader_sentiment'])
     conf_matrix_vader = confusion_matrix(data['target'], data['vader_sentiment'])
     report_vader = classification_report(data['target'], data['vader_sentiment'])
```

```python
print(f"VADER Accuracy: {accuracy_vader * 100:.2f}%")
print("\nVADER Confusion Matrix:")
print(conf_matrix_vader)
print("\nVADER Classification Report:")
print(report_vader)
```

```
VADER Accuracy: 63.52%

VADER Confusion Matrix:
[[561052 238948]
 [344712 455288]]

VADER Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.70      0.66    800000
           1       0.66      0.57      0.61    800000

    accuracy                           0.64   1600000
   macro avg       0.64      0.64      0.63   1600000
weighted avg       0.64      0.64      0.63   1600000
```

```python
# Visualize VADER confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix_vader, annot=True, fmt="d", cmap="Blues", xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('VADER Confusion Matrix')
plt.show()
```



Observations:

The VADER sentiment analysis model achieved an accuracy of 63.52%. The confusion matrix shows 561,052 true negatives, 238,948 false positives, 344,712 false negatives, and 455,288 true positives. The model has a precision of 0.62 for the negative class (0) and 0.66 for the positive class (1), indicating it performs slightly better at predicting positive sentiment. The recall is higher for the negative class (0) at 0.70 compared to 0.57 for the positive class (1), suggesting the model is better at identifying negative sentiment. The F1-scores are 0.66 for class 0 and 0.61 for class 1, reflecting a moderate balance in performance. Overall, the macro and weighted averages for precision, recall, and F1-score are 0.64, indicating a reasonably balanced but slightly more favorable performance for the negative class.

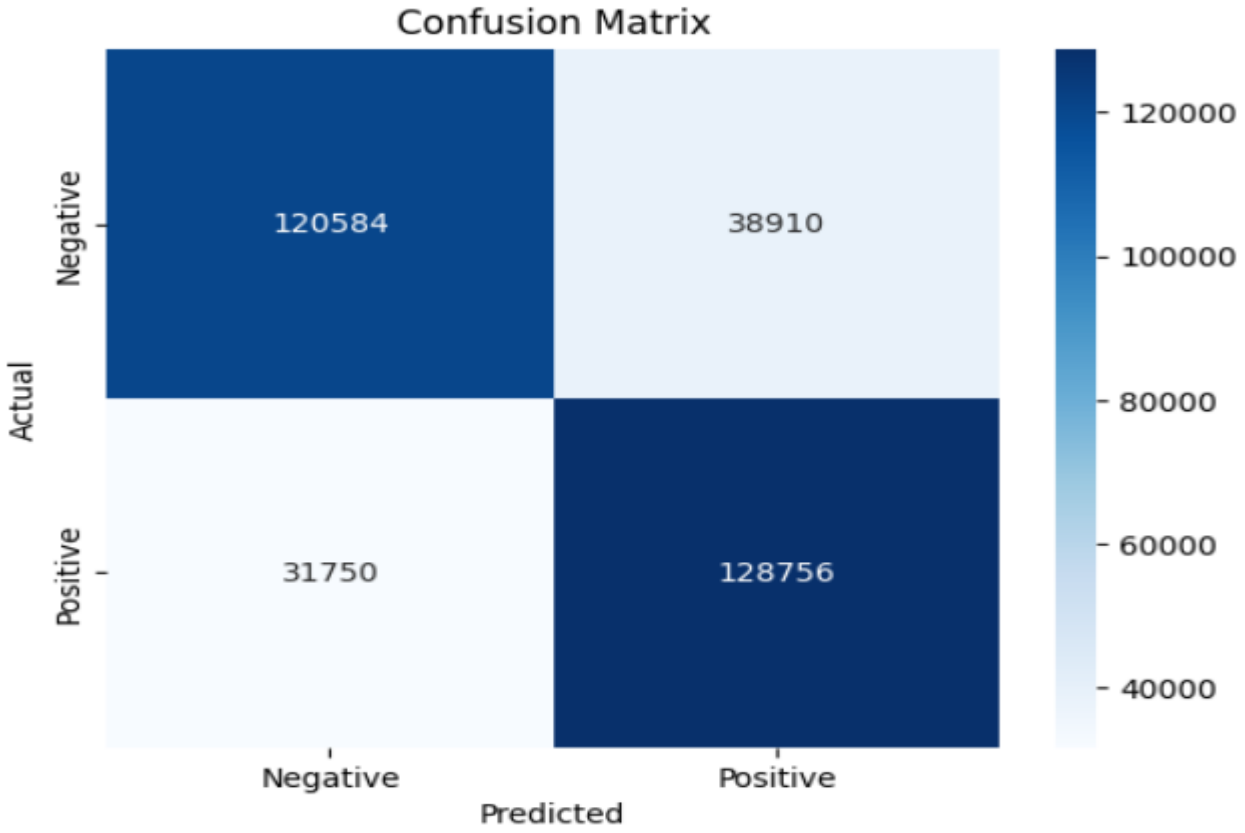
```
# TF-IDF and machine learning models
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1, 2))
X = vectorizer.fit_transform(data['text'])
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Define evaluation function
def evaluate_model(model):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    cf_matrix = confusion_matrix(y_test, y_pred)
    sns.heatmap(cf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix')
    plt.show()
```

```
# Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
print("Logistic Regression Results:")
evaluate_model(lr_model)
```

Logistic Regression Results:

	precision	recall	f1-score	support
0	0.79	0.76	0.77	159494
1	0.77	0.80	0.78	160506
accuracy			0.78	320000
macro avg	0.78	0.78	0.78	320000
weighted avg	0.78	0.78	0.78	320000



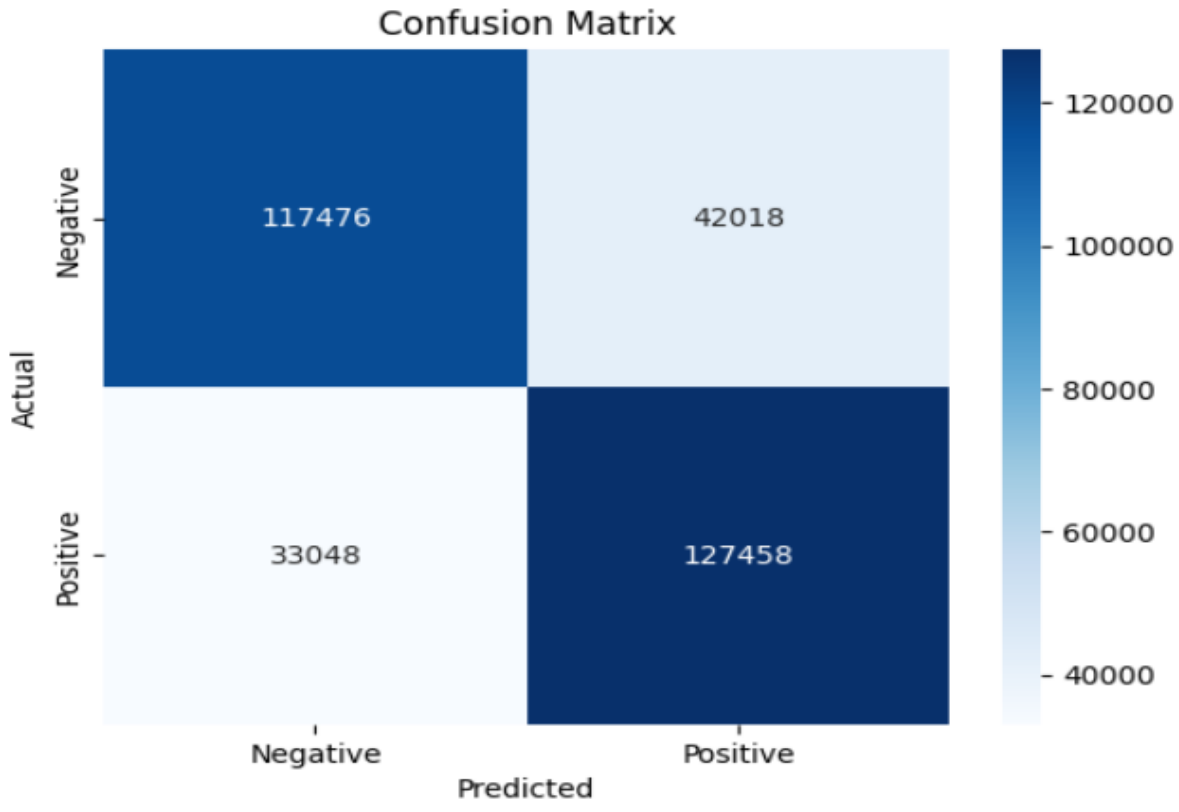
Observations:

The Logistic Regression model achieved an accuracy of 78%. The precision for the negative class (0) is 0.79, while for the positive class (1), it is 0.77. The recall is 0.76 for the negative class and 0.80 for the positive class, indicating the model is slightly better at identifying positive sentiment. The F1-scores are 0.77 for class 0 and 0.78 for class 1, demonstrating a balanced performance between both

classes. The macro and weighted averages for precision, recall, and F1-score are all 0.78, indicating consistent performance across both classes.

```
# Bernoulli Naive Bayes
bnb_model = BernoulliNB()
bnb_model.fit(X_train, y_train)
print("Bernoulli Naive Bayes Results:")
evaluate_model(bnb_model)
```

```
Bernoulli Naive Bayes Results:
              precision    recall  f1-score   support

           0       0.78      0.74      0.76    159494
           1       0.75      0.79      0.77    160506

    accuracy                           0.77    320000
   macro avg       0.77      0.77      0.77    320000
weighted avg       0.77      0.77      0.77    320000
```



Observations:

The Bernoulli Naive Bayes model achieved an accuracy of 77%. The precision for the negative class (0) is 0.78, while for the positive class (1), it is 0.75. The recall is 0.74 for the negative class and 0.79 for the positive class, indicating the model is slightly better at identifying positive sentiment. The F1-scores are 0.76 for class 0 and 0.77 for class 1, demonstrating a fairly balanced performance between both classes. The macro and weighted averages for precision, recall, and F1-score are all 0.77, indicating consistent performance across both classes.

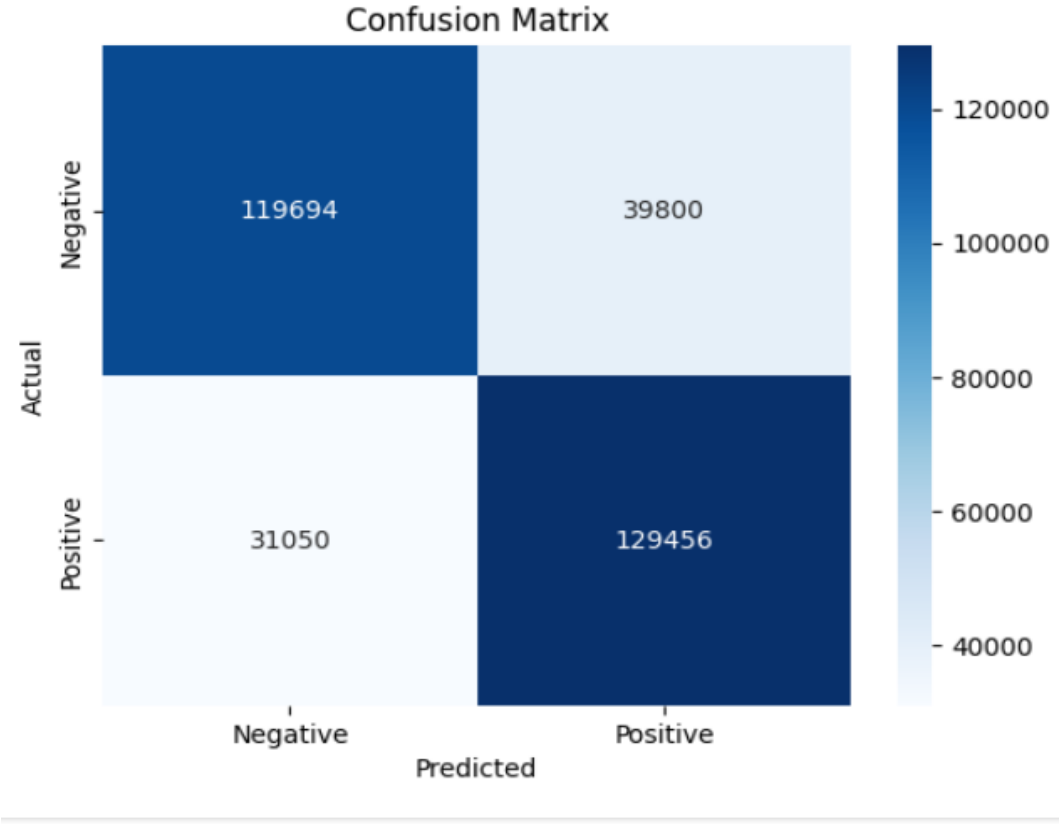
```
# Linear SVC
svc_model = LinearSVC()
svc_model.fit(X_train, y_train)
print("Linear SVC Results:")
evaluate_model(svc_model)
```

```
Linear SVC Results:
              precision    recall  f1-score   support

           0       0.79      0.75      0.77    159494
           1       0.76      0.81      0.79    160506

    accuracy                           0.78    320000
   macro avg       0.78      0.78      0.78    320000
weighted avg       0.78      0.78      0.78    320000
```

Observations:

The Linear SVC model achieved an accuracy of 78%. The precision for the negative class (0) is 0.79, while for the positive class (1), it is 0.76. The recall is 0.75 for the negative class and 0.81 for the positive class, indicating the model is slightly better at identifying positive sentiment. The F1-scores are 0.77 for class 0 and 0.79 for class 1, showing a balanced performance with a slight advantage for

the positive class. The macro and weighted averages for precision, recall, and F1-score are all 0.78, reflecting consistent performance across both classes.




```
# ROC Curve for Logistic Regression
fpr, tpr, thresholds = roc_curve(y_test, lr_model.predict_proba(X_test)[:, 1])
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'Logistic Regression AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

Observations:

The ROC Curve for Logistic Regression shows an Area Under the Curve (AUC) of 0.86, indicating good model performance. The curve is significantly above the diagonal (random guess line), demonstrating that the model effectively distinguishes between the positive and negative classes. The True Positive Rate (sensitivity) increases as the False Positive Rate increases, showing a strong trade-

off. The high AUC value suggests that the model has a good balance between sensitivity and specificity, making it reliable for classification tasks.

```python
def predict_sentiment(user_input, vectorizer, model):
    # Preprocess user input
    preprocessed_input = preprocess_text(user_input)
    preprocessed_input = remove_stopwords(preprocessed_input)
    preprocessed_input = lemmatize_text(preprocessed_input)

    # Transform input using TF-IDF vectorizer
    input_vector = vectorizer.transform([preprocessed_input])

    # Predict sentiment using the provided model
    prediction = model.predict(input_vector)[0]

    # Map prediction to sentiment
    sentiment = "Positive" if prediction == 1 else "Negative"
    return sentiment

# Example usage
user_input = input("Enter a sentence to analyze its sentiment: ")
predicted_sentiment = predict_sentiment(user_input, vectorizer, lr_model)  # Using Logistic Regression model
print(f"The sentiment of the input text is: {predicted_sentiment}")
```

```
Enter a sentence to analyze its sentiment: I love this product! It's amazing!
The sentiment of the input text is: Positive

Enter a sentence to analyze its sentiment: This is the worst experience I have ever had.
The sentiment of the input text is: Negative
```

5.1.10 Outcome

The sentiment analysis project using the Sentiment140 dataset involved evaluating VADER, Logistic Regression, Bernoulli Naive Bayes, and Linear SVC. **VADER** achieved an accuracy of **63.52%**, while **Logistic Regression and Linear SVC** performed best with **78%** accuracy, balancing precision and recall. **Bernoulli Naive Bayes** showed a slightly lower accuracy of **77%**. In practical tests, the models accurately classified sentiment: for example, "**I love this product! It's amazing!" was classified as Positive**, and **"This is the worst experience I have ever had." was classified as Negative**. These results highlight the effectiveness of the models, with **Logistic Regression and Linear SVC being the most suitable** for general sentiment analysis, and VADER excelling in informal text.

5.2 MAJOR PROJECT: TIME SERIES ANALYSIS AND FORECASTING OF DISNEY+ CONTENT ADDITIONS

5.2.1 Project Aim

The aim of this project is to perform time series analysis and forecasting on the historical data of Disney+ content additions in order to predict future trends in the number of titles added to the platform. The primary goal is to build a robust forecasting model using the ARIMA (AutoRegressive Integrated Moving Average) method to predict the number of new titles (movies and TV shows) that Disney+ will add over the next 12 months. This project seeks to provide insights into the growth patterns of Disney+ content, helping the platform plan and optimize its content strategy for better user engagement. The analysis will also include exploring seasonal trends and identifying potential influences that affect content addition patterns.

5.2.2 Existing Approach

Currently, Disney+ and other streaming platforms rely on traditional methods for analyzing content trends, often focusing on manual tracking of content release schedules and user engagement metrics. This includes using basic statistical analysis or historical trends to guide content strategies. However, these methods are often reactive and lack the ability to predict future trends with high accuracy.

Most platforms track content performance, such as viewership data and genre popularity, but there is limited integration of these metrics into comprehensive predictive models. Existing approaches also tend to overlook the complexity of seasonality and other subtle patterns in content addition, resulting in less efficient content strategies and missed opportunities for optimizing user engagement.

5.2.3 Proposed Solution

To address the limitations of current methods, the proposed solution involves utilizing an advanced time series analysis approach, specifically the ARIMA forecasting model, to predict the

number of titles added to Disney+ over the next 12 months. The solution includes preprocessing the data for accuracy, conducting exploratory data analysis to uncover patterns, and applying ARIMA to forecast future trends. By decomposing the time series data into trend and seasonal components, the model will provide more accurate predictions, aiding in proactive content planning and strategy optimization for Disney+.

5.2.4 Requirements

a) Hardware

- Processor: **Intel Core i5** or higher (recommended for faster data processing and model training)
- RAM: **Minimum 8 GB** (16 GB or higher recommended for handling large datasets efficiently)
- Storage: **Minimum 10 GB of free disk space** for dataset storage and software installation
- Graphics: Standard **integrated graphics** (for data visualization and chart rendering)
- Operating System: **Windows, macOS, or Linux** (with necessary software and library compatibility)

b) Software

- Programming Language: **Python**
- Libraries and Frameworks:
    - **Pandas**: For data manipulation and analysis
    - **NumPy**: For numerical operations
    - **Matplotlib/Seaborn**: For data visualization
    - **Statsmodels**: For time series analysis and ARIMA modeling
    - **WordCloud**: For generating word clouds from text data

- **Jupyter Notebook**: For interactive development and documentation

- Integrated Development Environment (IDE): **Visual Studio Code, Google Colab, PyCharm, or Jupyter Notebook** (for writing and running Python code)

- Operating System Requirements: Python 3.x, along with the relevant package managers (such as **pip**) to install dependencies

5.2.5 Dataset Description

The dataset consists of 1,450 entries and 12 columns, providing comprehensive details about Disney+ movies and TV shows. Key attributes include:

- Title: Name of the content.
- Type: Specifies whether the content is a Movie or a TV Show.
- Director: The director(s) of the content.
- Cast: Leading actors or actresses featured.
- Country: Country of origin.
- Date Added: The date the content was added to Disney+.
- Release Year: The year the content was originally released.

- Rating: Age suitability ratings (e.g., TV-G, PG).
- Duration: Duration of movies or the number of seasons for TV shows.
- Listed In: Genres or categories.
- Description: Brief summary of the content.

The dataset includes:

- Missing Values: 3 entries with missing date_added.
- "Unknown" Labels: Found in columns such as director, cast, and country.

5.2.6 Methodology

1. Data Collection and Preparation

- The dataset was loaded and inspected for consistency, missing values, and data types.
- Missing values were filled with appropriate placeholders or handled through imputation.
- The date_added column was converted to a proper datetime format to facilitate time-based analysis.

2. Exploratory Data Analysis (EDA)

- Conducted EDA to understand the dataset's structure, distribution, and key insights, including content type distribution, rating patterns, genre prevalence, and production trends across countries.
- Generated visualizations using libraries like Matplotlib and Seaborn to summarize findings effectively.

3. Time Series Creation

- Extracted and aggregated the date_added column to form a monthly frequency time series.
- The time series was inspected to identify trends, seasonality, and irregularities in the number of titles added over time.

4. Time Series Decomposition

- Decomposed the time series into trend, seasonal, and residual components using the additive decomposition model from the Statsmodels library.
- This helped isolate and analyze underlying patterns, providing clarity on long-term trends and periodic variations.

5. Model Selection and Implementation

- Selected the ARIMA model for forecasting due to its suitability for time series data with potential trends and stationarity concerns.

  Why ARIMA?

  The ARIMA model was chosen due to its ability to handle both stationary and non-stationary time series data effectively. Its flexibility in modeling the temporal dependencies (autoregressive and moving average components) and differencing to achieve stationarity made it suitable for capturing the trends and patterns in the dataset. The model's capacity to produce reliable short-term forecasts aligned with the project goals of predicting the number of titles to be added in the upcoming months.
- The model parameters were determined using iterative tuning and evaluation to optimize the forecast accuracy.

6. Forecasting

- Forecasted the number of titles expected to be added for the next 12 months.
- The forecasted data was plotted alongside the original time series to visualize future trends and provide actionable insights.

7. Visualization and Presentation

- Visualizations were created for all stages of analysis, including distribution charts, bar plots, word clouds, and time series plots, to support storytelling and data-driven conclusions.
- Presented the findings through detailed graphs and summaries to effectively communicate insights.

5.2.7 Challenges Faced

1. **Uneven Distribution of Ratings/Genres**:

   The distribution of ratings or genres might be skewed, which can affect the analysis. Some categories might have disproportionately large counts, making it harder to detect trends in smaller groups.

2. **High Frequency of 'Unknown' Values**:

   Many entries in the director and cast columns were marked as 'Unknown'. This caused challenges when attempting to rank the top directors or cast members, as a large portion of the data did not contain valuable information. Handling these unknowns effectively was crucial to avoid skewing the analysis and ensure that top contributors were properly recognized.

3. **Dealing with Inconsistent Data Formats**:

   Fields such as *date_added* required conversion to a consistent datetime format. Errors during conversion (e.g., coercing invalid dates) required careful handling to prevent data loss or inaccuracies.

4. **Visualization Constraints**:

   Creating effective visualizations was challenging, particularly for bar plots. The presence of deprecated warnings (e.g., assigning palette without hue) required adjustments to maintain compliance with updated library versions.

5.2.8 Code Implementation

```
[3]  # Import necessary libraries
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns


     from wordcloud import WordCloud
     from statsmodels.tsa.seasonal import seasonal_decompose
     from statsmodels.tsa.arima.model import ARIMA
```

```
[4]  # Load the dataset
     df = pd.read_csv('/content/drive/MyDrive/to_post/disney_plus_titles.csv')
```

```
[5]  # Handle missing values
     df.fillna('Unknown', inplace=True)
```

```
[6]  # Convert date_added to datetime
     df['date_added'] = pd.to_datetime(df['date_added'], errors='coerce')
```

```
[9]  print("Dataset Overview:")
     print(df.info())
     print("\n First few rows in the dataset: ")
     print(df.head())
```

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1450 entries, 0 to 1449
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       1450 non-null   object
 1   type          1450 non-null   object
 2   title         1450 non-null   object
 3   director      1450 non-null   object
 4   cast          1450 non-null   object
 5   country       1450 non-null   object
 6   date_added    1447 non-null   datetime64[ns]
 7   release_year  1450 non-null   int64
 8   rating        1450 non-null   object
 9   duration      1450 non-null   object
 10  listed_in     1450 non-null   object
 11  description   1450 non-null   object
dtypes: datetime64[ns](1), int64(1), object(10)
memory usage: 136.1+ KB
None
```

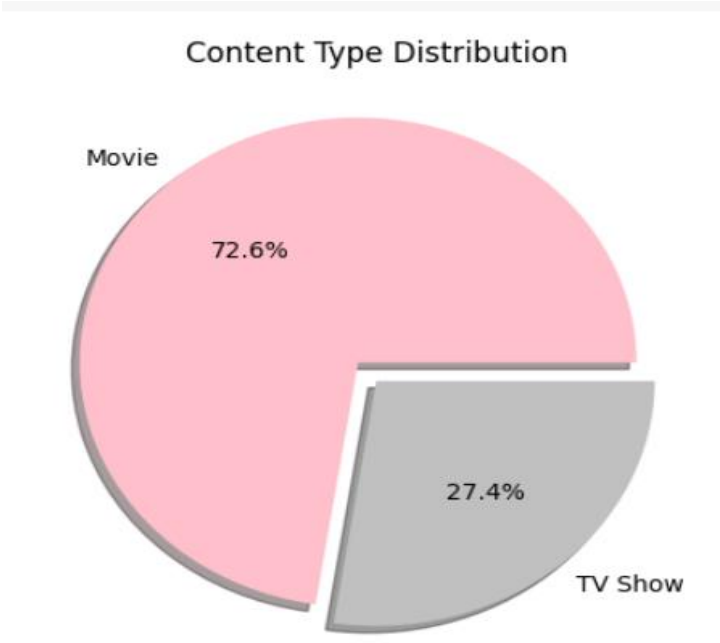
First few rows in the dataset:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Duck the Halls: A Mickey Mouse Christmas Special	Alonso Ramirez Ramos, Dave Wasson	Chris Diamantopoulos, Tony Anselmo, Tress MacN...	Unknown	2021-11-26	2016	TV-G	23 min	Animation, Family	Join Mickey and the gang as they duck the halls!
1	s2	Movie	Ernest Saves Christmas	John Cherry	Jim Varney, Noelle Parker, Douglas Seale	Unknown	2021-11-26	1988	PG	91 min	Comedy	Santa Claus passes his magic bag to a new St. ...
2	s3	Movie	Ice Age: A Mammoth Christmas	Karen Disher	Raymond Albert Romano, John Leguizamo, Denis L...	United States	2021-11-26	2011	TV-G	23 min	Animation, Comedy, Family	Sid the Sloth is on Santa's naughty list.
3	s4	Movie	The Queen Family Singalong	Hamish Hamilton	Darren Criss, Adam Lambert, Derek Hough, Alexa...	Unknown	2021-11-26	2021	TV-PG	41 min	Musical	This is real life, not just fantasy!
4	s5	TV Show	The Beatles: Get Back	Unknown	John Lennon, Paul McCartney, George Harrison, ...	Unknown	2021-11-25	2021	Unknown	1 Season	Docuseries, Historical, Music	A three-part documentary from Peter Jackson ca...

```python
# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Missing Values:
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      3
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

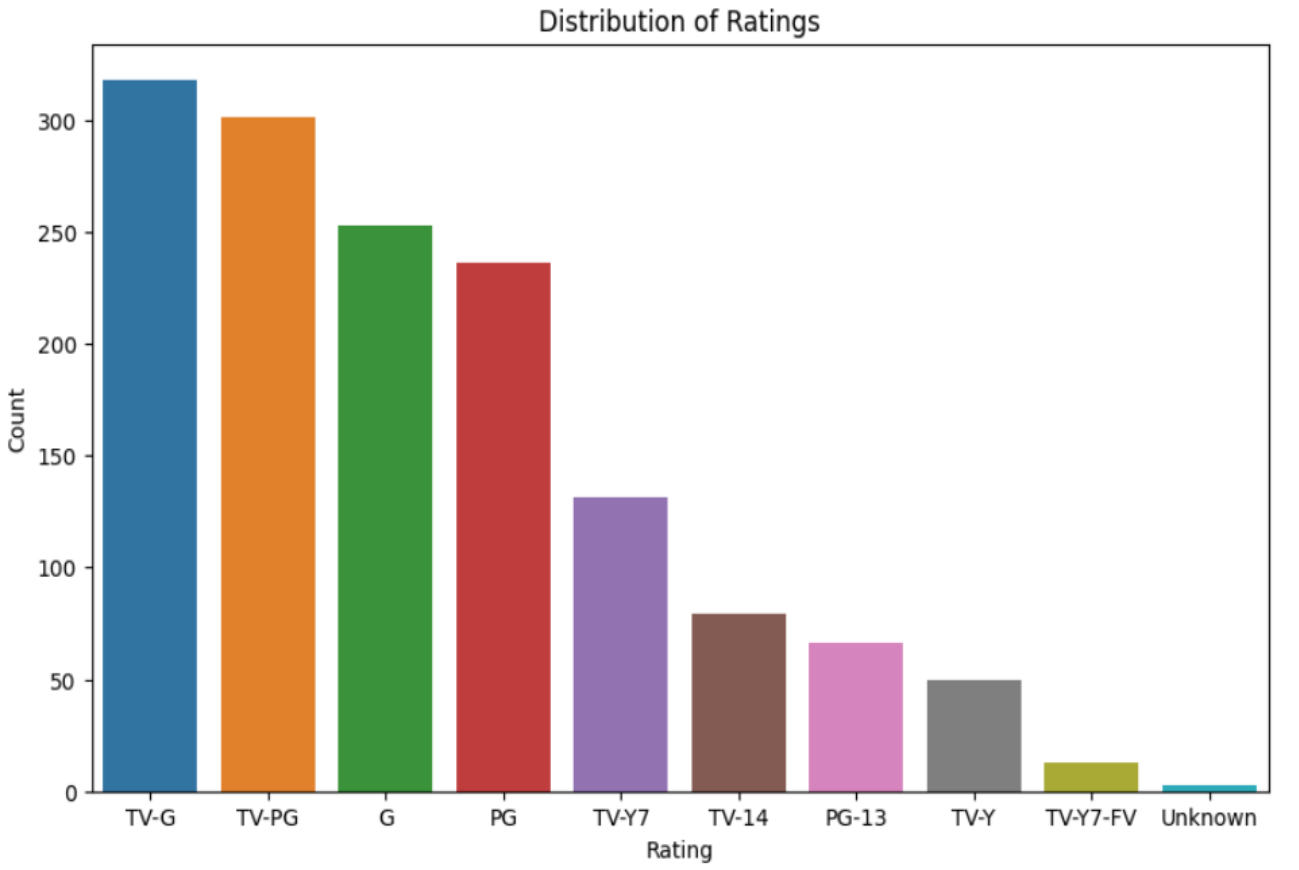
```python
# Distribution of content types (Movies vs. TV Shows)
content_counts = df['type'].value_counts()
plt.figure(figsize=(8, 5))
content_counts.plot(kind='pie', autopct='%1.1f%%', explode=[0.1, 0], shadow=True, colors=['#FFC0CB', '#C0C0C0'])
plt.title('Content Type Distribution')
plt.ylabel('')
plt.show()
```



Insights:

The pie chart illustrates the distribution of content types on the Disney+ platform, revealing a clear dominance of Movies. Movies constitute a substantial 72.6% of the platform's content library, while TV Shows represent the remaining 27.4%. This significant disparity highlights that Movies are the primary focus and draw for subscribers on Disney+.

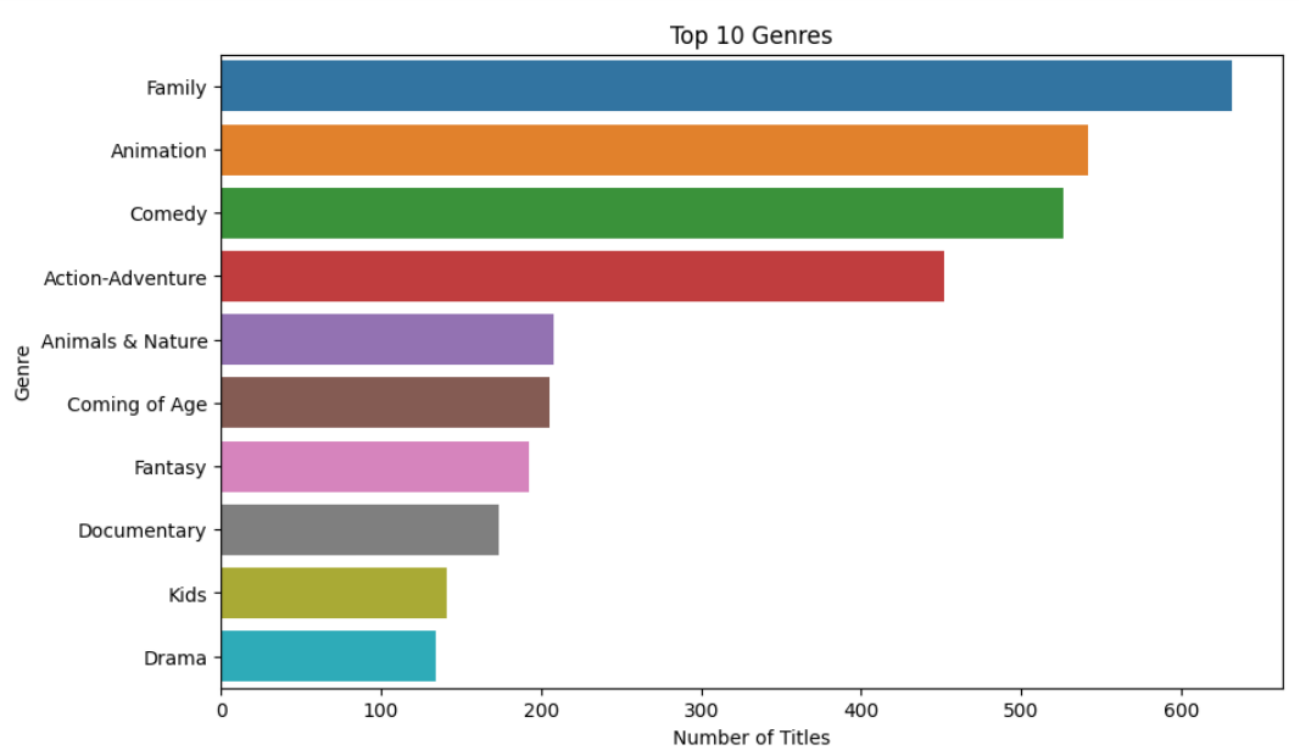
```
# Rating distribution
rating_counts = df['rating'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=rating_counts.index, y=rating_counts.values, hue=rating_counts.index)
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.show()
```



Insights:

The bar chart illustrates the distribution of ratings for the content available on Disney+. It reveals that "TV-G" (suitable for all audiences) is the most prevalent rating category, followed by "TV-PG" (parental guidance suggested). The ratings "G" and "PG" also have a significant presence, indicating a focus on family-friendly content. Ratings like "TV-Y7", "TV-Y", and "TV-Y7-FV" (intended for children) have a moderate presence, while "TV-14" and "PG-13" have a lower occurrence, suggesting a limited number of titles suitable for older audiences. Lastly, a small portion of the content is rated "TV-Y" (suitable for all children) and a very small fraction is labeled "Unknown".

```
# Top 10 genres
df['genre'] = df['listed_in'].str.split(', ')
genres = df['genre'].explode().value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=genres.values, y=genres.index, hue=genres.index)
plt.title('Top 10 Genres')
plt.xlabel('Number of Titles')
plt.ylabel('Genre')
plt.show()
```



Insights:

The bar chart reveals that "Family" is the dominant genre on Disney+, significantly outnumbering other categories. This aligns with Disney's reputation as a family-friendly entertainment provider. "Animation" and "Comedy" follow closely, indicating a strong focus on animated content and lighthearted entertainment. "Action-Adventure" also has a substantial presence, broadening the appeal to a wider audience. Genres like "Animals & Nature," "Coming of Age," "Fantasy," "Documentary," "Kids," and "Drama" are represented but with lower frequencies. Overall, the chart highlights Disney+'s emphasis on family-oriented content, with animation, comedy, and action-adventure as key areas of focus.

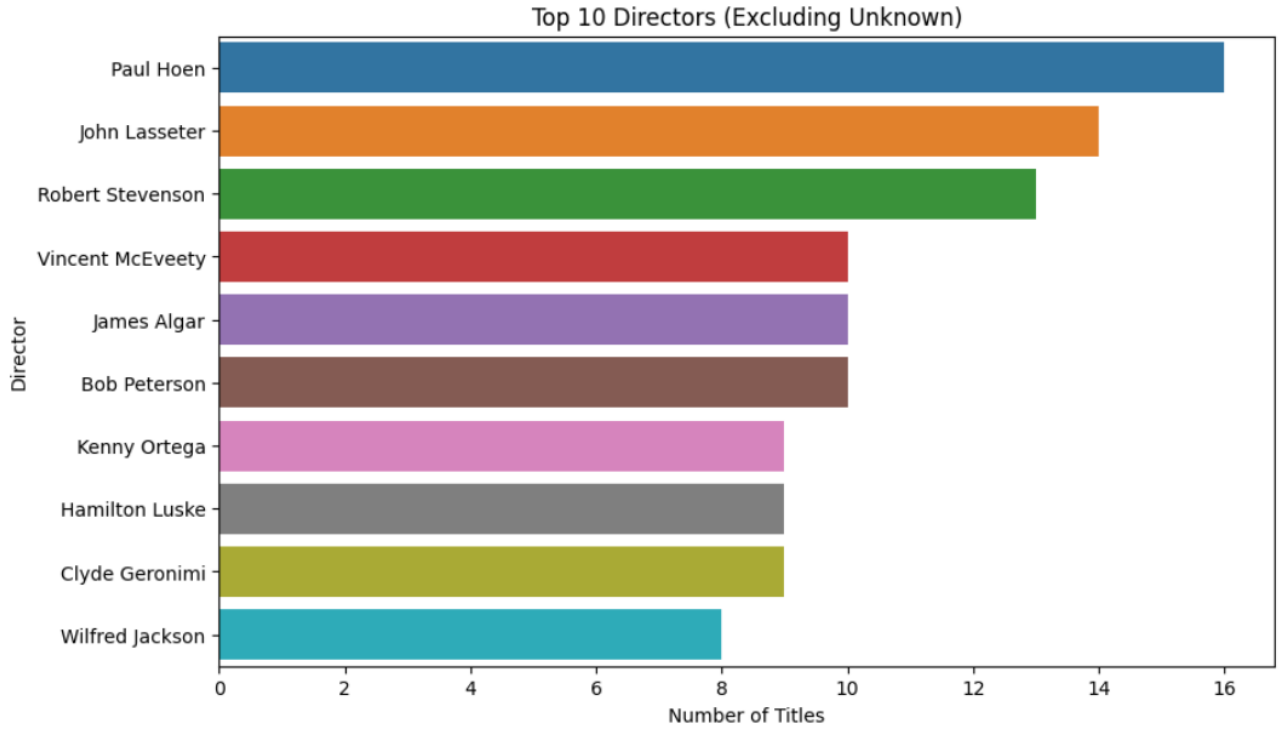
```
[14] # Handling 'Unknown' in director and cast columns
     # Filter out 'Unknown' entries from director and cast columns
     df_cleaned = df[df['director'] != 'Unknown']
     df_cleaned = df_cleaned[df_cleaned['cast'] != 'Unknown']
```

```
# Top 10 directors after cleaning
directors = df_cleaned['director'].str.split(', ').explode().value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=directors.values, y=directors.index, hue=directors.index)
plt.title('Top 10 Directors (Excluding Unknown)')
plt.xlabel('Number of Titles')
plt.ylabel('Director')
plt.show()
```

Insights:

The bar chart presents the top 10 directors in the Disney+ dataset, excluding those listed as "Unknown." Paul Hoen emerges as the most prolific director, having directed a significant number of titles on the platform. John Lasseter and Robert Stevenson follow closely, highlighting their substantial contributions to Disney's content library. Directors like Vincent McEveety, James Algar, and Bob Peterson also have a notable presence, indicating their significant roles in creating Disney+ content.

The chart reveals a diverse range of directors with varying levels of contribution, providing valuable insights into the creative forces behind the platform's content.

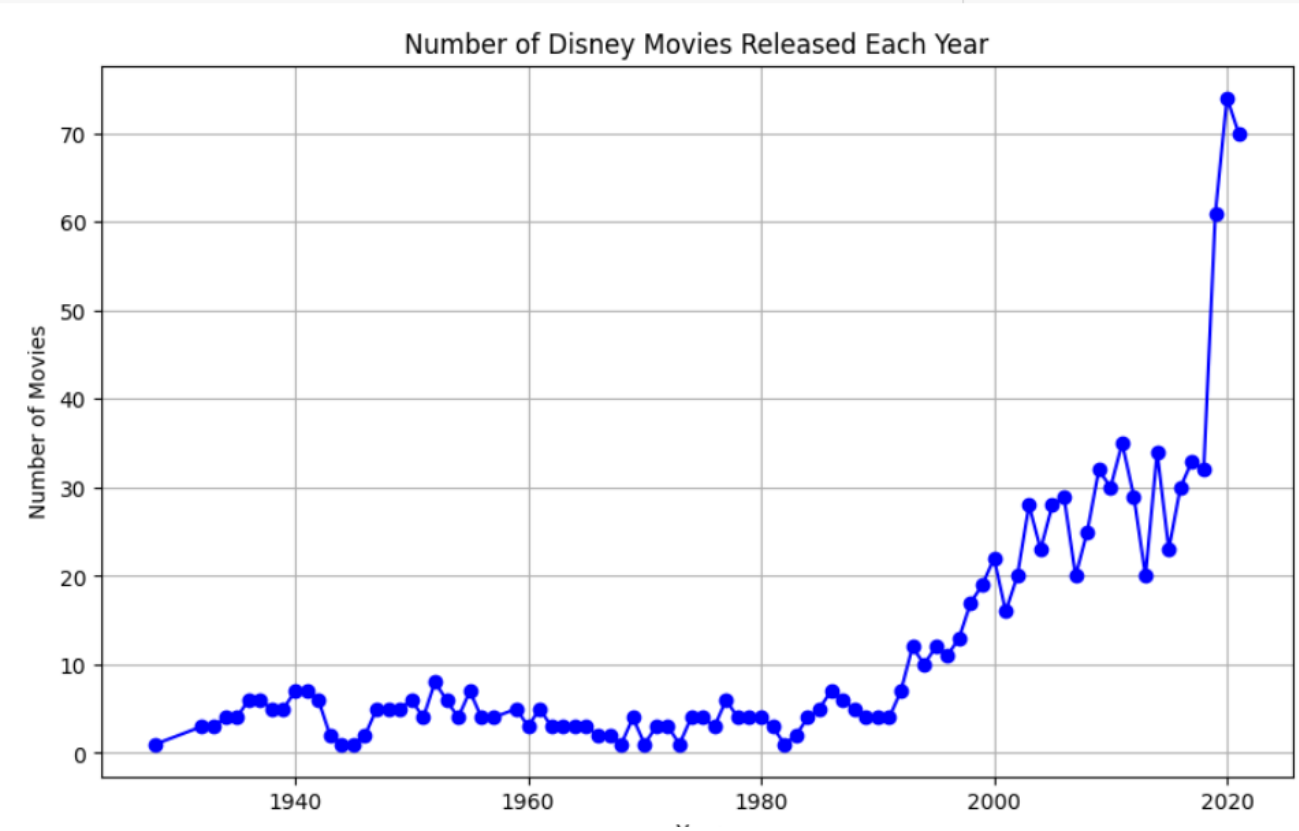



```
# Movies released each year
movies_per_year = df[df['type'] == 'Movie'].groupby('release_year')['title'].count()
plt.figure(figsize=(10, 6))
plt.plot(movies_per_year, marker='o', color='b')
plt.title('Number of Disney Movies Released Each Year')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.grid()
plt.show()
```

Insights:

The line graph depicts the number of Disney movies released each year. It reveals a fluctuating trend with periods of high and low output. The graph shows a significant increase in the number of Disney movies released in recent years, particularly after 2010. This suggests a recent surge in Disney's movie production activity, likely driven by factors such as the success of their franchises and the growing streaming market. The graph also shows a period of lower movie production in the mid-20th century,

followed by a gradual increase in output. This suggests that Disney's movie production has evolved over time, with periods of higher and lower activity.

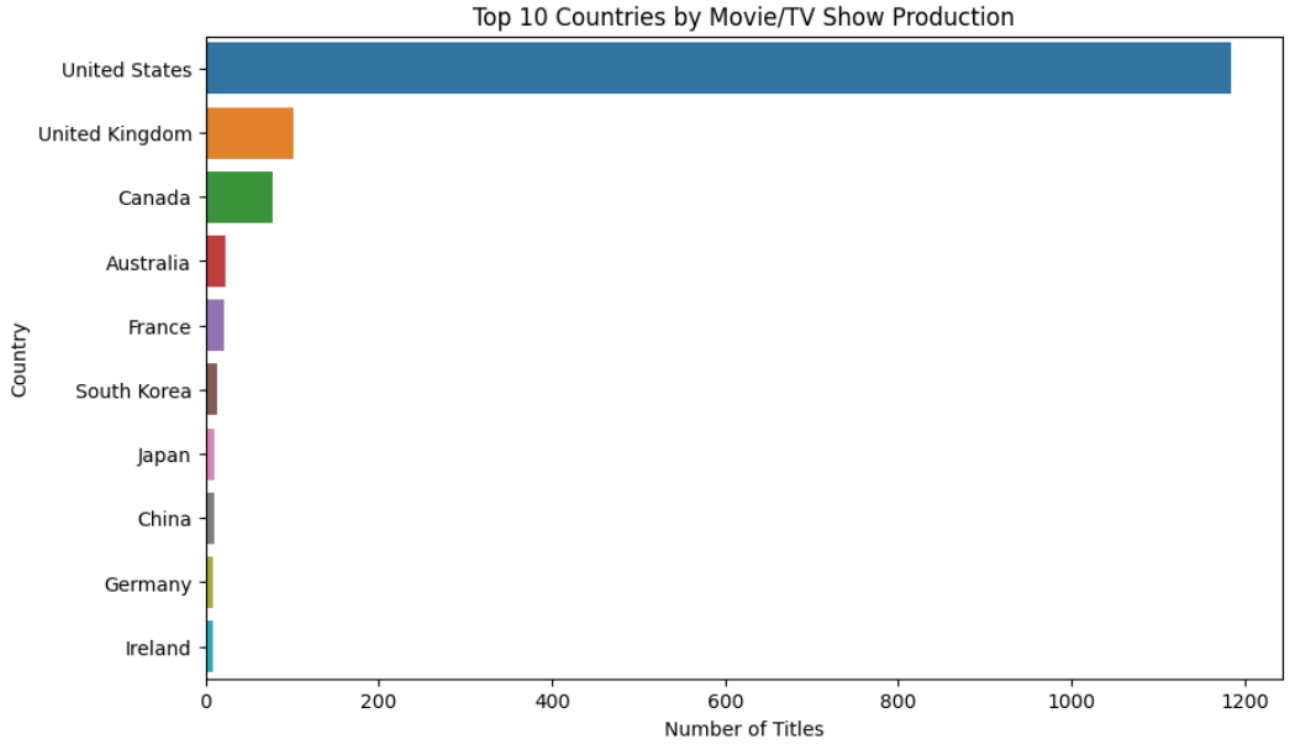



```
# Top countries by production
countries = df['country'].str.split(', ').explode().value_counts()
top_countries = countries[countries.index != 'Unknown'].head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_countries.values, y=top_countries.index, hue=top_countries.index)
plt.title('Top 10 Countries by Movie/TV Show Production')
plt.xlabel('Number of Titles')
plt.ylabel('Country')
plt.show()
```

Insights:

The bar chart clearly demonstrates that the United States is the dominant country in terms of movie and TV show production for Disney+. It significantly outpaces all other countries, producing a far greater number of titles. The United Kingdom follows as a distant second, with Canada, Australia, and

France also contributing a smaller, but still noticeable, number of productions. The remaining countries in the top 10 (South Korea, Japan, China, Germany, and Ireland) have considerably fewer productions compared to the top few. This distribution highlights the strong influence of American productions on Disney+ content, with some international collaborations and productions from other key countries.

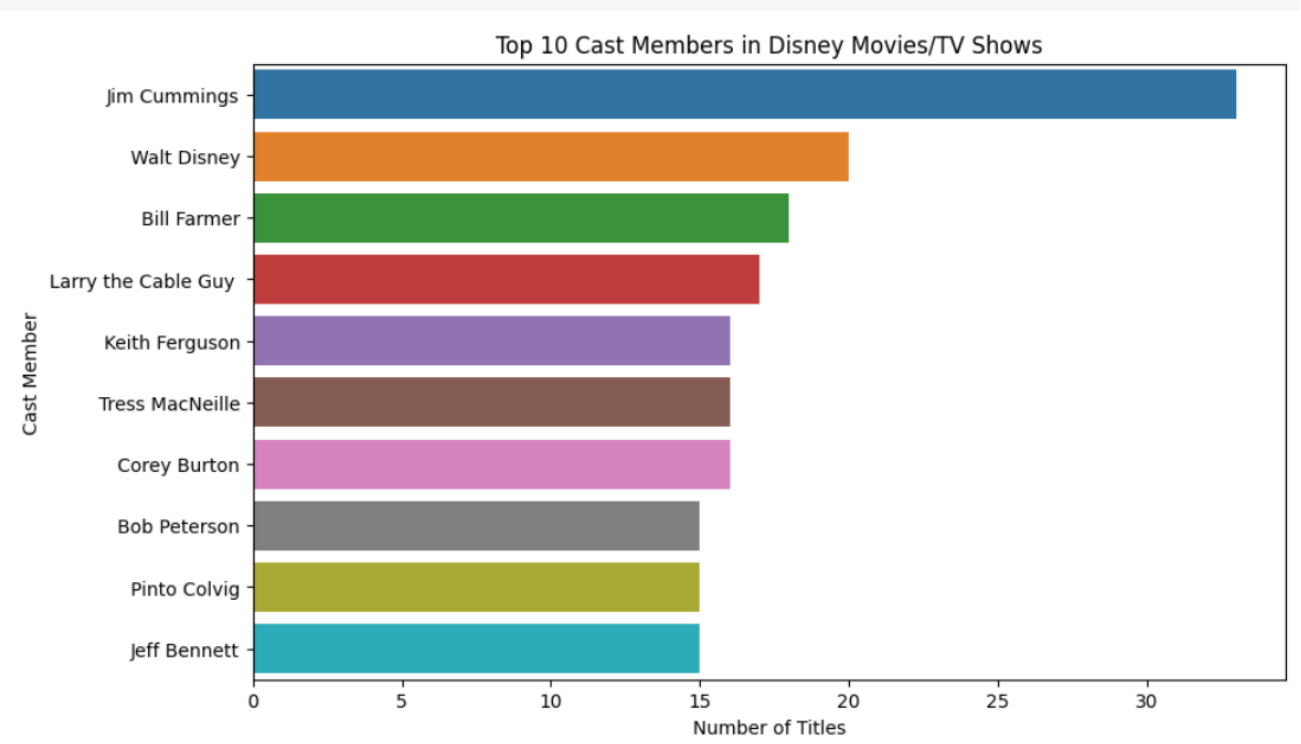



```
# Top 10 cast members
cast = df['cast'].str.split(', ').explode().value_counts()
top_cast = cast[cast.index != 'Unknown'].head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=top_cast.values, y=top_cast.index, hue=top_cast.index)
plt.title('Top 10 Cast Members in Disney Movies/TV Shows')
plt.xlabel('Number of Titles')
plt.ylabel('Cast Member')
plt.show()
```

Insights:

The bar chart presents the top 10 cast members appearing in Disney movies and TV shows, excluding entries where the cast is listed as "Unknown." Jim Cummings emerges as the most prolific cast

member, having appeared in a significantly higher number of titles compared to others. Walt Disney follows closely, likely due to his historical association with the studio.

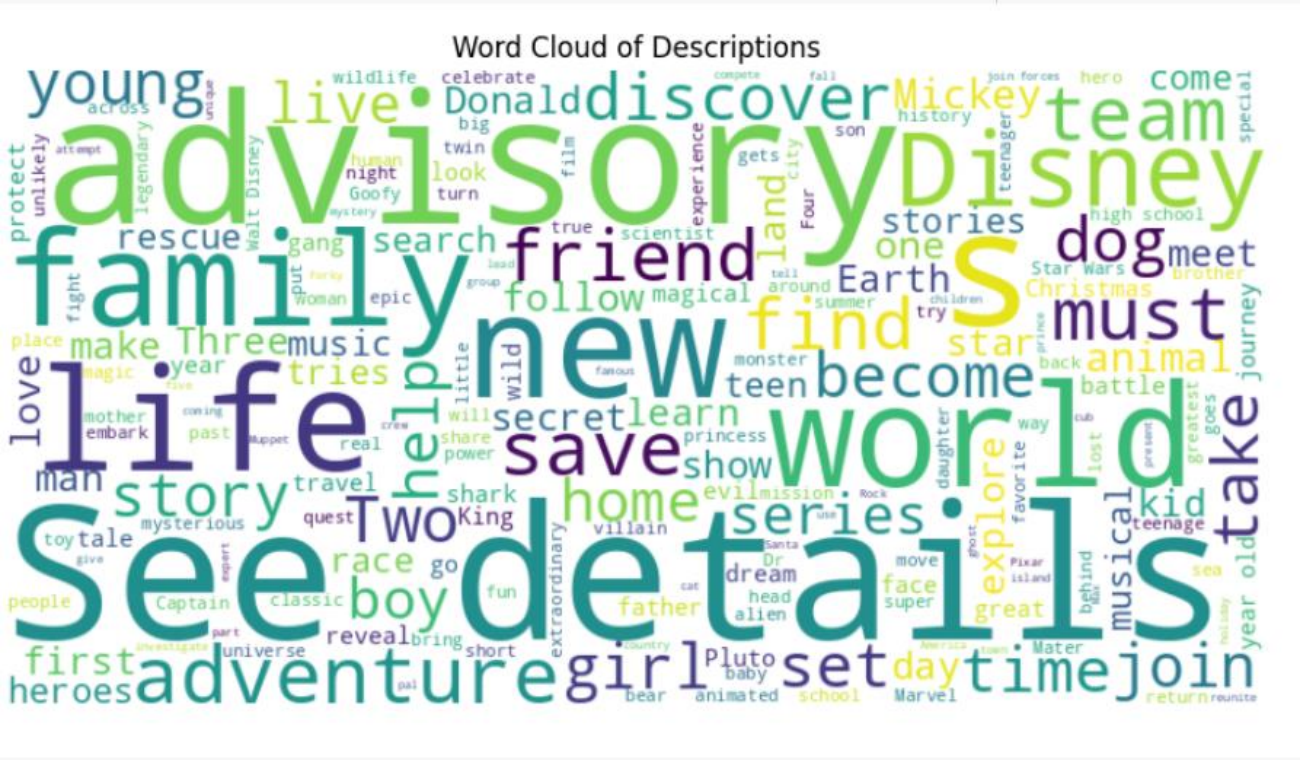



```
# WordCloud for descriptions
all_descriptions = ' '.join(df['description'])
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_descriptions)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Descriptions')
plt.show()
```

Insights:

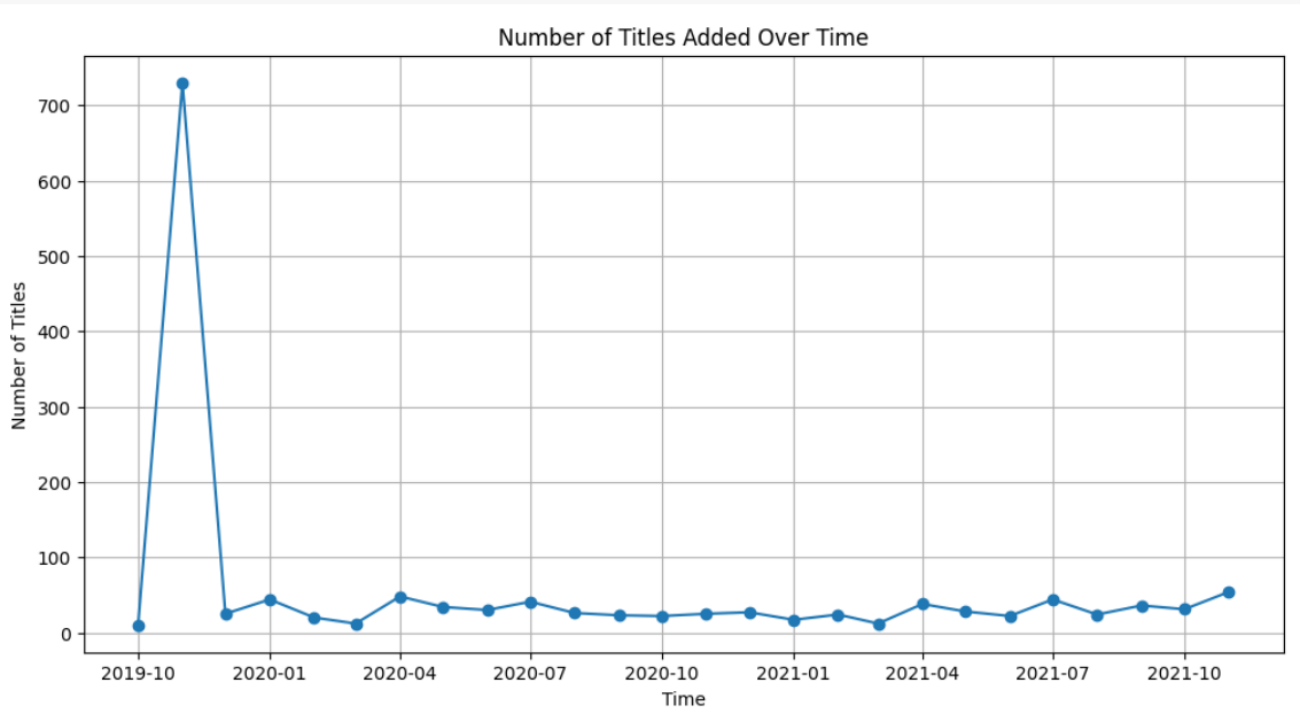
The word cloud visually represents the most frequent words used in the descriptions of content on Disney+. Words like "adventure," "Disney," "family," "new," and "world" appear prominently, highlighting the emphasis on themes of exploration, storytelling, and global appeal. Words like "friend," "love," and "journey" suggest narratives focused on relationships and personal growth. The presence of words like "music," "magical," and "star" indicates a focus on entertainment and fantasy.

Overall, the word cloud provides a quick visual summary of the key themes and narratives that are prevalent in the descriptions of Disney+ content.




```
[20] # Create a monthly frequency time series of titles added
     df['year_month'] = df['date_added'].dt.to_period('M').dt.to_timestamp()
     time_series = df['year_month'].value_counts().sort_index()
     time_series = time_series.asfreq('MS')
```

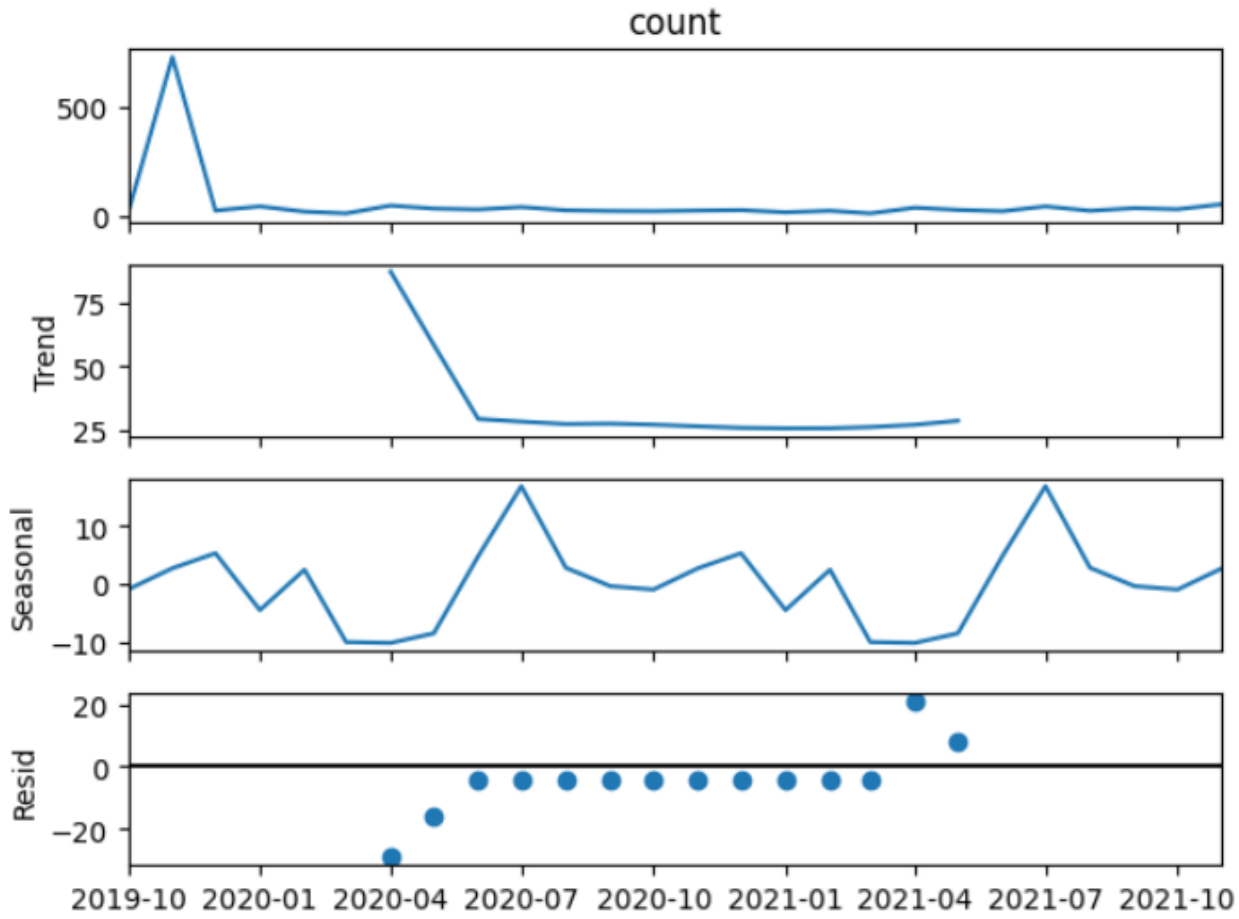
```
# Plot time series
plt.figure(figsize=(12, 6))
plt.plot(time_series, marker='o')
plt.title('Number of Titles Added Over Time')
plt.xlabel('Time')
plt.ylabel('Number of Titles')
plt.grid(True)
plt.show()
```



Insights:

The line graph illustrates the number of titles added to Disney+ over time. A striking observation is the massive spike in content additions around October 2019, coinciding with the platform's launch. This initial surge is followed by a sharp decline and then a period of relative stability with minor fluctuations. This pattern indicates a strategic focus on building a substantial content library at launch to attract subscribers, followed by a more consistent but less dramatic rate of content additions in subsequent months. The graph effectively visualizes the initial content push and the subsequent stabilization of content releases on Disney+.

```
# Decompose the time series
decomposition = seasonal_decompose(time_series, model='additive', period=12)
decomposition.plot()
plt.show()
```

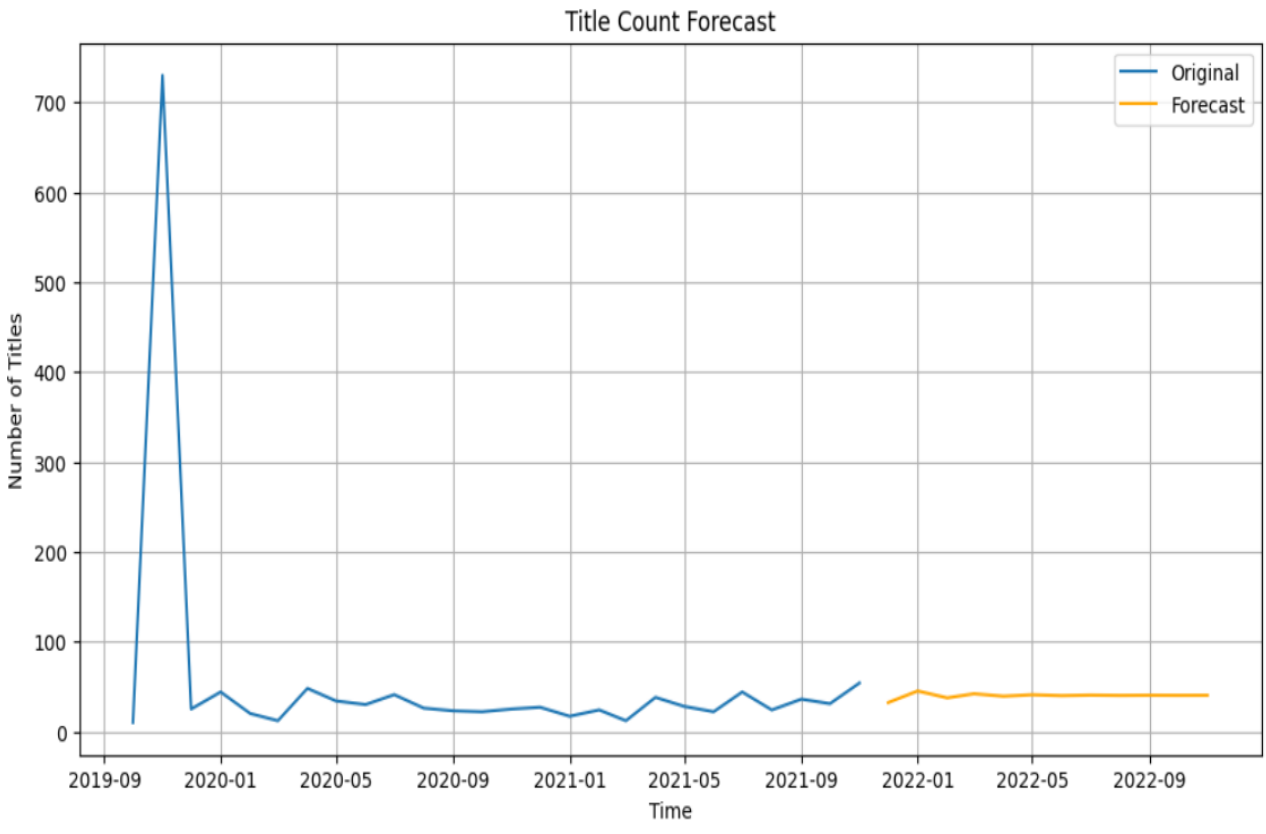


Insights:

The plot shows the results of a time series decomposition using the additive model. The top plot shows the original time series data, which exhibits a sharp peak followed by a decline and then a period of relatively stable fluctuations. The middle plot shows the trend component, which captures the overall upward or downward movement of the series. We can see a steep initial increase followed by a gradual decline. The bottom plot shows the seasonal component, which reveals any recurring patterns within the data. In this case, we observe some fluctuations that may be due to seasonal factors. Finally, the bottom plot shows the residuals, which represent the remaining variability in the data after accounting for the trend and seasonal components.

```
[23] # Forecast future trends using ARIMA
     model = ARIMA(time_series, order=(1, 1, 1))
     fit = model.fit()
     forecast = fit.get_forecast(steps=12)
     forecast_index = pd.date_range(start=time_series.index[-1] + pd.DateOffset(months=1), periods=12, freq='MS')
     forecast_series = pd.Series(forecast.predicted_mean, index=forecast_index)
```

```
# Plot original and forecasted time series
plt.figure(figsize=(12, 6))
plt.plot(time_series, label='Original')
plt.plot(forecast_series, label='Forecast', color='orange')
plt.title('Title Count Forecast')
plt.xlabel('Time')
plt.ylabel('Number of Titles')
plt.legend()
plt.grid(True)
plt.show()
```



Insights:

The chart displays the original time series of titles added to Disney+ along with a forecast for future title additions. The original data shows a large spike around the platform's launch in late 2019, followed by a period of relatively stable, low-level additions. The forecast, generated by an ARIMA model, projects a continued low and relatively flat number of title additions in the coming year. This suggests that Disney+ is not expected to drastically increase its rate of content additions in the near future, maintaining a more consistent, albeit lower, pace compared to its initial launch period.

```
[25] print("\nForecasted Data:")
     print(forecast_series)
```

```
Forecasted Data:
2021-12-01    32.171124
2022-01-01    45.106603
2022-02-01    37.441224
2022-03-01    41.983618
2022-04-01    39.291860
2022-05-01    40.886957
2022-06-01    39.941726
2022-07-01    40.501856
2022-08-01    40.169931
2022-09-01    40.366625
2022-10-01    40.250067
2022-11-01    40.319138
Freq: MS, Name: predicted_mean, dtype: float64
```

Insights:

The provided text shows the forecasted data for the number of titles added to Disney+ for the next 12 months, starting from December 2021. The forecast suggests a relatively stable range of additions, with the predicted number fluctuating between 32 and 45 titles per month. This indicates a consistent content release strategy from Disney+ without any major spikes or dips in the foreseeable future.

5.2.9 Outcome

The major project, focused on Time Series Analysis and Forecasting, successfully explored Disney+ content trends and provided actionable insights. A time series model was developed to analyze monthly content additions, revealing key patterns, including a **notable spike during the platform's launch in October 2019 and a subsequent period of stabilization**. Seasonal decomposition highlighted recurring trends and residual variations, while **ARIMA forecasting projected a steady rate of content additions over the next year**. These outcomes demonstrate the effective use of data analytics techniques for understanding platform strategies and future planning. The visualizations and analysis provide a clear and concise understanding of Disney+'s content release trends.

CHAPTER VI

CONCLUSION

CONCLUSION

The internship at **Main Flow Services and Technologies** has been an enriching experience, offering significant exposure to the practical applications of data analytics using Python. Through the completion of the **mini project** on sentiment analysis and the **major project** on time series analysis and forecasting, I have gained valuable insights into working with real-world datasets, exploring statistical and machine learning models, and deriving meaningful conclusions.

The mini project highlighted the importance of various classification models, such as **VADER, Logistic Regression, Bernoulli Naive Bayes, and Linear SVC**, in analyzing sentiment. Each model provided unique strengths and demonstrated their ability to process large datasets, with **Logistic Regression and Linear SVC achieving the highest accuracies**. This project not only honed my skills in text analysis but also reinforced the importance of model evaluation metrics like precision, recall, and F1-score.

The major project on Disney+ dataset analysis provided comprehensive insights into user engagement, content distribution, and trends over time. By employing visualization techniques, statistical analyses, and time series forecasting with **ARIMA**, I was able to identify patterns, uncover critical business insights, and predict future trends. This project showcased the power of data-driven decision-making in understanding consumer behavior and optimizing content strategies.

Overall, the internship has solidified my knowledge of Python for data analytics, enhanced my proficiency with libraries such as **pandas, numpy, matplotlib, seaborn, scikit-learn, and statsmodels**, and provided hands-on experience in leveraging data to solve complex problems. It has further strengthened my ability to interpret results, document findings, and present actionable insights effectively. This experience has been instrumental in preparing me for future endeavors in data analytics and related fields.

CHAPTER VII

FUTURE WORKS

FUTURE WORKS

While the internship provided valuable insights and accomplishments in data analytics using Python, there is scope for further exploration and improvements. Future work in this area could include the following:

1) Enhanced Sentiment Analysis Techniques

- Experimenting with advanced natural language processing (NLP) models such as **BERT, GPT, or Transformer-based** architectures to improve the accuracy of sentiment analysis.
- Exploring feature engineering techniques and fine-tuning hyperparameters for the models used.
- Expanding the analysis to include multilingual sentiment analysis for broader applicability.

2) Improved Time Series Forecasting Models

- Implementing and comparing advanced forecasting models like **Prophet, LSTM, or GRU** for improved performance and accuracy in predicting future trends.
- Incorporating external factors (e.g., events, holidays, or marketing campaigns) into the forecasting models to enhance predictive capabilities.

3) Automation of Analytics Pipelines

- Developing end-to-end data pipelines using tools like **Apache Airflow** or **Luigi** for automation and scalability.
- Deploying machine learning models using frameworks like **Flask**, **FastAPI**, or **Streamlit** for interactive and real-time analytics dashboards.

CHAPTER VIII

REFERENCES

REFERENCES

[1] Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, *8*(1), 216-225. Retrieved from https://doi.org/10.1609/icwsm.v8i1.14550

[2] Rish, I. (2001). *An Empirical Study of the Naive Bayes Classifier*. Proceedings of the 14th International FLAIRS Conference, 312-319.

[3] Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks*. Machine Learning, 20(3), 273-297. https://doi.org/10.1007/BF00994018

[4] Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control* (Revised Edition). Holden-Day. ISBN: 978-0816211043.

[5] Disney+ Official Website. (n.d.). *About Disney+ and its Content*. Retrieved from https://www.disneyplus.com/

[6] Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts. Retrieved from https://otexts.com/fpp3/

[7] Hamborg, Felix; Donnay, Karsten (2021). "NewsMTSC: A Dataset for (Multi-)Target-dependent Sentiment Classification in Political News Articles". "Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume"

[8] Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar (2002). "Thumbs up? Sentiment Classification using Machine Learning Techniques"

[9] Vryniotis, Vasilis (2013). *The importance of Neutral Class in Sentiment Analysis*

[10] Bandaru, D. S. S. G. (2025). *Codebase for Main Flow Services and Technologies Internship Projects*. GitHub Repository. Available at: https://github.com/deva-04/MainFlow-Internship