Applied Project Report
Analysis of Real-life Business Cases

By

Arijit Dey

A Master’s Project Report submitted to Scaler Neovarsity - Woolf in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

August, 2024

Scaler Mentee Email ID : arijit.aurko@icloud.com
Thesis Supervisor : Shivank Agarwal
Date of Submission : 05/08/2024

Certification

I confirm that I have overseen / reviewed this applied project and, in my judgment, it adheres to the appropriate standards of academic presentation. I believe it satisfactorily meets the criteria, in terms of both quality and breadth, to serve as an applied project report for the attainment of Master of Science in Computer Science degree. This applied project report has been submitted to Woolf and is deemed sufficient to fulfill the prerequisites for the Master of Science in Computer Science degree.

Shivank Agarwal

…………………

Project Guide / Supervisor

DECLARATION

I confirm that this project report, submitted to fulfill the requirements for the Master of Science in Computer Science degree, completed by me from **December 1, 2022** to **June 30, 2024**, is the result of my own individual endeavor. The Project has been made on my own under the guidance of my supervisor with proper acknowledgement and without plagiarism. Any contributions from external sources or individuals, including the use of AI tools, are appropriately acknowledged through citation. By making this declaration, I acknowledge that any violation of this statement constitutes academic misconduct. I understand that such misconduct may lead to expulsion from the program and/or disqualification from receiving the degree.

Arijit Dey

Arijit Dey

<Signature of the Candidate> **Date:** August 5, 2024

ACKNOWLEDGMENT

I would like to express my sincere gratitude to everyone who has contributed to the successful completion of this project, which explores the application of advanced data analytics, machine learning, and artificial intelligence in diverse business domains.

First and foremost, I extend my deepest thanks to my project advisor and mentor **Mr. Shivank Agarwal** for his invaluable guidance and support throughout his endeavor. His expertise and insights have been instrumental in shaping the direction and depth of our research.

I am grateful to the companies and organizations that provided us with the necessary datasets and practical insights. The access to real-world data and industry-specific challenges has been crucial in developing and testing our models and methodologies.

I also acknowledge the contributions of our technical team, whose dedication and hard work in data preprocessing, model development, and evaluation have been key to the project's success. Their technical acumen and problem-solving skills have been critical in overcoming the challenges encountered during the project.

Special thanks go to our Scaler Instructors **Mr. Subhodeep Dey** and **Mr. Vidya Sagar N**, who provided the necessary resources and facilities to support this research. The learning environment and academic support have been vital in enabling us to pursue this project with rigor and depth.

Lastly, I extend our appreciation to our peers and colleagues for their constructive feedback and collaboration. Their critical evaluations and discussions have enriched my work and helped me refine our approaches.

This project has been a collaborative effort, and I am grateful for the collective contributions that have made it possible. I hope that the findings and insights derived from this work will contribute to the advancement of knowledge and practical applications in the fields of digital advertising, logistics, supply chain management, financial literacy, and content analysis.

Thank you all for your support and encouragement.

Table of Contents

List of Figures

(List of Images, Graphs, Charts sequentially as they appear in the text)

Applied Real-life Business Cases

ABSTRACT

This collection of the five business case studies explores the application of advanced data analytics, machine learning, and artificial intelligence in diverse industry sectors. The first study focuses on optimizing digital advertising through forecasting page views for 145k Wikipedia pages, leveraging daily view count data to strategically place ads and maximize client engagement. The second study centers on India's largest intra-city logistics provider, aiming to estimate delivery times for food orders by developing a regression model that factors in order details, pickup locations, and delivery partner availability. The third study examines a fresh produce supply chain company, which seeks to automate the classification of vegetables using a robust multiclass classifier, distinguishing between various vegetables and identifying noise in market scene images. The fourth study delves into a fintech company that uses AI and ML to enhance financial literacy among Indians, classifying news articles into categories such as politics, technology, and entertainment using natural language processing techniques. Lastly, the fifth study investigates a microblogging platform's initiative to automatically tag and analyze tweets through Named Entity Recognition (NER), facilitating better understanding of trends and topics without reliance on user-generated hashtags.

These studies collectively demonstrate the transformative potential of AI and machine learning in optimizing business processes, enhancing customer experiences, and enabling data-driven decision-making. The methodologies employed across these cases, including time series forecasting, regression analysis, image classification, text categorization, and entity recognition, showcase the versatility of these tools in addressing complex business challenges and creating value.

Chapter 1 : Business Case Study 1 — **Ad and Marketing based Company**

Problem Description

An advertisement and marketing based company is helping businesses elicit maximum clicks at minimum cost. The company is an advertisement infrastructure to help businesses promote themselves easily, effectively, and economically. The interplay of 3 AI modules — Design, Dispense, and Decipher, come together to make it this an end-to-end 3 step process digital advertising solution for all.

As a Data Scientist, we are trying to understand the per page view report for different wikipedia pages for 550 days, and forecasting the number of views so that you can predict and optimize the ad placement for our clients. We are provided with the data of 145k wikipedia pages and daily view count for each of them. Their respective clients belong to different regions and need data on how their ads will perform on pages in different languages.

Business Questions to be answered from Analysis

The following business questions can be addressed from the given problem statement and the corresponding analysis.

Performance Forecasting:

- How can we accurately forecast the number of views for each Wikipedia page over the next 30, 60, or 90 days?
- What trends can we identify in the page views that can inform future ad placement strategies?

Ad Placement Optimization:

- Which Wikipedia pages are expected to receive the highest number of views in the upcoming period, and thus, are the best candidates for ad placement?
- How can we optimize ad placements on Wikipedia pages to maximize client engagement and click-through rates?

Regional and Language Analysis:

- How do page view trends vary across different languages and regions?
- Which regions or language-specific Wikipedia pages are likely to yield the highest ad engagement for our clients?

Client-Specific Insights:

- For clients targeting specific demographics or regions, which Wikipedia pages should be prioritized for ad placement?
- How can we tailor ad strategies for clients based on the predicted performance of Wikipedia pages in their target regions?

Cost Efficiency:

- How can we minimize the cost of ad placements while maximizing visibility and engagement for our clients?
- What is the cost-per-click (CPC) effectiveness on different Wikipedia pages, and how can it be improved?

Impact of Design and Dispense Modules:

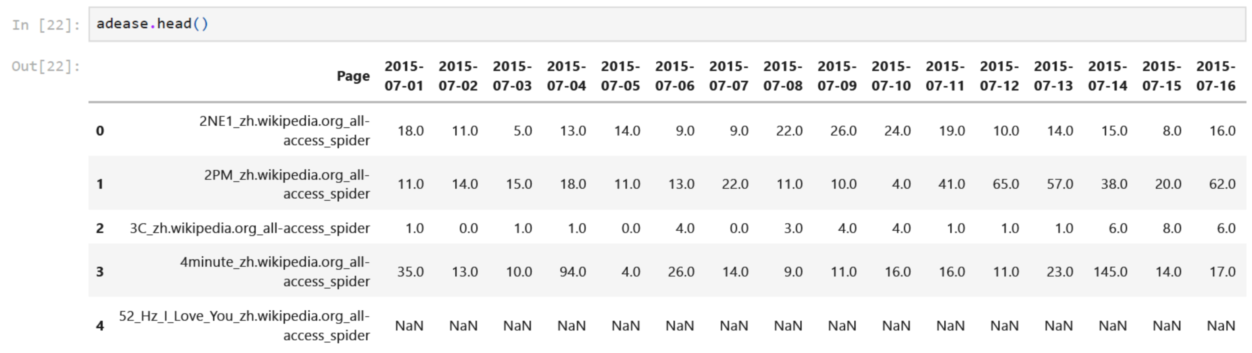
- How do different design elements of ads impact the click-through rates on various Wikipedia pages?
- What is the optimal frequency and timing for dispensing ads to maximize their impact on Wikipedia pages?

Deciphering User Behavior:

- What patterns can be observed in user behavior based on the historical data of Wikipedia page views?
- How do external factors (e.g., current events, seasonal trends) influence the view count of Wikipedia pages and, consequently, the performance of ads?

Analysis

The snapshot of the data shared by the respective Ad and Marketing Based Company looks like this:



```
In [22]: adease.head()
```

Out[22]:

	Page	2015-07-01	2015-07-02	2015-07-03	2015-07-04	2015-07-05	2015-07-06	2015-07-07	2015-07-08	2015-07-09	2015-07-10	2015-07-11	2015-07-12	2015-07-13	2015-07-14	2015-07-15	2015-07-16
0	2NE1_zh.wikipedia.org_all-access_spider	18.0	11.0	5.0	13.0	14.0	9.0	9.0	22.0	26.0	24.0	19.0	10.0	14.0	15.0	8.0	16.0
1	2PM_zh.wikipedia.org_all-access_spider	11.0	14.0	15.0	18.0	11.0	13.0	22.0	11.0	10.0	4.0	41.0	65.0	57.0	38.0	20.0	62.0
2	3C_zh.wikipedia.org_all-access_spider	1.0	0.0	1.0	1.0	0.0	4.0	0.0	3.0	4.0	4.0	1.0	1.0	1.0	6.0	8.0	6.0
3	4minute_zh.wikipedia.org_all-access_spider	35.0	13.0	10.0	94.0	4.0	26.0	14.0	9.0	11.0	16.0	16.0	11.0	23.0	145.0	14.0	17.0
4	52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 1.1: Time-Series Data provided by the Ad and Marketing Based Company

- Basic Exploratory Data Analysis

1. **No. of Pages per Each Language**

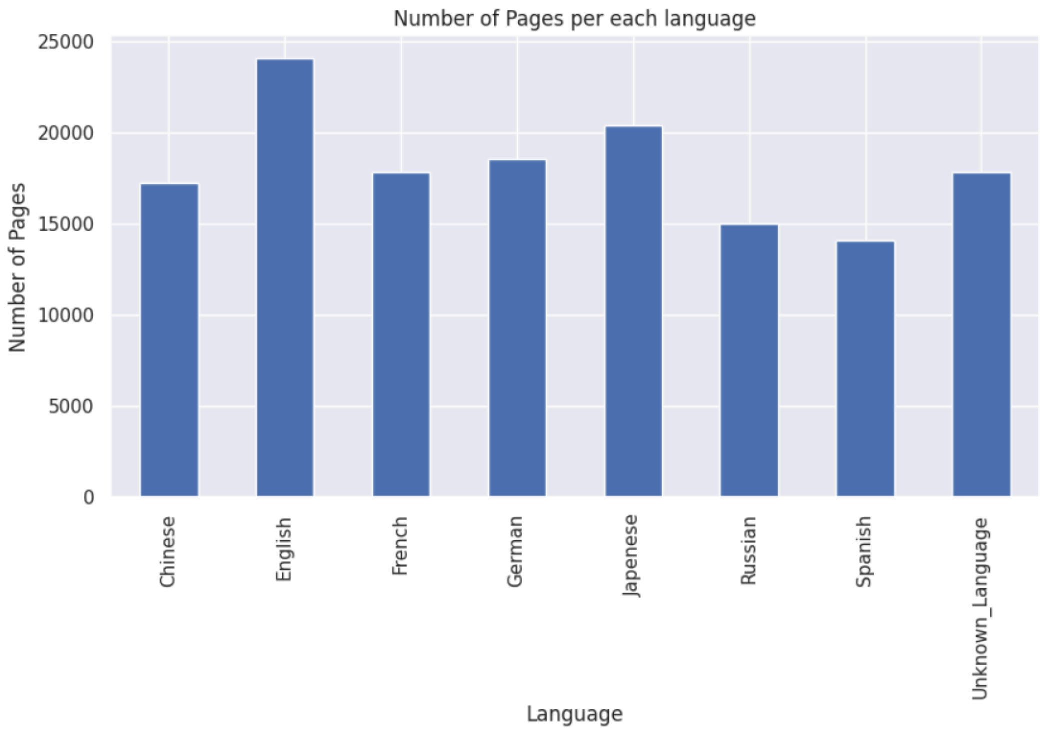
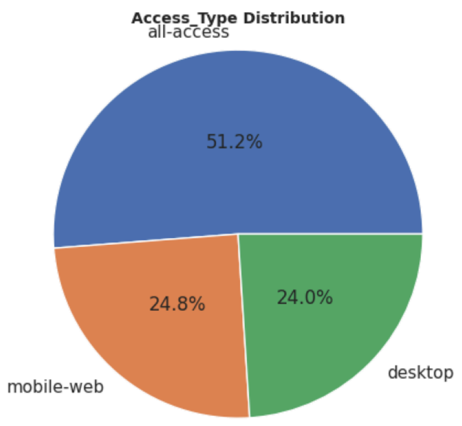


Figure 1.2: Barplot Showing Number of Pages for Each Language

2. Access-Type and Access-Origin Distribution



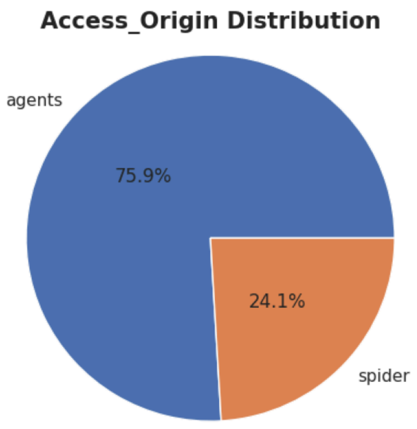


Figure 1.3: Pie-Chart Showing Access-Type and Access-Origin Distribution

3. Visualizing Time-Series for Each Languages

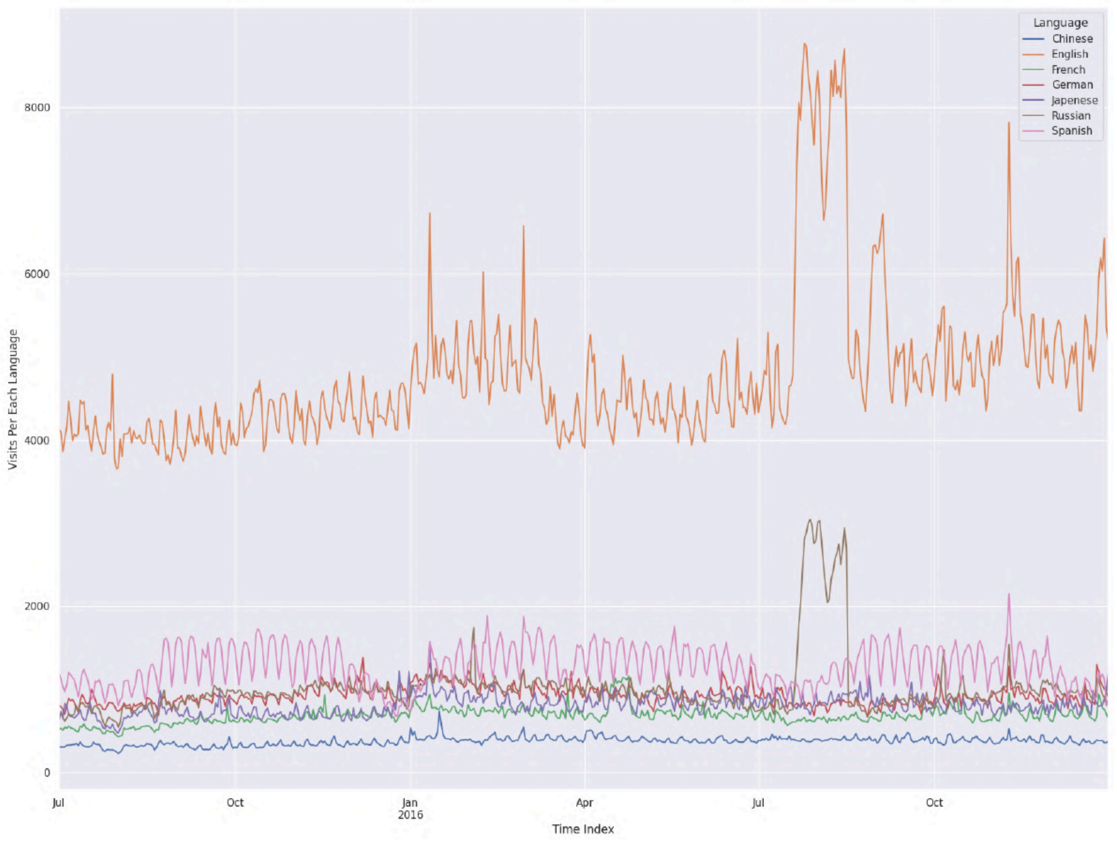


Figure 1.4: Time-Series Visualization for Visits for Each Language

- **Dickey-Fuller Test** to Check Stationarity of Time Series

It is a Hypothesis Testing to check if Time Series is Stationary or Trending.

a. **Null Hypothesis:** The series is **Non-Stationary**
b. **Alternative Hypothesis:** The series is **Stationary**
c. Significance Level (α) = 0.05
   i. p-value > α:
      we **fail to reject** Null Hypothesis, i.e. Time Series is Non-Stationary
   ii. p-value <= α:
      we **reject** Null Hypothesis, i.e., Time Series is Stationary

Results:

```
Chinese                                    German
Time Series is NOT Stationary.             Time Series is NOT Stationary.
p-value is: 0.218                          Japanese
                                           Time Series is Stationary.
English                                    p-value is: 0.0485
Time Series is NOT Stationary.
p-value is: 0.1059                         Russian
                                           Time Series is Stationary.
French                                     p-value is: 0.0015
Time Series is Stationary.
p-value is: 0.0274                         Spanish
                                           Time Series is Stationary.
                                           p-value is: 0.0333
p-value is: 0.1196
```

Based on Dickey-Fuller test of Stationarity, we can observe:

1. Spanish and Russian languages Pages visits Time series are **Stationary**.
2. Chinese, English, German, Japanese and French are **Not Stationary**.

- **Autocorrelation plot** to Verify the Periodicity of Time-Series

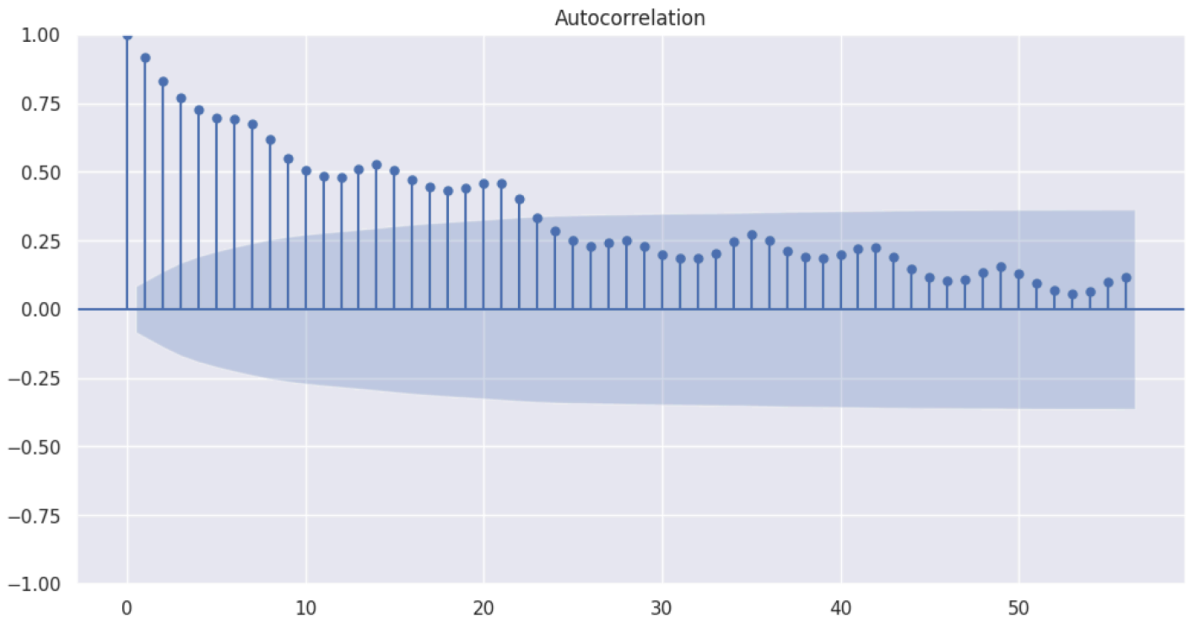


Figure 1.5: Autocorrelation Plot of the Entire Time-Series

- **Time-Series Decomposition** to Observe Trend, Seasonality and Residuals

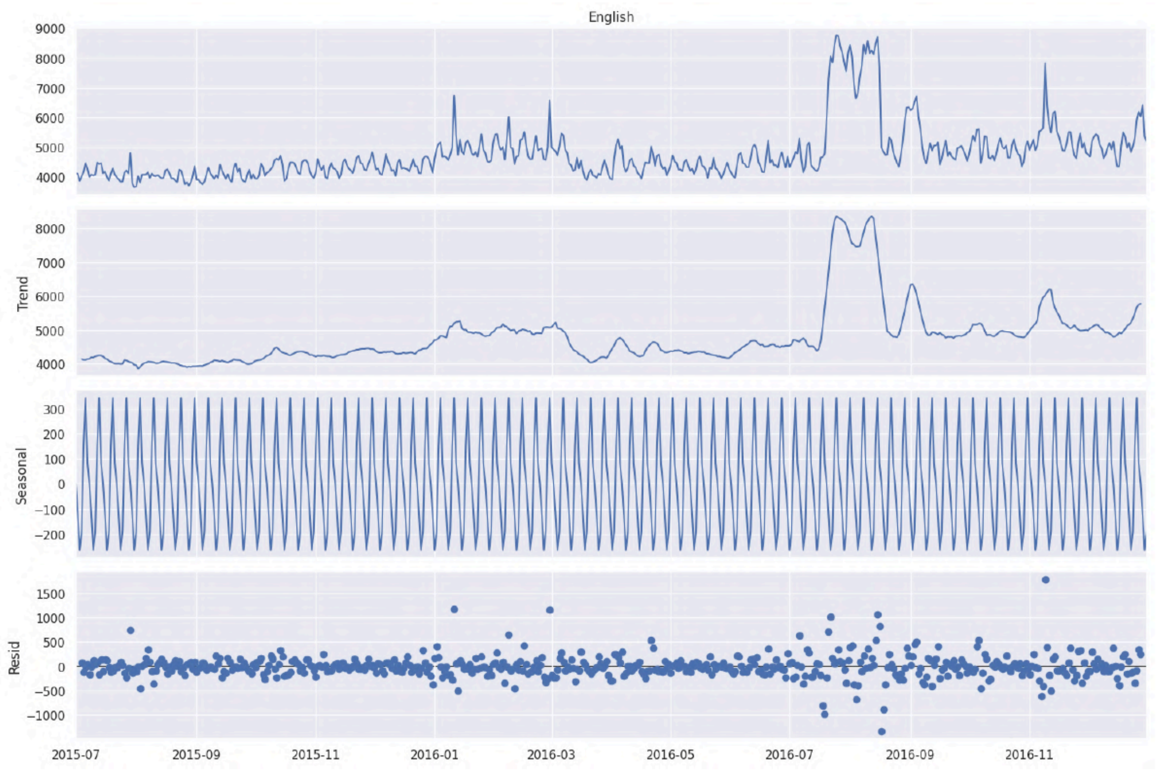


Figure 1.6: Time-Series Decomposition for Trend, Seasonality and Residuals

- **Time-Series Forecasting**

1. **Exponential Smoothing**

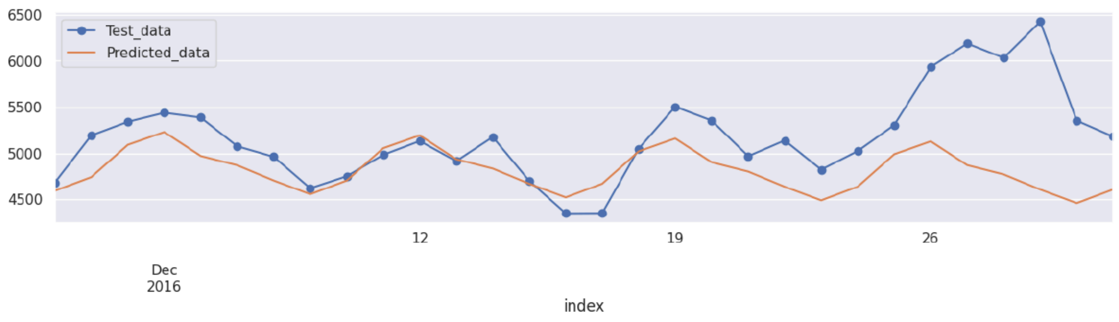


Figure 1.7: Exponential Smoothing for Test and Predicted Data

Mean Absolute Error (MAE): **406.676**
Root Mean Square Error (RMSE): **583.974**
Mean Absolute Percentage Error (MAPE): **0.074**

2. **Autoregressive Integrated Moving Average (ARIMA)**

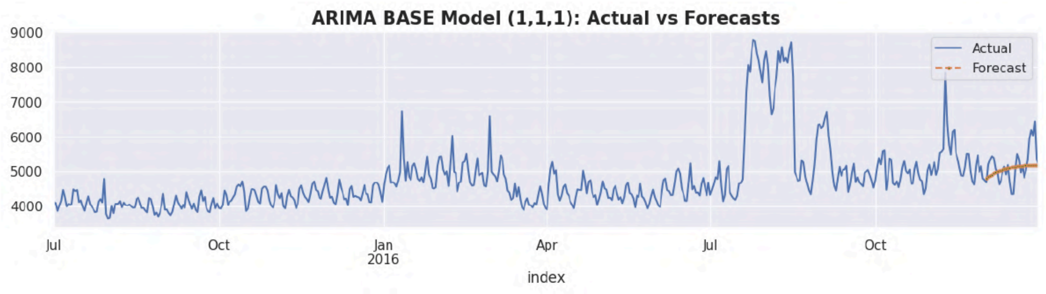


Figure 1.8: ARIMA Base Model

Mean Absolute Percentage Error (MAPE): **0.06645**
Root Mean Square Error (RMSE): **474.868**

3. Seasonal Autoregressive Integrated Moving Average Exogenous (SARIMAX)

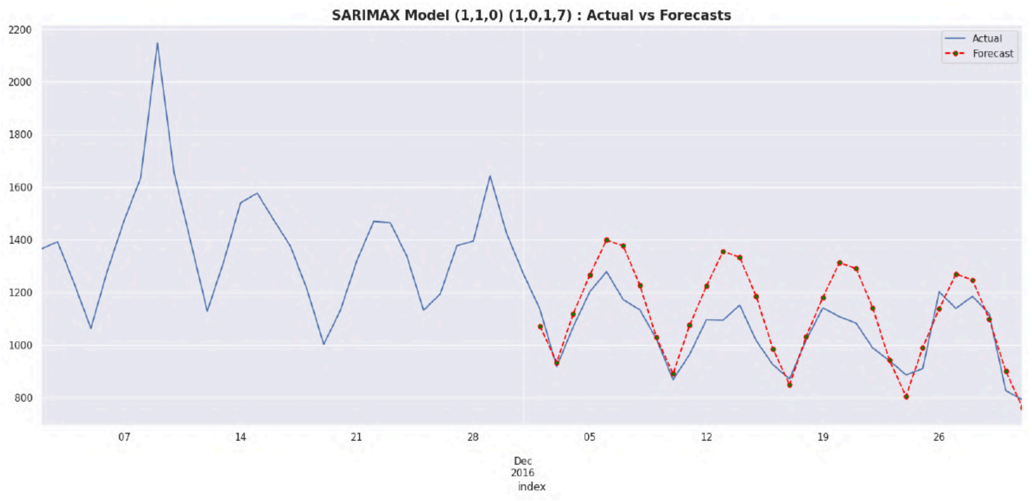


Figure 1.9: SARIMAX model for Spanish Time Series

SARIMAX model for Spanish Time Series
Parameters of Model: **(1,1,0) (1,0,1,7)**
Mean Absolute Percentage Error (MAPE) of Model: **0.08565**
Root Mean Square Error (RMSE) of Model: **114.648**

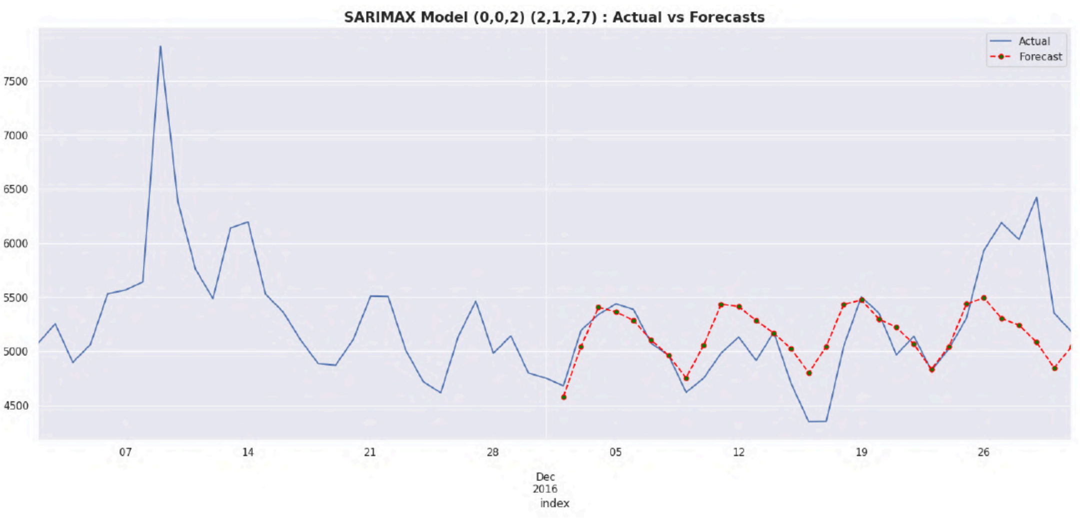


Figure 1.10: SARIMAX model for English Time Series

SARIMAX model for English Time Series
Parameters of Model: **(0,0,2) (2,1,2,7)**
Mean Absolute Percentage Error (MAPE) of Model: **0.05379**
Root Mean Square Error (RMSE) of Model: **419.714**

4. Prophet:

a. With Exogenous Regressor:

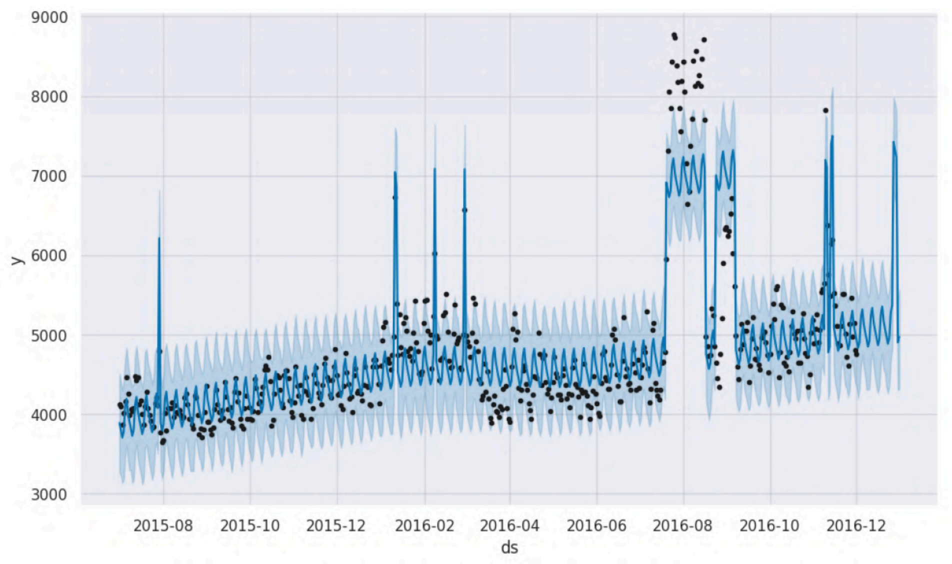


Figure 1.11: Prophet Model with Exogenous Regressor

b. Actual vs. Forecasts:

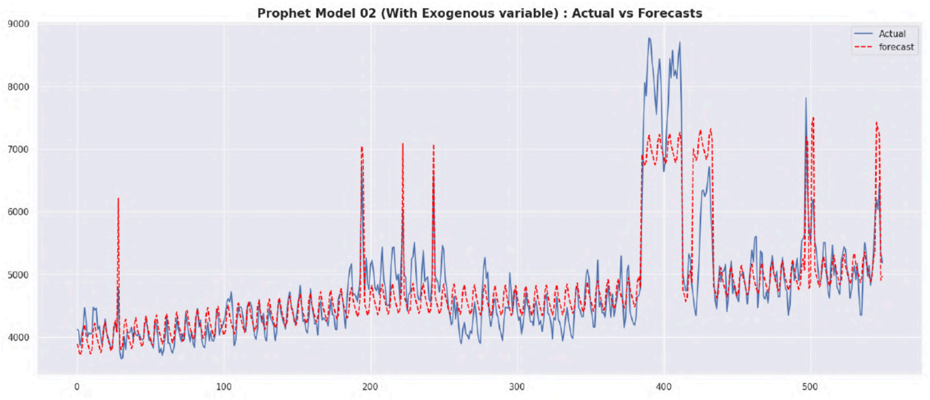


Figure 1.12: Prophet Model: Comparison with Actual vs. Forecasts

Mean Absolute Percentage Error (MAPE): **0.059**

Insights and Recommendations

- **Business Insights**

  1. **Language-Based Engagement:**
     - While English pages dominate in number, the presence of six other languages indicates a diverse user base. Even though they may not have as many pages, these other languages likely cater to specific regional or linguistic groups. This diversity suggests opportunities for targeted marketing and localized content to engage non-English-speaking users.
  2. **Device Usage Trends:**
     - The data indicates a nearly equal distribution between mobile-web (24.7748%) and desktop (23.9958%) access types, with a slight dominance of all-access (51.2295%). This reflects a balanced usage of different devices, suggesting that users are accessing content from a variety of platforms. Businesses should note the significant share of mobile-web users, which aligns with the global trend of increasing mobile device usage.
  3. **Human vs. Automated Traffic:**
     - With 75.932526% of traffic coming from human users (agents) and 24.067474% from spiders (automated systems), it's clear that a substantial portion of interactions are genuine. However, the relatively high percentage of automated traffic indicates the need for vigilance in distinguishing between real user engagement and bot activity, which can skew analytics and ad effectiveness measurements.
  4. **Opportunity for Multi-Language Content:**
     - The presence of seven languages suggests that there's a potential audience for multilingual content. Catering to these different language groups can help increase engagement and expand market reach. This is particularly relevant for international businesses or those targeting multilingual regions.
  5. **Ad Placement Strategy:**
     - Given the significant use of all-access and the comparable use of mobile-web and desktop, an effective ad placement strategy should consider the strengths and limitations of each platform. Mobile-friendly ads are essential, but maintaining a strong desktop presence is equally important, especially for more detailed or interactive ad formats.

- **Business Recommendations**

  1. **Expand Multilingual Advertising:**
     - Develop and implement advertising campaigns in the six other languages identified in the dataset. Localized advertisements can cater to the cultural and linguistic preferences of these user groups, potentially leading to higher engagement and conversion rates. Ensure that translations and cultural adaptations are accurate and resonate with the target audience.
  2. **Optimize for Mobile and Desktop:**
     - Create responsive ad designs that work seamlessly across both mobile-web and desktop platforms. Given the significant usage of both, ads should be optimized for different screen sizes and user experiences. For instance, mobile ads should prioritize concise messaging and quick load times, while desktop ads can support more detailed content and interactive elements.
  3. **Enhance Bot Detection and Filtering:**
     - Implement advanced analytics tools to distinguish between human and bot traffic. This will help ensure that advertising metrics accurately reflect real user engagement. Understanding the nature and behavior of bot traffic can also aid in refining ad strategies and improving the relevance of content presented to human users.
  4. **Leverage Language-Specific Analytics:**
     - Use analytics to track the performance of content and ads in different languages. This data can provide insights into the preferences and behaviors of different linguistic groups, allowing for more targeted content creation and ad placement. Businesses can adjust their strategies based on which languages show higher engagement or conversion rates.
  5. **Localized Content Strategy:**
     - Beyond advertising, consider developing localized content strategies for the non-English languages. This could include blog posts, social media content, and customer support resources. A comprehensive approach to localization can enhance user experience and build stronger connections with diverse user groups.
  6. **Regularly Update and Test Ad Content:**
     - Given the changing dynamics of user behavior and preferences, regularly update and test ad content across different platforms and languages. A/B testing can help determine which ad versions perform best in terms of

engagement and conversions. Continuously iterating on ad designs and messaging can lead to more effective campaigns.

7. **Focus on High-Engagement Pages:**
   - While English pages may have the highest number, it is crucial to identify which specific pages or types of content within each language group have the highest engagement. Prioritize ad placements on these high-engagement pages to maximize visibility and impact. Use heatmaps and user flow analysis to understand where users spend the most time and place ads strategically.
8. **Explore Cross-Platform Advertising:**
   - Consider cross-platform advertising strategies that leverage both mobile and desktop channels. For example, an ad campaign could start with a teaser on mobile and drive users to a more detailed landing page on desktop. This approach can take advantage of the strengths of each platform and create a cohesive user journey.

By implementing these additional insights and recommendations, businesses can optimize their advertising strategies, improve user engagement, and effectively reach a diverse audience. This comprehensive approach will not only enhance the effectiveness of ad campaigns but also contribute to overall business growth and customer satisfaction.

Chapter 2 : Business Case Study 2 — **Intra-City Logistics Company**

Problem Description

The organization is India's Largest Marketplace for Intra-City Logistics. Leader in the country's $40 billion intra-city logistics market, it strives to improve the lives of 1,50,000+ driver-partners by providing them with consistent earning & independence. Currently, the company has serviced 5+ million customers. They work with a wide range of restaurants for delivering their items directly to the people. It has a number of delivery partners available for delivering the food, from various restaurants and wants to get an estimated delivery time that it can provide the customers on the basis of what they are ordering, from where and also the delivery partners. This dataset has the required data to train a regression model that will do the delivery time estimation, based on all those features.

Business Questions to be answered from Analysis

There can be broadly 10 categories of Business Questions that can be answered from this analysis.

Delivery Time Estimation:

- What is the expected delivery time for various orders based on the type of food, restaurant location, and delivery partner availability?
- How can we improve the accuracy of our estimated delivery times provided to customers?

Factors Influencing Delivery Time:

- What are the key factors that most significantly impact delivery time, such as distance, traffic conditions, time of day, or order size?
- How do different types of food or restaurants affect the delivery time estimates?

Optimizing Delivery Partner Allocation:

- How can we optimize the allocation of delivery partners to minimize delivery times and improve efficiency?
- What is the optimal number of delivery partners needed at different times of day to meet delivery time expectations?

Customer Satisfaction:

- How do accurate delivery time estimates impact customer satisfaction and repeat business?
- What is the acceptable range of deviation from estimated delivery times that customers are willing to tolerate?

Regional and Temporal Analysis:

- How do delivery times vary across different regions and times of day?
- What are the busiest times and regions, and how can we adjust our logistics to handle peak demand efficiently?

Operational Efficiency:

- What strategies can be implemented to reduce delivery times without compromising the quality of service?
- How can we use data on past deliveries to improve route planning and reduce delays?

Impact on Driver-Partner Experience:

- How does the delivery time estimation impact the earnings and job satisfaction of driver-partners?
- What measures can be taken to ensure fair distribution of delivery tasks among driver-partners?

Predictive Analytics:

- Can we predict potential delays and communicate these to customers proactively?
- How can machine learning models be leveraged to continuously improve delivery time predictions based on real-time data?

Integration with Restaurant Partners:

- How can collaboration with restaurant partners be optimized to reduce preparation time and improve overall delivery efficiency?
- What information from restaurants (e.g., order preparation time) can be integrated into the delivery time estimation model?

Cost-Benefit Analysis:

- What is the cost-benefit ratio of different strategies aimed at reducing delivery times?
- How can we balance the cost of faster deliveries with the potential increase in customer satisfaction and retention?

Analysis

- Basic Exploratory Data Analysis

1. **Boxplots for Categorical Variables**

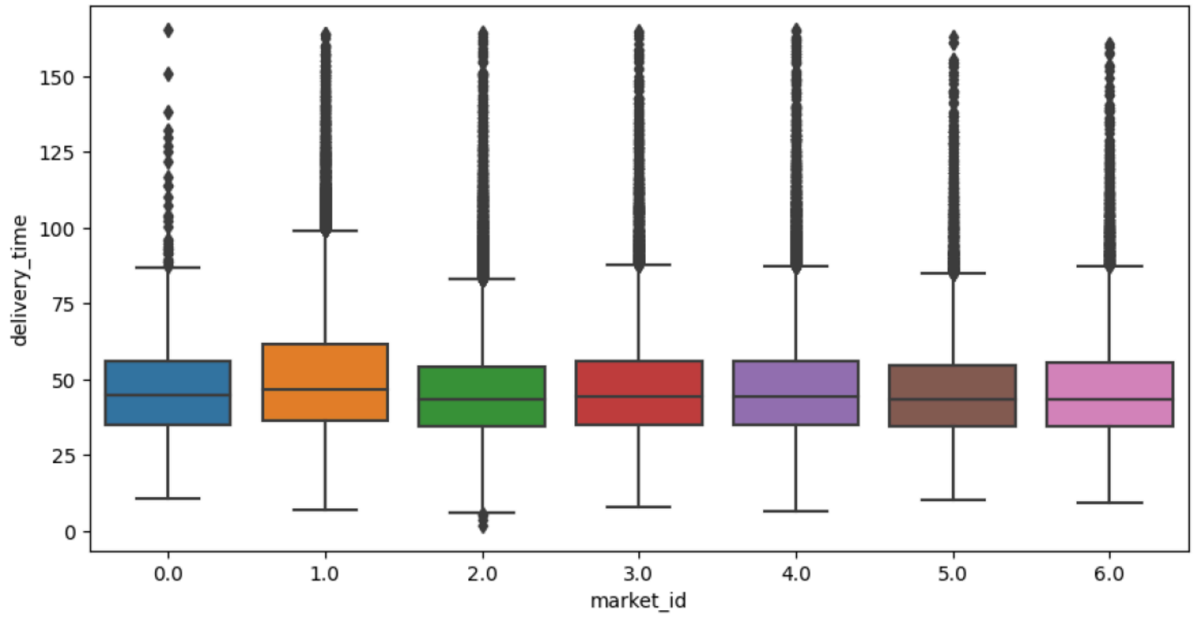


Figure 2.1: Boxplot for Market ID with respect to Delivery Time

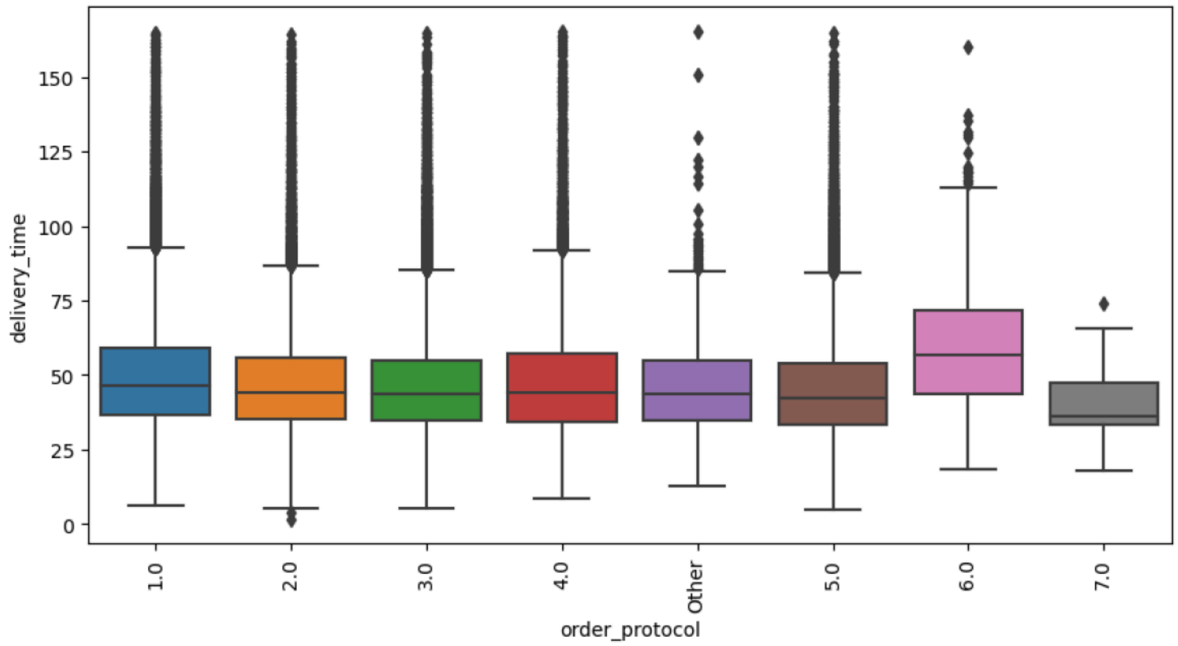


Figure 2.2: Boxplot for Order Protocol with respect to Delivery Time

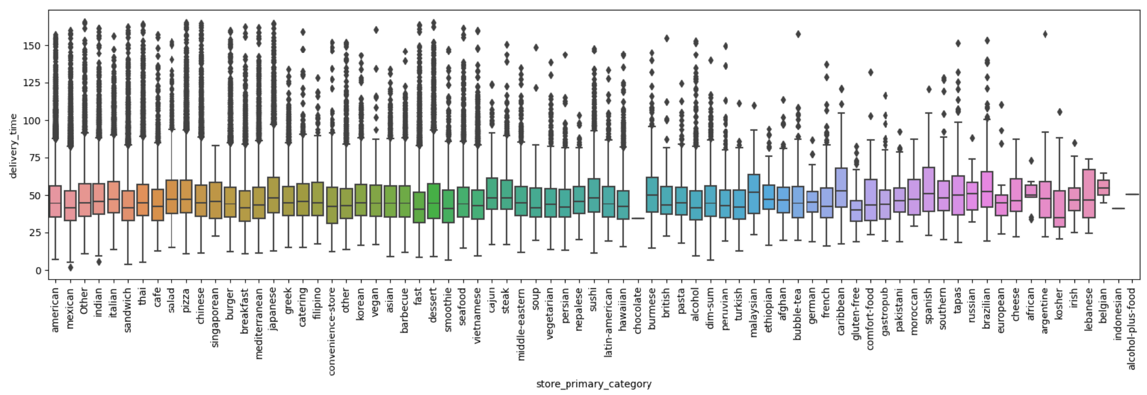


Figure 2.3: Boxplot for Store Primary Category with respect to Delivery Time

2. Time Series Plots

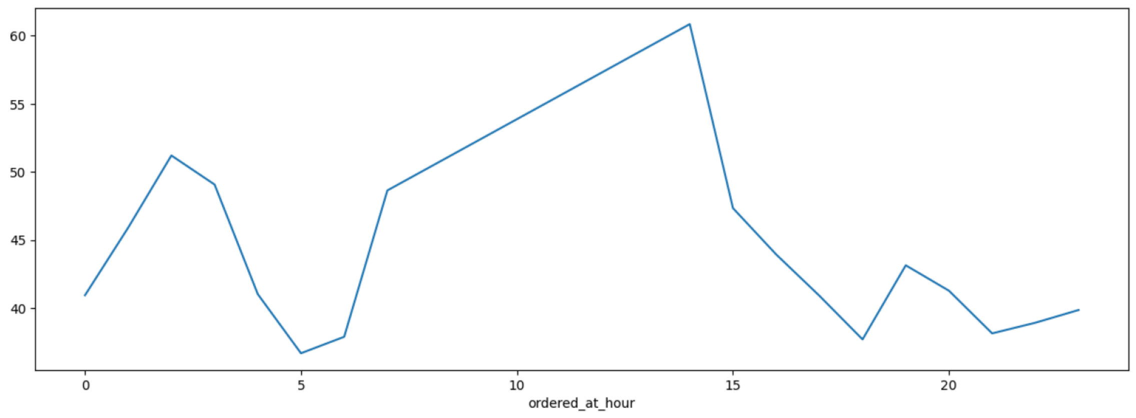


Figure 2.4: Time Series Plot for Median Delivery Time with respect to Hour of the Day

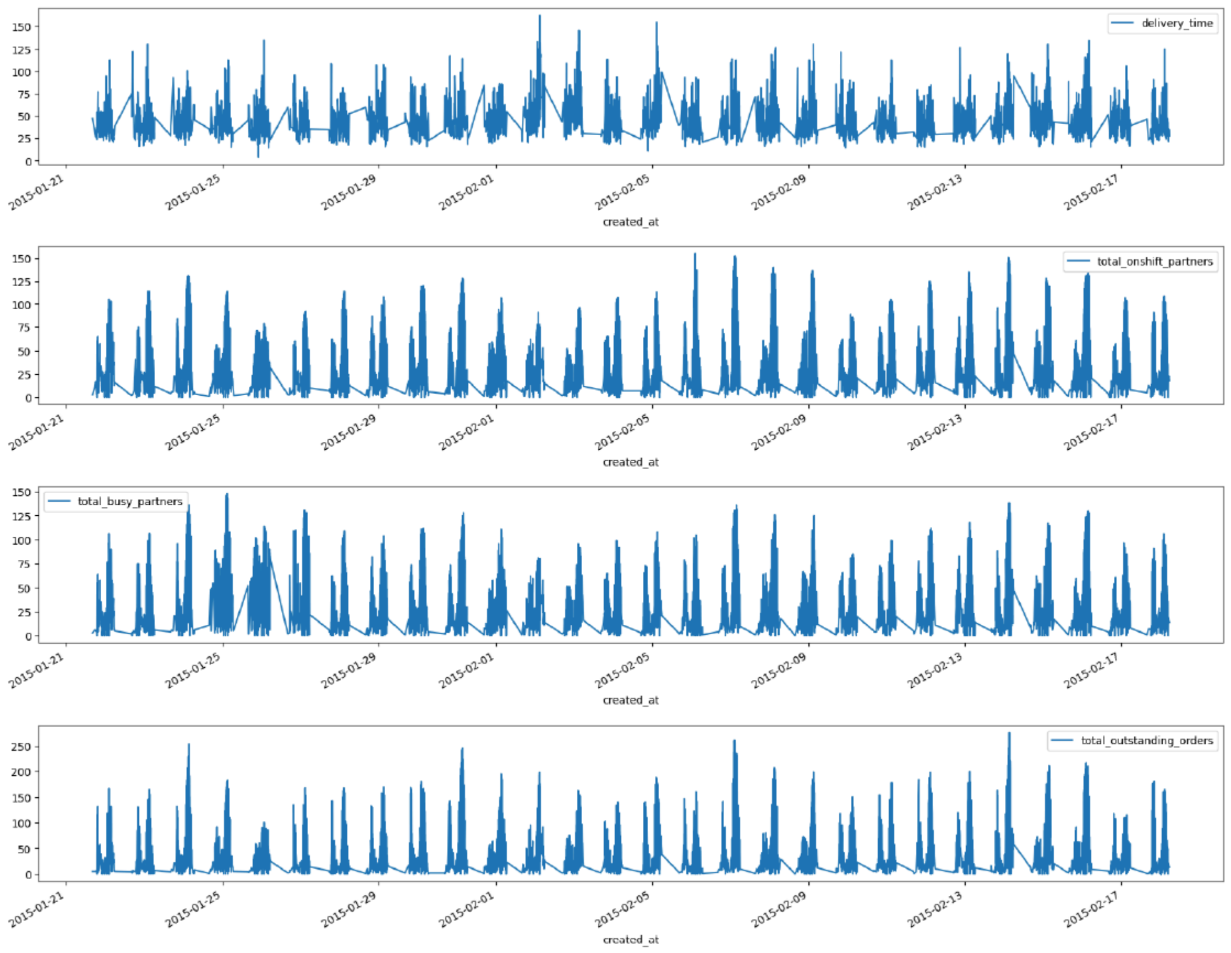


Figure 2.5: Time Series Plot to Check Trend and Seasonality for Delivery Time, Total Onshift Partners, Total Busy Partners and Total Outstanding Partners

- **Correlation Plot among Features**

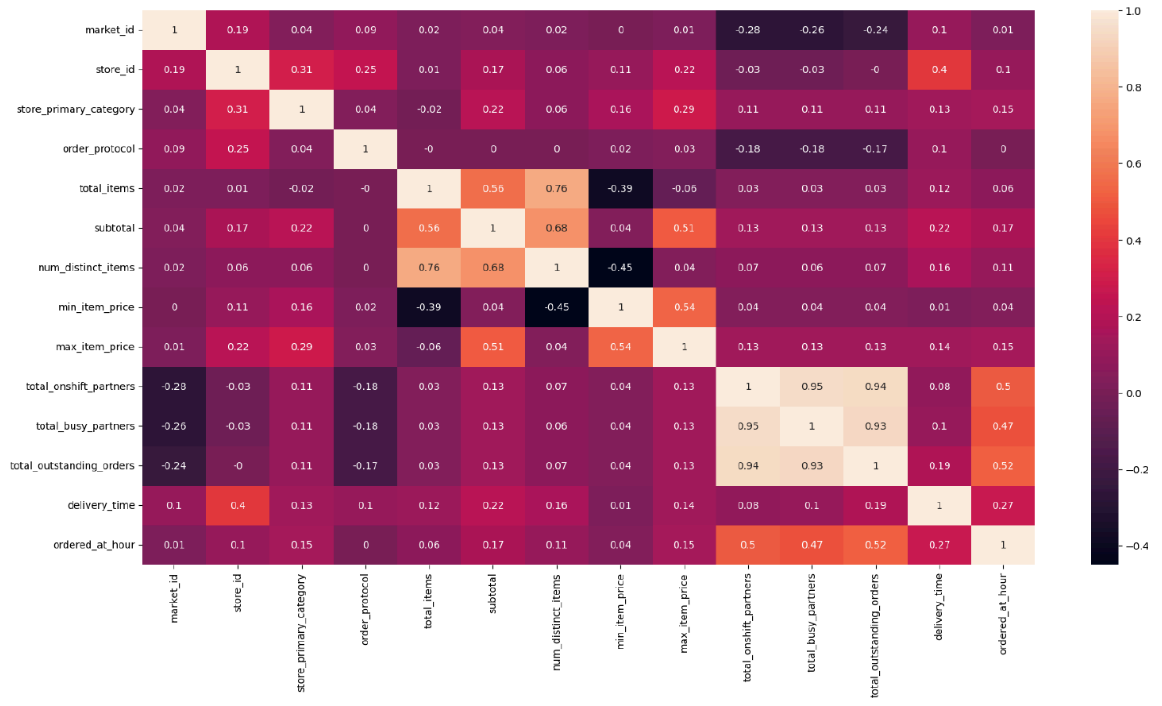
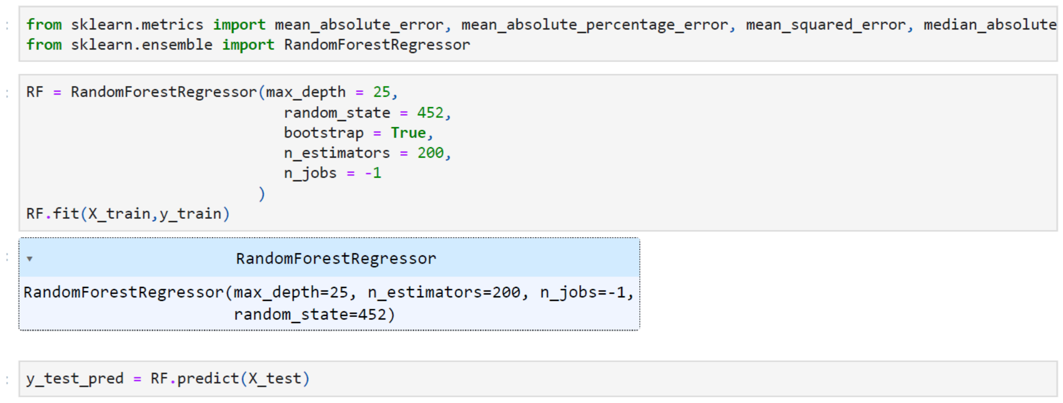


Figure 2.6: Correlation Plot amongst Features

- **Implementation of Classical Machine Learning Model:**

a. Random Forest Regressor



```
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error, median_absolute
from sklearn.ensemble import RandomForestRegressor
```

```
RF = RandomForestRegressor(max_depth = 25,
                           random_state = 452,
                           bootstrap = True,
                           n_estimators = 200,
                           n_jobs = -1
                         )
RF.fit(X_train,y_train)
```

```
                    RandomForestRegressor
RandomForestRegressor(max_depth=25, n_estimators=200, n_jobs=-1,
                      random_state=452)
```

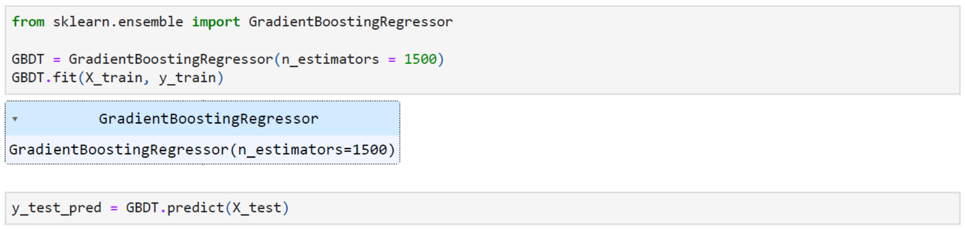
```
y_test_pred = RF.predict(X_test)
```

Figure 2.7: Random Forest Regressor Model Implementation

Observations for **Random Forest Regressor Model**:

i. Root Mean Squared Error (RMSE): 12.19
ii. Mean Absolute Error (MAE): 9.40
iii. R^2-Score for Test Data: 0.33

b. **Gradient Boosting Regressor**



```
from sklearn.ensemble import GradientBoostingRegressor

GBDT = GradientBoostingRegressor(n_estimators = 1500)
GBDT.fit(X_train, y_train)
```

```
▾          GradientBoostingRegressor
GradientBoostingRegressor(n_estimators=1500)
```

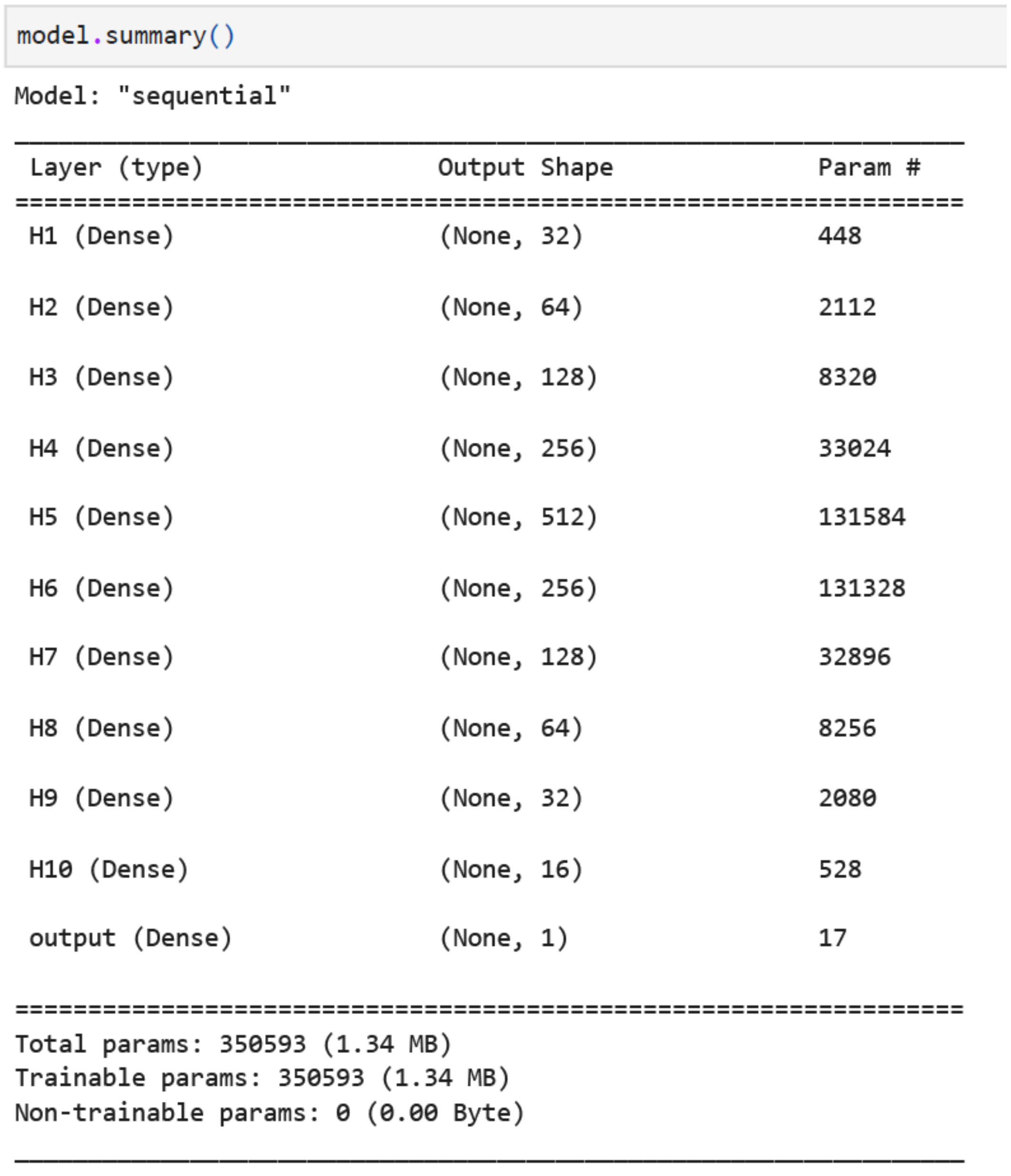
```
y_test_pred = GBDT.predict(X_test)
```

Figure 2.8: Gradient Boosting Regressor Model Implementation

Observations for **Random Forest Regressor Model**:

i. Root Mean Squared Error (RMSE): 11.85
ii. Mean Absolute Error (MAE): 9.1
iii. R^2-Score for Test Data: 0.37

- Implementation of **Neural Network Model** using **TensorFlow:**



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 H1 (Dense)                  (None, 32)                448

 H2 (Dense)                  (None, 64)                2112

 H3 (Dense)                  (None, 128)               8320

 H4 (Dense)                  (None, 256)               33024

 H5 (Dense)                  (None, 512)               131584

 H6 (Dense)                  (None, 256)               131328

 H7 (Dense)                  (None, 128)               32896

 H8 (Dense)                  (None, 64)                8256

 H9 (Dense)                  (None, 32)                2080

 H10 (Dense)                 (None, 16)                528

 output (Dense)              (None, 1)                 17

=================================================================
Total params: 350593 (1.34 MB)
Trainable params: 350593 (1.34 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2.9: TensorFlow Based Neural Network Model Summary

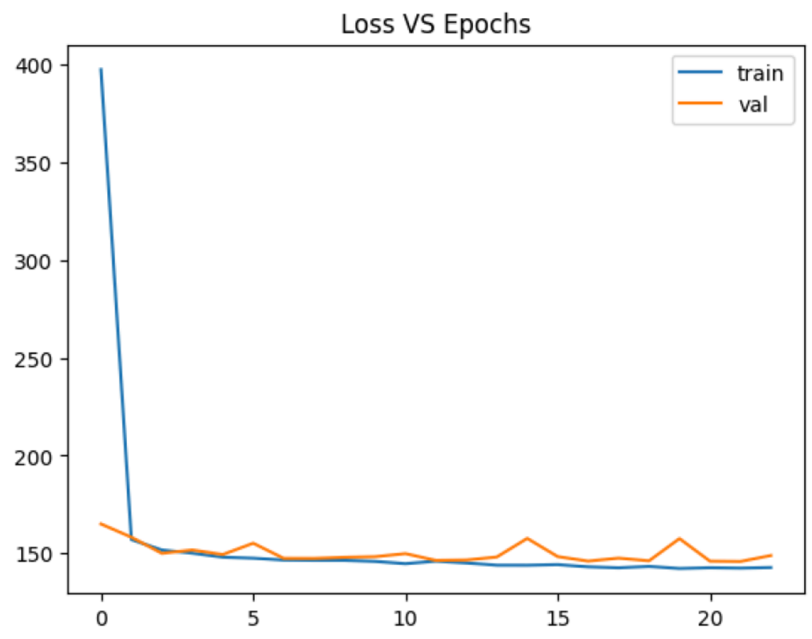


Figure 2.10: Loss Vs. Epoch Plot

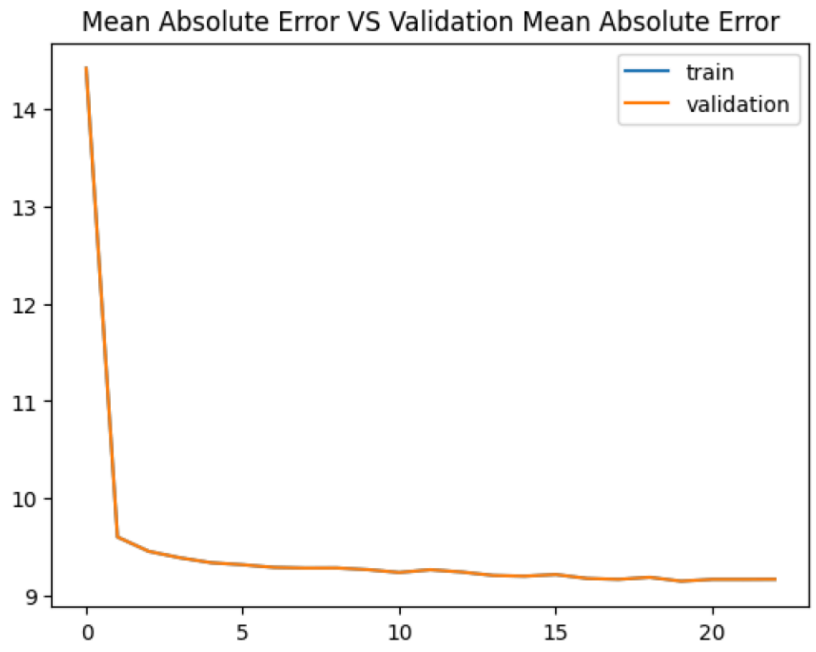


Figure 2.11: Plot for Mean Absolute Error vs. Validation Mean Absolute Error

Observations for **Random Forest Regressor Model**:

1. Mean Square Error (MSE): 147.69
2. Root Mean Squared Error (RMSE): 12.15
3. Mean Absolute Error (MAE): 9.49

Insights and Recommendations

- **Business Insights**

1. **Customer Expectation Management:** Accurate delivery time estimates are crucial for managing customer expectations. Customers value transparency and reliability, and providing precise delivery times can enhance trust and satisfaction, leading to repeat business and positive word-of-mouth.

2. **Impact of Multiple Factors:** Delivery time is influenced by various factors, including the type of order, restaurant location, traffic conditions, distance, and delivery partner availability. Understanding and analyzing these factors can help in developing a more robust delivery time estimation model.

3. **Operational Efficiency:** Efficient allocation of delivery partners and optimized route planning can significantly reduce delivery times. This efficiency can lead to cost savings, improved customer satisfaction, and better utilization of resources.

4. **Regional and Temporal Variability:** Delivery times can vary widely depending on the region and time of day. Peak hours, high-traffic areas, and specific regions may require additional considerations to ensure timely deliveries.

5. **Driver-Partner Experience:** The accuracy of delivery time estimates also affects driver-partners. Fair and efficient allocation of tasks can enhance driver-partner satisfaction and retention, which is critical for maintaining a reliable delivery network.

6. **Data-Driven Decision Making:** Leveraging historical data and real-time analytics can enhance the accuracy of delivery time predictions. Businesses can use data to identify patterns, forecast demand, and adjust operations dynamically.

7. **Cross-Industry Applicability:** The approach to estimating delivery times can be applied across various industries, including food delivery, e-commerce, logistics, courier services, and ride-sharing. This cross-industry relevance highlights the value of developing a versatile and scalable delivery time estimation model.

- **Business Recommendations**

1. **Develop a Comprehensive Predictive Model:** Invest in building a machine learning model that considers all relevant factors, such as order details, pickup location, traffic data, distance, and delivery partner availability. This model should be continuously updated with new data to improve accuracy.

2. **Implement Real-Time Traffic and Weather Data:** Incorporate real-time traffic and weather data into the predictive model. This can help adjust delivery time estimates dynamically and provide more accurate predictions during adverse conditions.

3. **Optimize Delivery Partner Allocation:** Use data-driven insights to optimize the allocation of delivery partners. Consider implementing dynamic dispatch systems that assign partners based on real-time availability and proximity to the pickup location.

4. **Enhance Communication with Customers:** Provide customers with real-time updates on their delivery status, including estimated time of arrival (ETA) and any delays. Transparent communication can improve customer satisfaction and reduce anxiety about waiting times.

5. **Analyze Peak Times and High-Traffic Areas:** Identify peak times and regions with frequent traffic congestion. Develop strategies to manage these challenges, such as adjusting delivery windows, increasing the number of available delivery partners during peak hours, or offering incentives for off-peak deliveries.

6. **Focus on Driver-Partner Well-being:** Ensure fair distribution of delivery tasks among driver-partners and consider factors like traffic and distance in their assignments. Offer incentives for consistent performance and provide support to improve their working conditions.

7. **Expand Across Industries:** Explore opportunities to apply the delivery time estimation model to other industries, such as e-commerce and courier services. Customizing the model for different sectors can create new revenue streams and strengthen market presence.

8. **Continuous Monitoring and Improvement:** Regularly monitor the performance of the delivery time estimation model and gather customer feedback. Use this feedback to make necessary adjustments and improvements, ensuring the model remains accurate and reliable.

9. **Leverage Customer Data for Personalization:** Use customer data to personalize delivery experiences. For example, offer loyalty programs or special offers for frequent customers who experience consistent and timely deliveries.

10. **Explore Partnerships:** Consider partnerships with traffic data providers, weather forecasting services, and other relevant stakeholders to enhance the accuracy and reliability of the predictive model.

By implementing these recommendations, the company can not only provide accurate delivery time estimates but also enhance customer satisfaction, improve operational efficiency, and expand its market reach.

Chapter 3 : Business Case Study 3 — **Fresh Produce Supply-Chain Company**

Problem Description

This company is India's largest fresh produce supply chain company. They are pioneers in solving one of the toughest supply chain problems of the world by leveraging innovative technology. They source fresh produce from farmers and deliver them to businesses within 12 hours. An integral component of their automation process is the development of robust classifiers which can distinguish between images of different types of vegetables, while also correctly labeling images that do not contain any one type of vegetable as noise. As a starting point, the company has provided us with a dataset scraped from the web which contains train and test folders, each having 4 subfolders with images of onions, potatoes, tomatoes and some market scenes. We have been tasked with preparing a multiclass classifier for identifying these vegetables. The dataset provided has all the required images to achieve the task. This dataset contains images of the following food items: Noise-Indian market and images of vegetables — Onion, Potato and Tomato. The images in this dataset were scraped from Google.

Business Questions to be answered from Analysis

- **Classification Accuracy and Reliability:**
  - How accurately can the developed multiclass classifier distinguish between onions, potatoes, tomatoes, and noise (non-vegetable images)?
  - What is the expected accuracy, precision, recall, and F1-score for each class (onion, potato, tomato, and noise) using the current dataset?
- **Operational Efficiency:**
  - How can the classification model improve the efficiency of sorting and handling fresh produce in the supply chain?
  - What are the potential time and cost savings from automating the vegetable classification process compared to manual sorting?
- **Quality Control and Assurance:**
  - How can the classifier help in maintaining the quality of fresh produce by accurately identifying and excluding non-vegetable (noise) images?
  - What impact does accurate classification have on reducing the rejection rate of fresh produce deliveries to businesses?

- **Scalability and Adaptability:**
    - Can the classifier be scaled to identify additional types of vegetables or other produce in the future? What are the steps and challenges involved in expanding the classifier's capabilities?
    - How well does the classifier handle variations in image quality, lighting, and background noise in the provided dataset?

- **Data Management and Expansion:**
    - What are the data augmentation and preprocessing strategies that can enhance the performance of the classifier, given the diverse nature of images scraped from the web?
    - How can additional data sources or real-time image data from supply chain operations be integrated to improve the classifier's accuracy and robustness?

- **Impact on Supply Chain Logistics:**
    - How does the implementation of the classifier affect the logistics of sourcing and delivering fresh produce within the 12-hour window?
    - What are the implications of using this technology on inventory management, especially concerning the sorting and categorization of different vegetables?

- **Customer and Business Impact:**
    - How can the accurate classification of vegetables contribute to better service quality for business clients (such as retailers and restaurants)?
    - What is the potential impact on customer satisfaction and business relationships when errors in vegetable classification are minimized?

- **Technology Integration and Deployment:**
    - What are the requirements and challenges for integrating the classifier into the existing automation and technology infrastructure of the company?
    - How can the classifier be deployed in real-time scenarios, such as at sorting centers or during transport, to enhance operational workflows?

- **Future Development and Research:**
    - What are the opportunities for further research and development in the area of image-based classification for the fresh produce industry?
    - How can the company leverage machine learning and computer vision advancements to enhance other aspects of the supply chain, such as predicting produce spoilage or optimizing storage conditions?

These questions aim to explore the practical applications, benefits, and challenges of implementing an image-based vegetable classification system in the fresh produce supply chain, as well as its broader implications for the company's operations and customer relations.

Analysis

These are the sample images (with labels) provided in the Train Dataset.

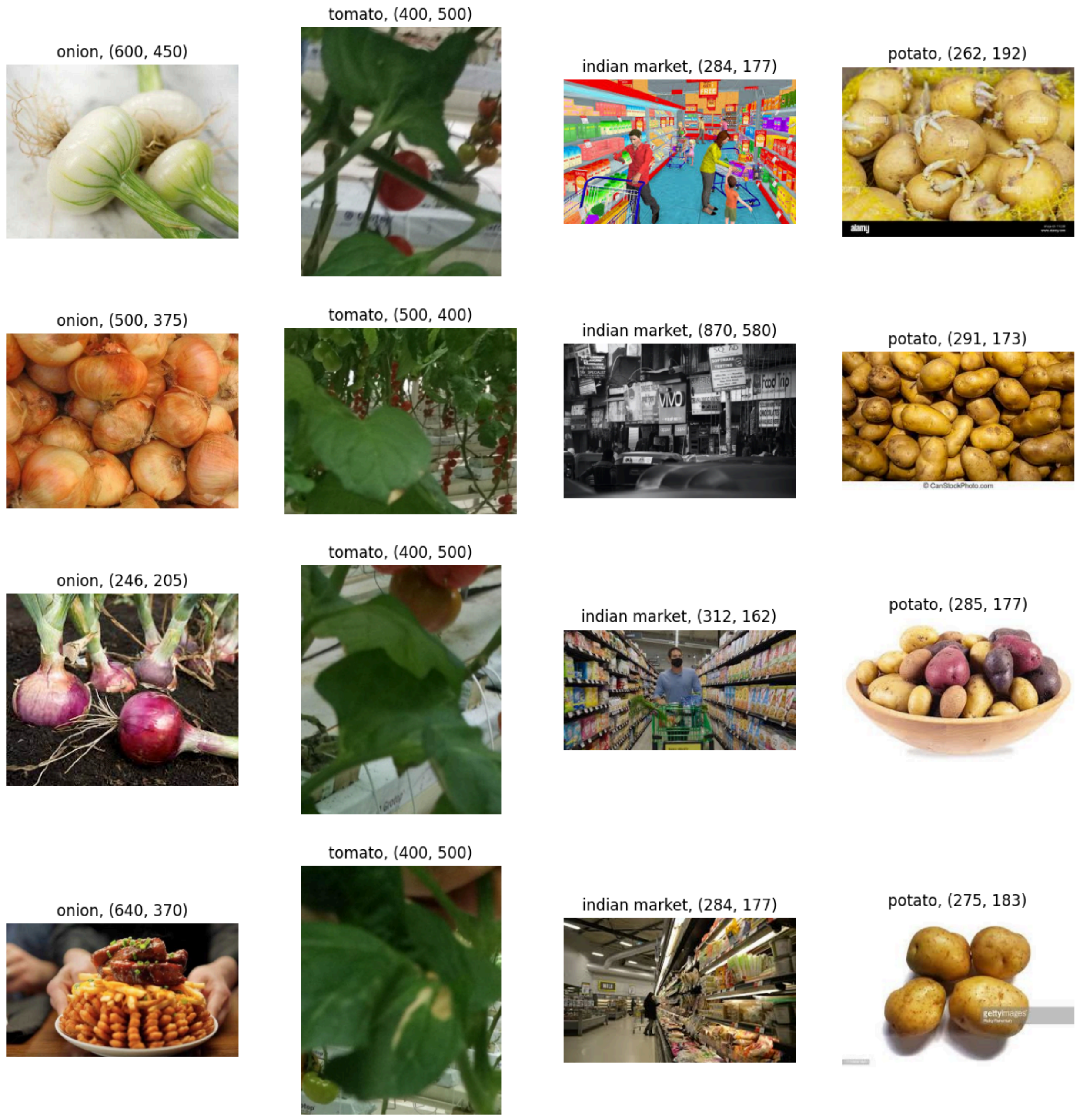


Figure 3.1: Different Sample Images Present in the Training Dataset

- Basic Exploratory Data Analysis

1. **Training Data Count per Class:**

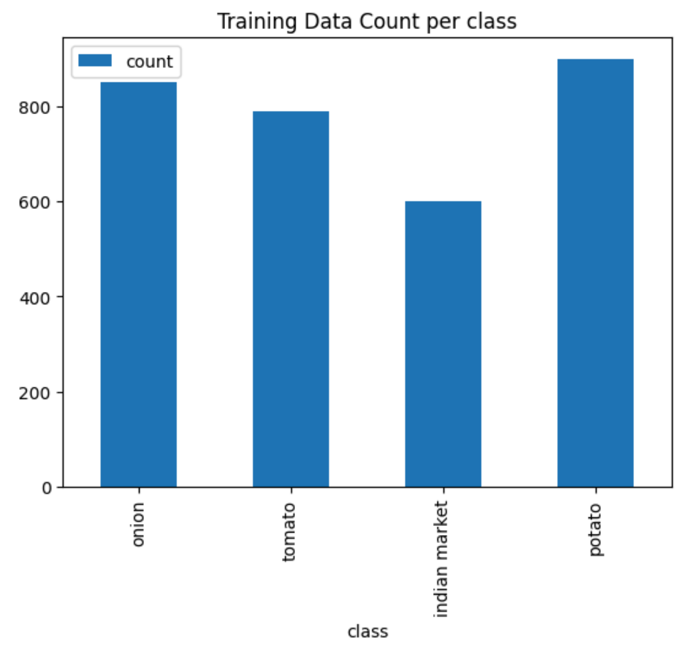


Figure 3.2:Countplot Showing Training Data Count per Class

2. **Test Data Count per Class:**

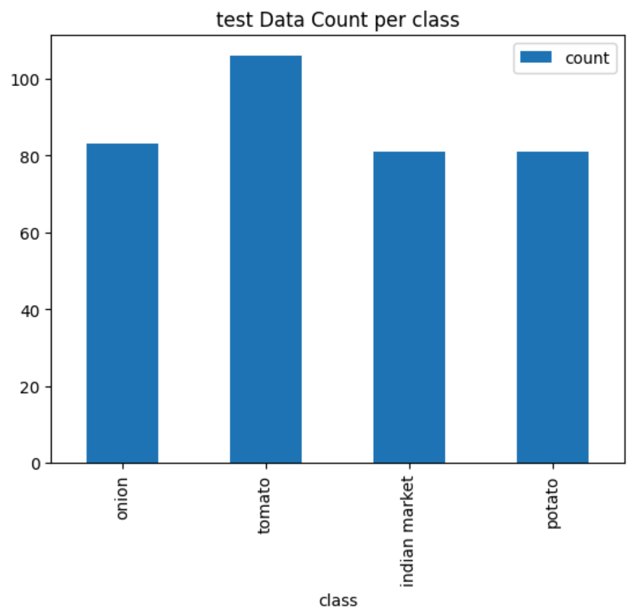


Figure 3.3: Countplot Showing Test Data Count per Class

- **Artificial Neural Network (ANN)** Model

1. ANN Model Summary

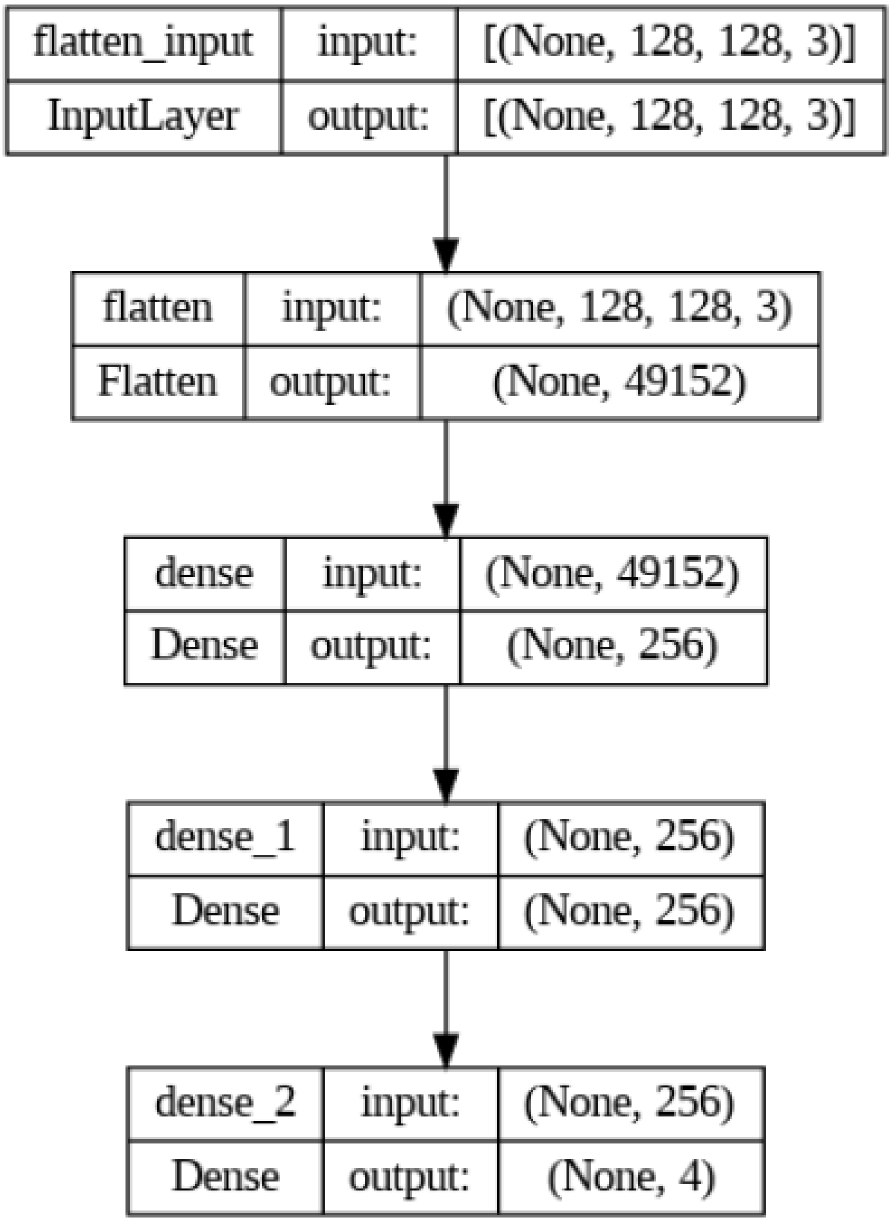
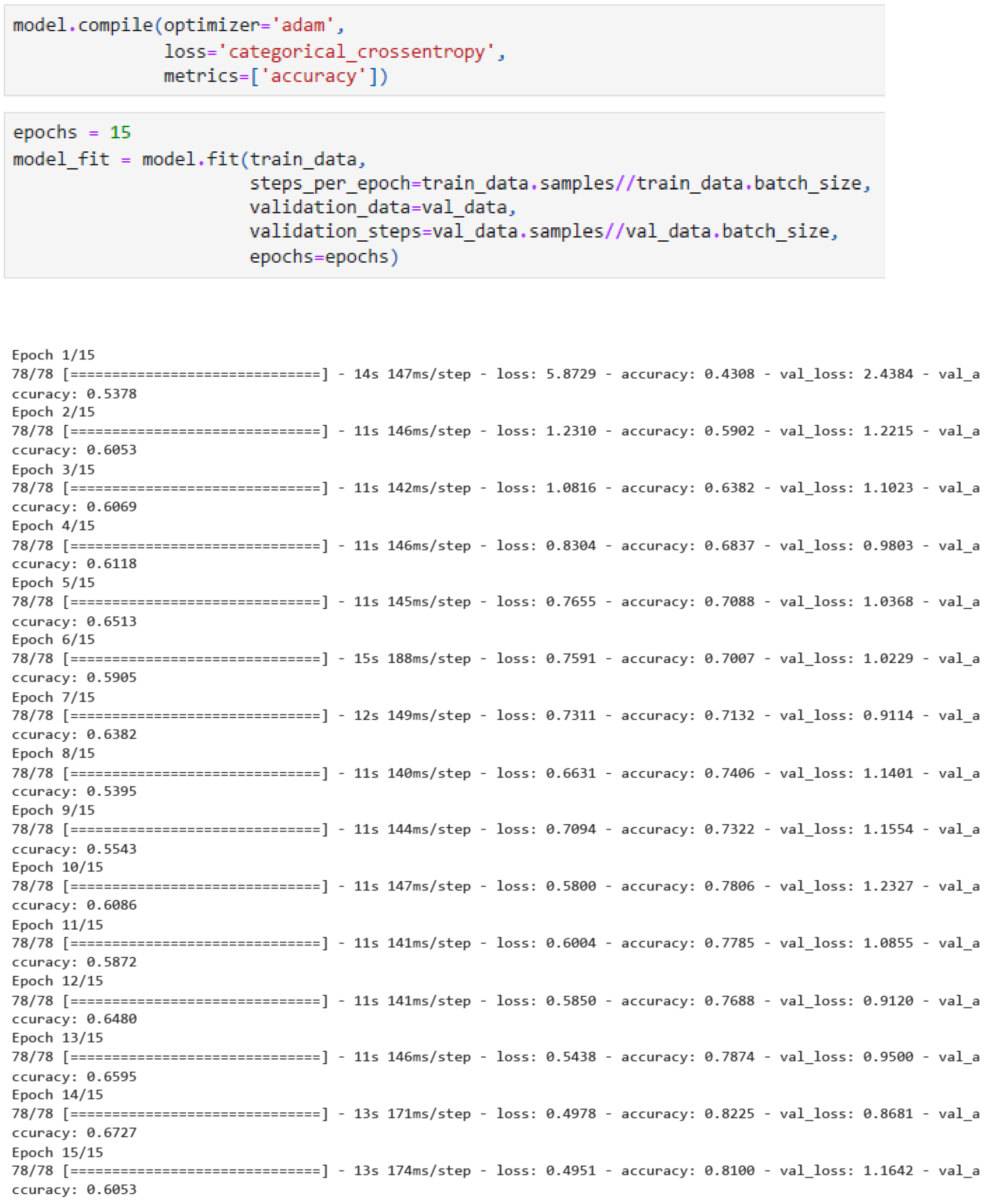


Figure 3.4: ANN Model Summary

2. ANN Model Training



```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
epochs = 15
model_fit = model.fit(train_data,
                      steps_per_epoch=train_data.samples//train_data.batch_size,
                      validation_data=val_data,
                      validation_steps=val_data.samples//val_data.batch_size,
                      epochs=epochs)
```

```
Epoch 1/15
78/78 [==============================] - 14s 147ms/step - loss: 5.8729 - accuracy: 0.4308 - val_loss: 2.4384 - val_a
ccuracy: 0.5378
Epoch 2/15
78/78 [==============================] - 11s 146ms/step - loss: 1.2310 - accuracy: 0.5902 - val_loss: 1.2215 - val_a
ccuracy: 0.6053
Epoch 3/15
78/78 [==============================] - 11s 142ms/step - loss: 1.0816 - accuracy: 0.6382 - val_loss: 1.1023 - val_a
ccuracy: 0.6069
Epoch 4/15
78/78 [==============================] - 11s 146ms/step - loss: 0.8304 - accuracy: 0.6837 - val_loss: 0.9803 - val_a
ccuracy: 0.6118
Epoch 5/15
78/78 [==============================] - 11s 145ms/step - loss: 0.7655 - accuracy: 0.7088 - val_loss: 1.0368 - val_a
ccuracy: 0.6513
Epoch 6/15
78/78 [==============================] - 15s 188ms/step - loss: 0.7591 - accuracy: 0.7007 - val_loss: 1.0229 - val_a
ccuracy: 0.5905
Epoch 7/15
78/78 [==============================] - 12s 149ms/step - loss: 0.7311 - accuracy: 0.7132 - val_loss: 0.9114 - val_a
ccuracy: 0.6382
Epoch 8/15
78/78 [==============================] - 11s 140ms/step - loss: 0.6631 - accuracy: 0.7406 - val_loss: 1.1401 - val_a
ccuracy: 0.5395
Epoch 9/15
78/78 [==============================] - 11s 144ms/step - loss: 0.7094 - accuracy: 0.7322 - val_loss: 1.1554 - val_a
ccuracy: 0.5543
Epoch 10/15
78/78 [==============================] - 11s 147ms/step - loss: 0.5800 - accuracy: 0.7806 - val_loss: 1.2327 - val_a
ccuracy: 0.6086
Epoch 11/15
78/78 [==============================] - 11s 141ms/step - loss: 0.6004 - accuracy: 0.7785 - val_loss: 1.0855 - val_a
ccuracy: 0.5872
Epoch 12/15
78/78 [==============================] - 11s 141ms/step - loss: 0.5850 - accuracy: 0.7688 - val_loss: 0.9120 - val_a
ccuracy: 0.6480
Epoch 13/15
78/78 [==============================] - 11s 146ms/step - loss: 0.5438 - accuracy: 0.7874 - val_loss: 0.9500 - val_a
ccuracy: 0.6595
Epoch 14/15
78/78 [==============================] - 13s 171ms/step - loss: 0.4978 - accuracy: 0.8225 - val_loss: 0.8681 - val_a
ccuracy: 0.6727
Epoch 15/15
78/78 [==============================] - 13s 174ms/step - loss: 0.4951 - accuracy: 0.8100 - val_loss: 1.1642 - val_a
ccuracy: 0.6053
```

Figure 3.5: ANN Model Training

3. Accuracy and Loss vs. Epoch

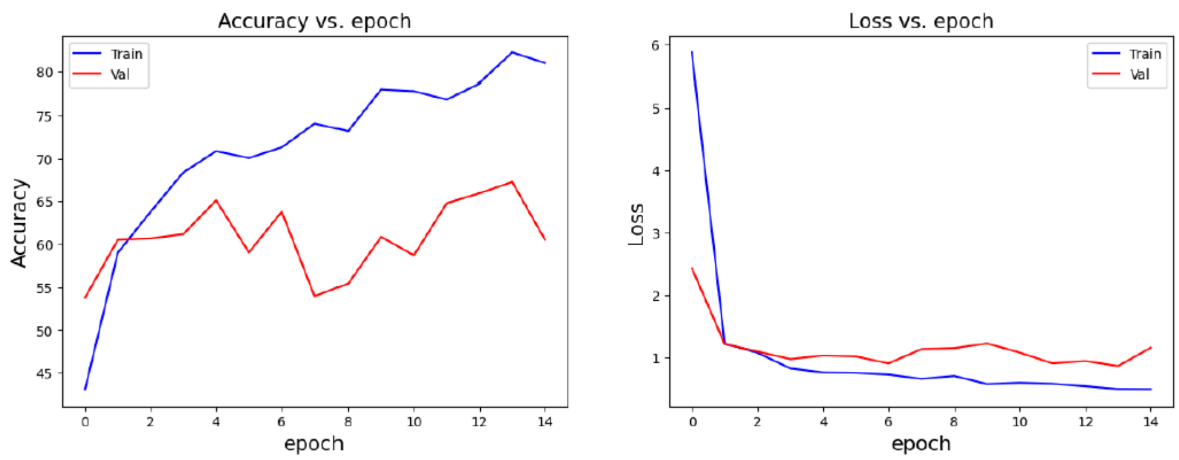


Figure 3.6: Plot for Accuracy and Loss vs. Epoch

4. Confusion Matrix

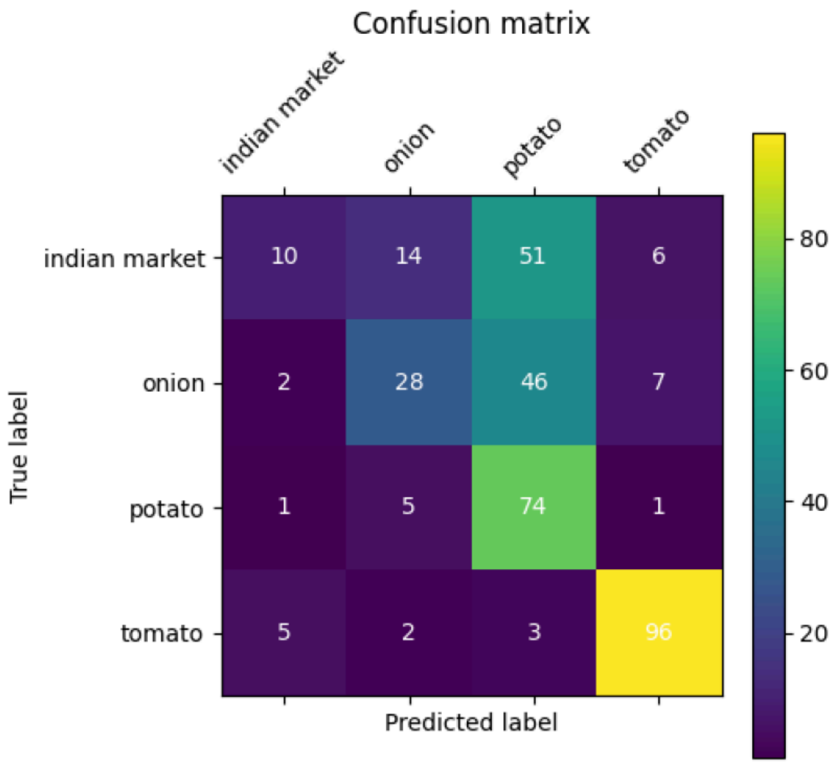
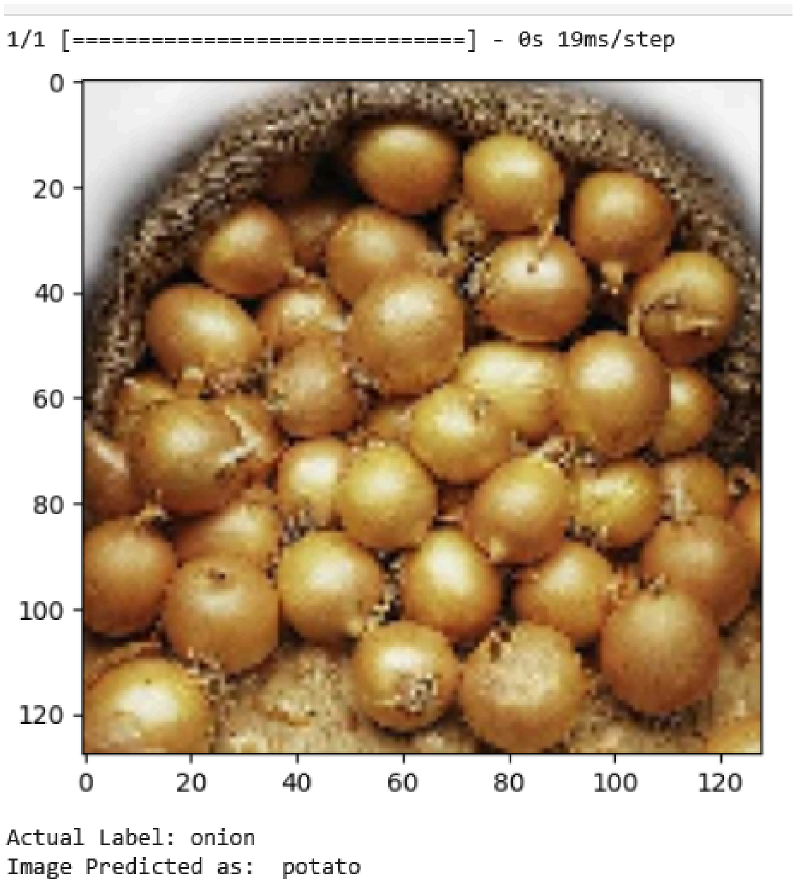


Figure 3.7: Confusion Matrix after ANN Classification

5. Model Parameters and Performance Metrics

```
Total params: 12,649,988                Test accuracy: 0.62
Trainable params: 12,649,988            Precision: 0.62
Non-trainable params: 0                 Recall: 0.62
Loss: 1.0572
```

6. Result on Random Datapoint



```
1/1 [==============================] - 0s 19ms/step
```

```
Actual Label: onion
Image Predicted as:  potato
```

Figure 3.8: Result from ANN Model on a Random Datapoint

- **Convolutional Neural Network (CNN)** Model

    1. CNN Model Summary

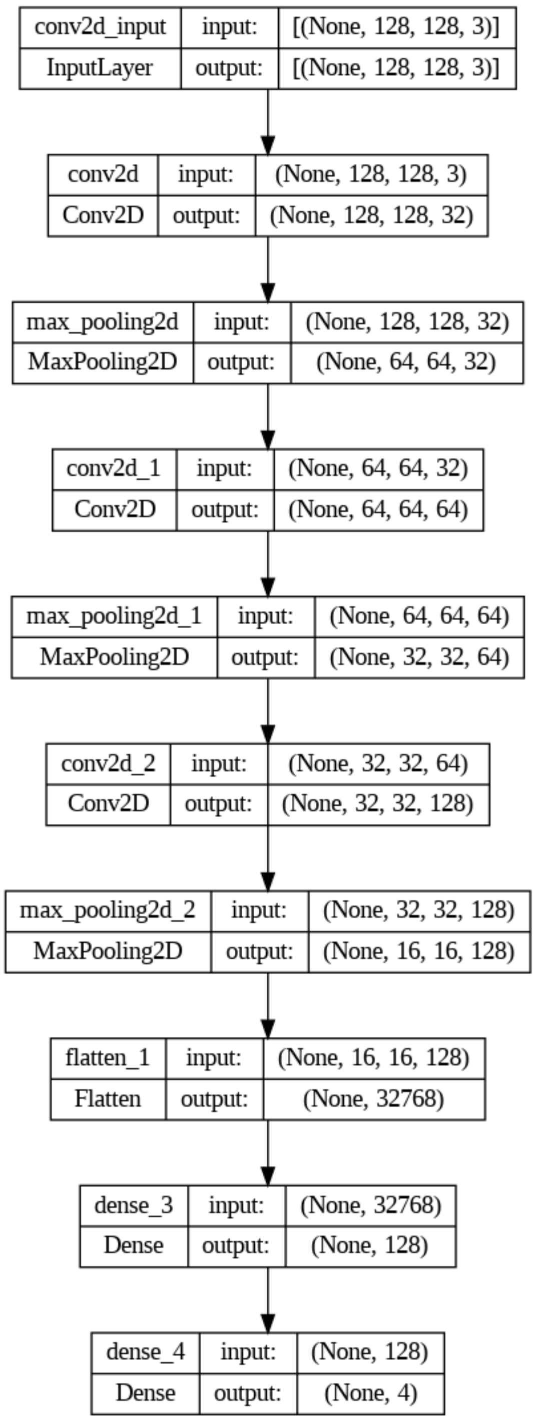
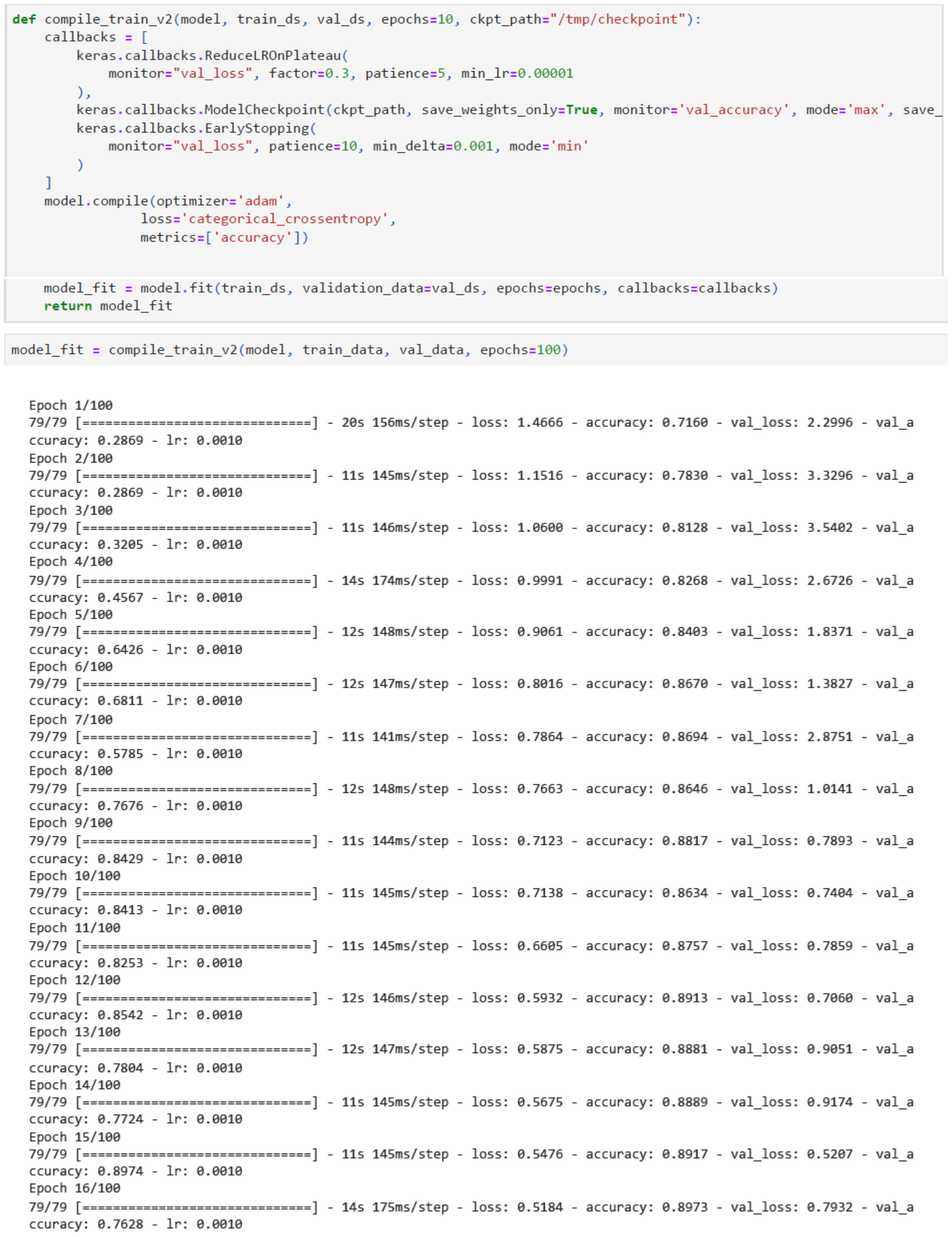

conv2d_input | input: | [(None, 128, 128, 3)]
InputLayer | output: | [(None, 128, 128, 3)]
conv2d | input: | (None, 128, 128, 3)
Conv2D | output: | (None, 128, 128, 32)
max_pooling2d | input: | (None, 128, 128, 32)
MaxPooling2D | output: | (None, 64, 64, 32)
conv2d_1 | input: | (None, 64, 64, 32)
Conv2D | output: | (None, 64, 64, 64)
max_pooling2d_1 | input: | (None, 64, 64, 64)
MaxPooling2D | output: | (None, 32, 32, 64)
conv2d_2 | input: | (None, 32, 32, 64)
Conv2D | output: | (None, 32, 32, 128)
max_pooling2d_2 | input: | (None, 32, 32, 128)
MaxPooling2D | output: | (None, 16, 16, 128)
flatten_1 | input: | (None, 16, 16, 128)
Flatten | output: | (None, 32768)
dense_3 | input: | (None, 32768)
Dense | output: | (None, 128)
dense_4 | input: | (None, 128)
Dense | output: | (None, 4)

Figure 3.9: CNN Model Summary

2. CNN Model Training



```python
def compile_train_v2(model, train_ds, val_ds, epochs=10, ckpt_path="/tmp/checkpoint"):
    callbacks = [
        keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monitor='val_accuracy', mode='max', save_
        keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )
    ]
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])


    model_fit = model.fit(train_ds, validation_data=val_ds, epochs=epochs, callbacks=callbacks)
    return model_fit
```

```python
model_fit = compile_train_v2(model, train_data, val_data, epochs=100)
```

```
Epoch 1/100
79/79 [==============================] - 20s 156ms/step - loss: 1.4666 - accuracy: 0.7160 - val_loss: 2.2996 - val_a
ccuracy: 0.2869 - lr: 0.0010
Epoch 2/100
79/79 [==============================] - 11s 145ms/step - loss: 1.1516 - accuracy: 0.7830 - val_loss: 3.3296 - val_a
ccuracy: 0.2869 - lr: 0.0010
Epoch 3/100
79/79 [==============================] - 11s 146ms/step - loss: 1.0600 - accuracy: 0.8128 - val_loss: 3.5402 - val_a
ccuracy: 0.3205 - lr: 0.0010
Epoch 4/100
79/79 [==============================] - 14s 174ms/step - loss: 0.9991 - accuracy: 0.8268 - val_loss: 2.6726 - val_a
ccuracy: 0.4567 - lr: 0.0010
Epoch 5/100
79/79 [==============================] - 12s 148ms/step - loss: 0.9061 - accuracy: 0.8403 - val_loss: 1.8371 - val_a
ccuracy: 0.6426 - lr: 0.0010
Epoch 6/100
79/79 [==============================] - 12s 147ms/step - loss: 0.8016 - accuracy: 0.8670 - val_loss: 1.3827 - val_a
ccuracy: 0.6811 - lr: 0.0010
Epoch 7/100
79/79 [==============================] - 11s 141ms/step - loss: 0.7864 - accuracy: 0.8694 - val_loss: 2.8751 - val_a
ccuracy: 0.5785 - lr: 0.0010
Epoch 8/100
79/79 [==============================] - 12s 148ms/step - loss: 0.7663 - accuracy: 0.8646 - val_loss: 1.0141 - val_a
ccuracy: 0.7676 - lr: 0.0010
Epoch 9/100
79/79 [==============================] - 11s 144ms/step - loss: 0.7123 - accuracy: 0.8817 - val_loss: 0.7893 - val_a
ccuracy: 0.8429 - lr: 0.0010
Epoch 10/100
79/79 [==============================] - 11s 145ms/step - loss: 0.7138 - accuracy: 0.8634 - val_loss: 0.7404 - val_a
ccuracy: 0.8413 - lr: 0.0010
Epoch 11/100
79/79 [==============================] - 11s 145ms/step - loss: 0.6605 - accuracy: 0.8757 - val_loss: 0.7859 - val_a
ccuracy: 0.8253 - lr: 0.0010
Epoch 12/100
79/79 [==============================] - 12s 146ms/step - loss: 0.5932 - accuracy: 0.8913 - val_loss: 0.7060 - val_a
ccuracy: 0.8542 - lr: 0.0010
Epoch 13/100
79/79 [==============================] - 12s 147ms/step - loss: 0.5875 - accuracy: 0.8881 - val_loss: 0.9051 - val_a
ccuracy: 0.7804 - lr: 0.0010
Epoch 14/100
79/79 [==============================] - 11s 145ms/step - loss: 0.5675 - accuracy: 0.8889 - val_loss: 0.9174 - val_a
ccuracy: 0.7724 - lr: 0.0010
Epoch 15/100
79/79 [==============================] - 11s 145ms/step - loss: 0.5476 - accuracy: 0.8917 - val_loss: 0.5207 - val_a
ccuracy: 0.8974 - lr: 0.0010
Epoch 16/100
79/79 [==============================] - 14s 175ms/step - loss: 0.5184 - accuracy: 0.8973 - val_loss: 0.7932 - val_a
ccuracy: 0.7628 - lr: 0.0010
```

Figure 3.10: CNN Model Training

3. Accuracy vs. Epoch

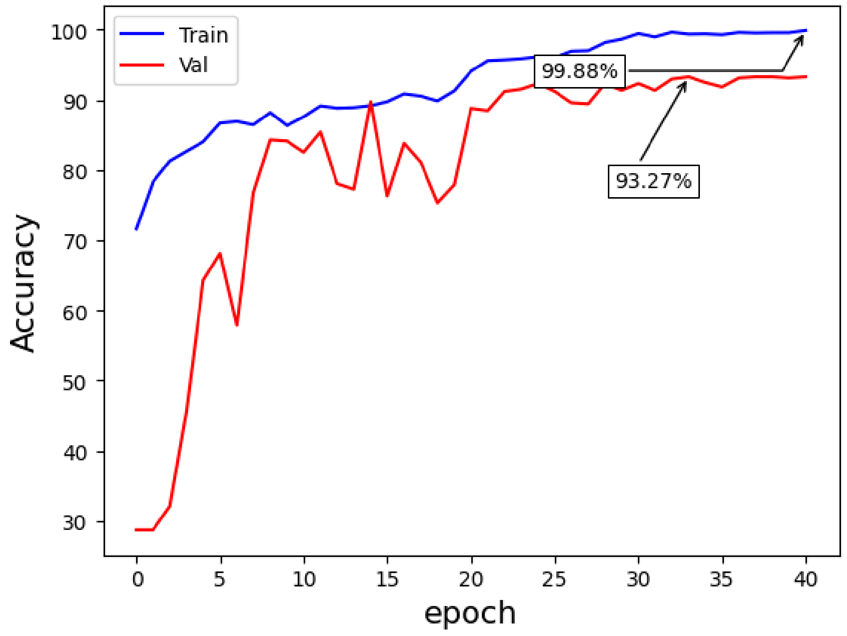
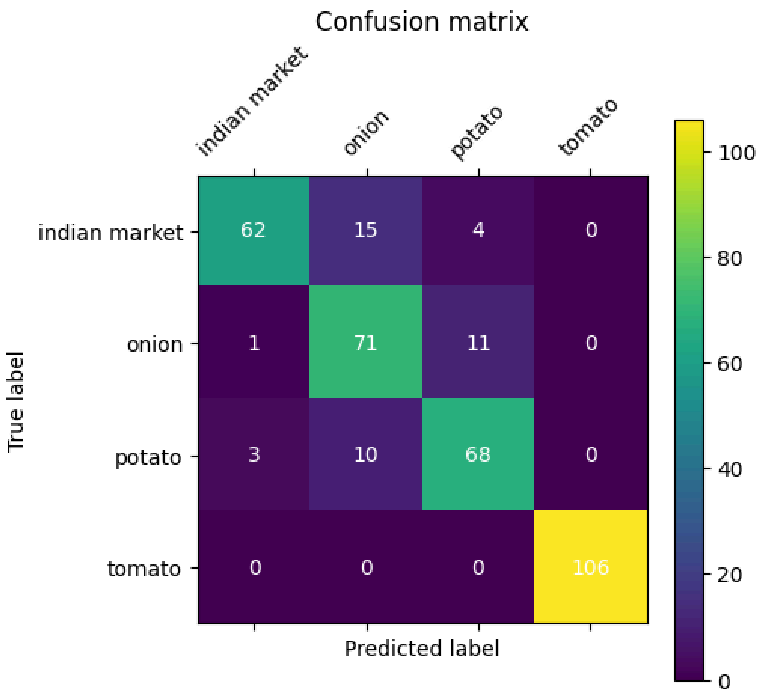


Figure 3.11: Plot for Accuracy vs. Epoch

4. Confusion Matrix



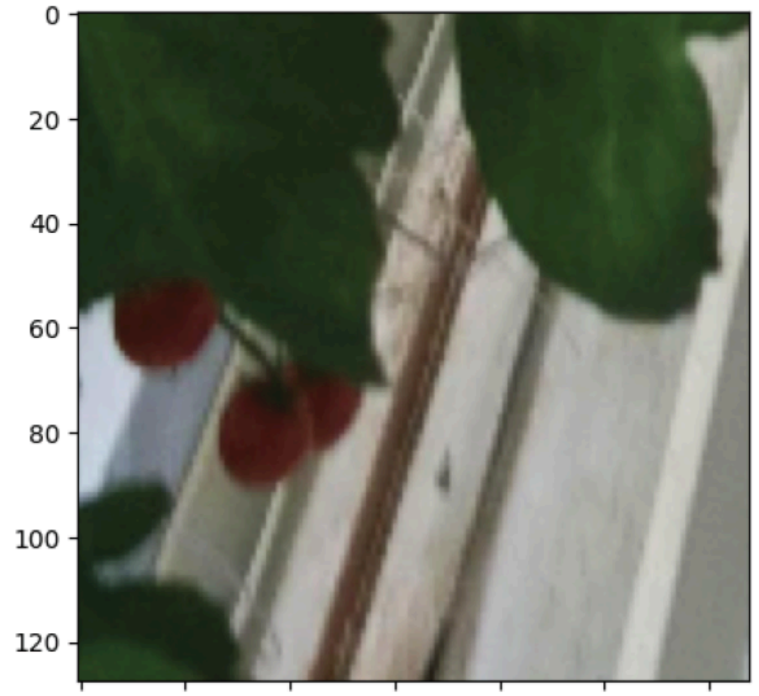
	indian market	onion	potato	tomato
indian market	62	15	4	0
onion	1	71	11	0
potato	3	10	68	0
tomato	0	0	0	106

Figure 3.12: Confusion Matrix after CNN Classification

5. Model Parameters and Performance Metrics

```
Total params: 462,436               Test accuracy: 0.85
Trainable params: 460,932                Precision: 0.86
Non-trainable params: 1,504              Recall: 0.85
Loss: 0.6415
```

6. Result on Random Datapoint

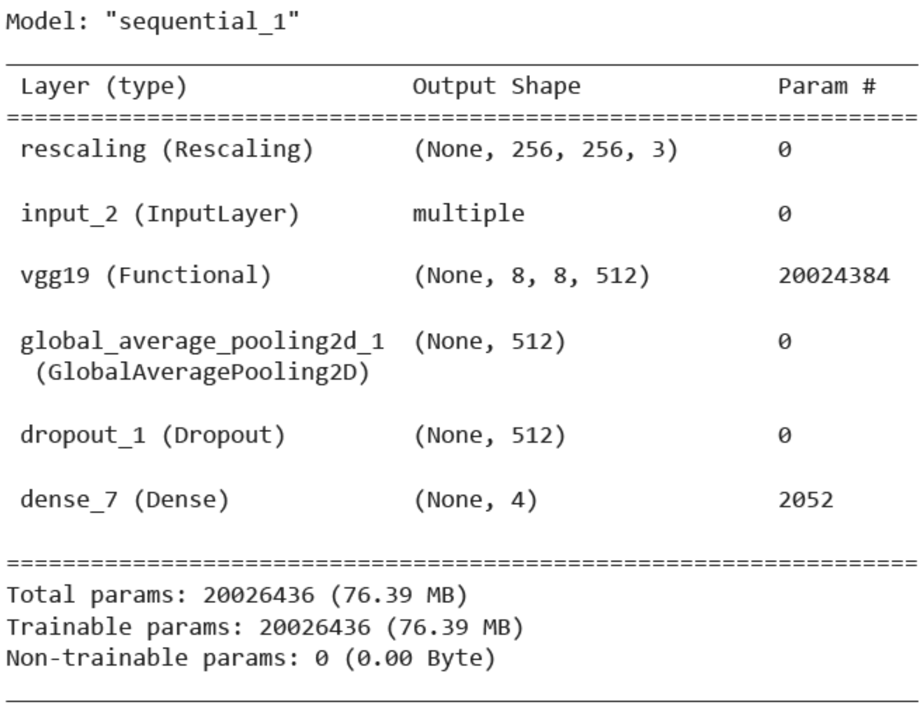


```
Actual Label: tomato
Image Predicted as:  tomato
```

Figure 3.13: Result from CNN Model on a Random Datapoint

- **Pretrained Model: VGG-19**

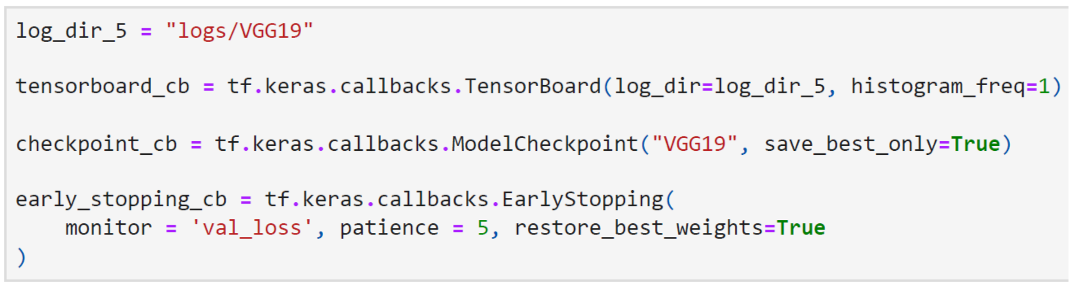
1. VGG-19 Model Summary



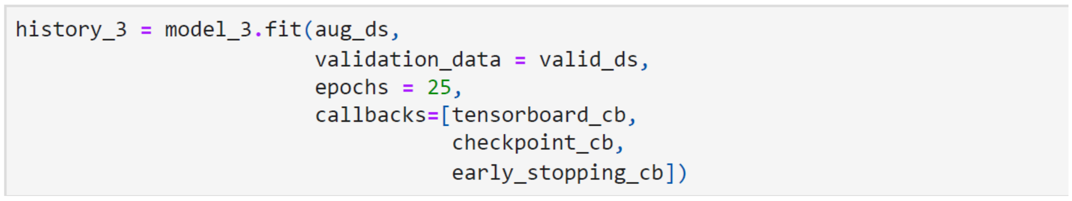
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 256, 256, 3)       0

 input_2 (InputLayer)        multiple                  0

 vgg19 (Functional)          (None, 8, 8, 512)         20024384

 global_average_pooling2d_1  (None, 512)               0
  (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 512)               0

 dense_7 (Dense)             (None, 4)                 2052

=================================================================
Total params: 20026436 (76.39 MB)
Trainable params: 20026436 (76.39 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.14: VGG-19 Model Summary

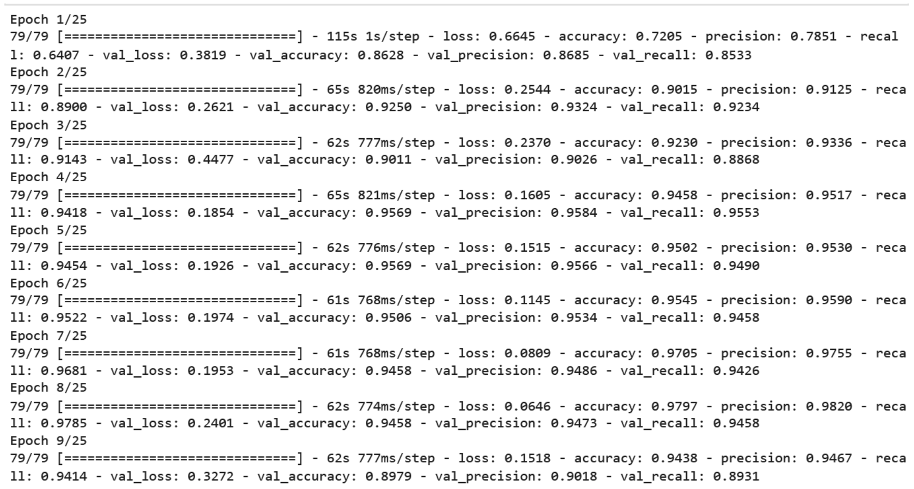
2. VGG-19 Model Training



```
log_dir_5 = "logs/VGG19"

tensorboard_cb = tf.keras.callbacks.TensorBoard(log_dir=log_dir_5, histogram_freq=1)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("VGG19", save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(
    monitor = 'val_loss', patience = 5, restore_best_weights=True
)
```



```
history_3 = model_3.fit(aug_ds,
                        validation_data = valid_ds,
                        epochs = 25,
                        callbacks=[tensorboard_cb,
                                   checkpoint_cb,
                                   early_stopping_cb])
```



```
Epoch 1/25
79/79 [==============================] - 115s 1s/step - loss: 0.6645 - accuracy: 0.7205 - precision: 0.7851 - recal
l: 0.6407 - val_loss: 0.3819 - val_accuracy: 0.8628 - val_precision: 0.8685 - val_recall: 0.8533
Epoch 2/25
79/79 [==============================] - 65s 820ms/step - loss: 0.2544 - accuracy: 0.9015 - precision: 0.9125 - reca
ll: 0.8900 - val_loss: 0.2621 - val_accuracy: 0.9250 - val_precision: 0.9324 - val_recall: 0.9234
Epoch 3/25
79/79 [==============================] - 62s 777ms/step - loss: 0.2370 - accuracy: 0.9230 - precision: 0.9336 - reca
ll: 0.9143 - val_loss: 0.4477 - val_accuracy: 0.9011 - val_precision: 0.9026 - val_recall: 0.8868
Epoch 4/25
79/79 [==============================] - 65s 821ms/step - loss: 0.1605 - accuracy: 0.9458 - precision: 0.9517 - reca
ll: 0.9418 - val_loss: 0.1854 - val_accuracy: 0.9569 - val_precision: 0.9584 - val_recall: 0.9553
Epoch 5/25
79/79 [==============================] - 62s 776ms/step - loss: 0.1515 - accuracy: 0.9502 - precision: 0.9530 - reca
ll: 0.9454 - val_loss: 0.1926 - val_accuracy: 0.9569 - val_precision: 0.9566 - val_recall: 0.9490
Epoch 6/25
79/79 [==============================] - 61s 768ms/step - loss: 0.1145 - accuracy: 0.9545 - precision: 0.9590 - reca
ll: 0.9522 - val_loss: 0.1974 - val_accuracy: 0.9506 - val_precision: 0.9534 - val_recall: 0.9458
Epoch 7/25
79/79 [==============================] - 61s 768ms/step - loss: 0.0809 - accuracy: 0.9705 - precision: 0.9755 - reca
ll: 0.9681 - val_loss: 0.1953 - val_accuracy: 0.9458 - val_precision: 0.9486 - val_recall: 0.9426
Epoch 8/25
79/79 [==============================] - 62s 774ms/step - loss: 0.0646 - accuracy: 0.9797 - precision: 0.9820 - reca
ll: 0.9785 - val_loss: 0.2401 - val_accuracy: 0.9458 - val_precision: 0.9473 - val_recall: 0.9458
Epoch 9/25
79/79 [==============================] - 62s 777ms/step - loss: 0.1518 - accuracy: 0.9438 - precision: 0.9467 - reca
ll: 0.9414 - val_loss: 0.3272 - val_accuracy: 0.8979 - val_precision: 0.9018 - val_recall: 0.8931
```

Figure 3.15: VGG-19 Model Training

3. Accuracy vs. Epoch

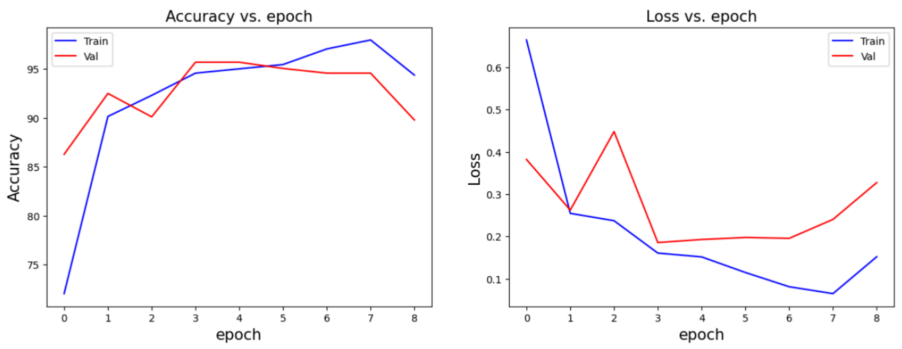


Figure 3.16: Plot for Accuracy and Loss vs. Epoch

4. Confusion Matrix

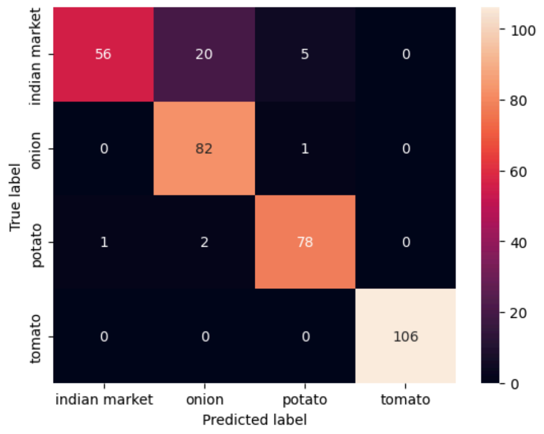
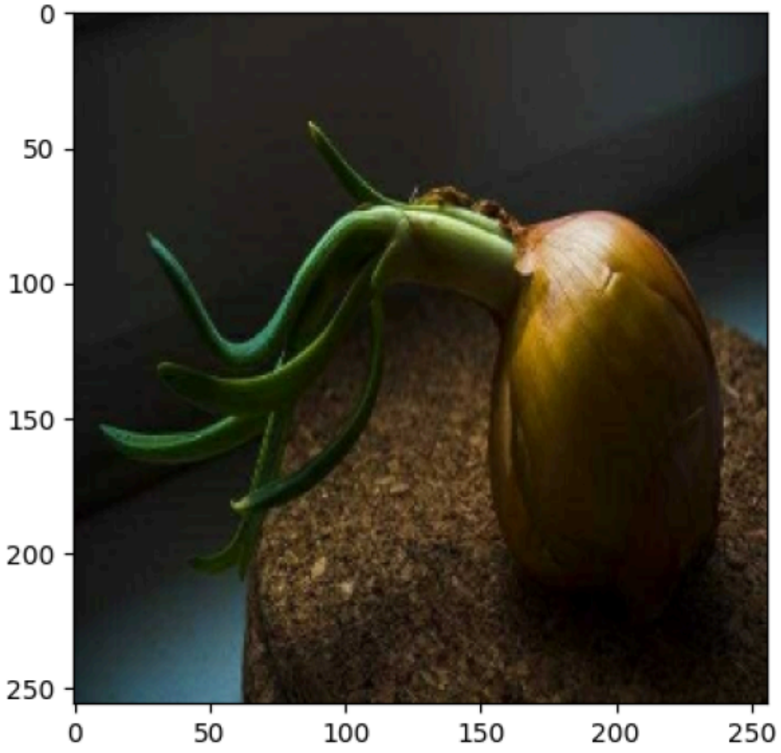


Figure 3.17: Confusion Matrix after VGG19 Classification

5. Model Parameters and Performance Metrics

```
Total params: 20,026,436        Test-accuracy: 0.9402
Trainable params: 20,026,436    Precision: 0.9398
Non-trainable params: 0         Recall: 0.9345
Loss: 0.2026
```

6. Result on Random Datapoint

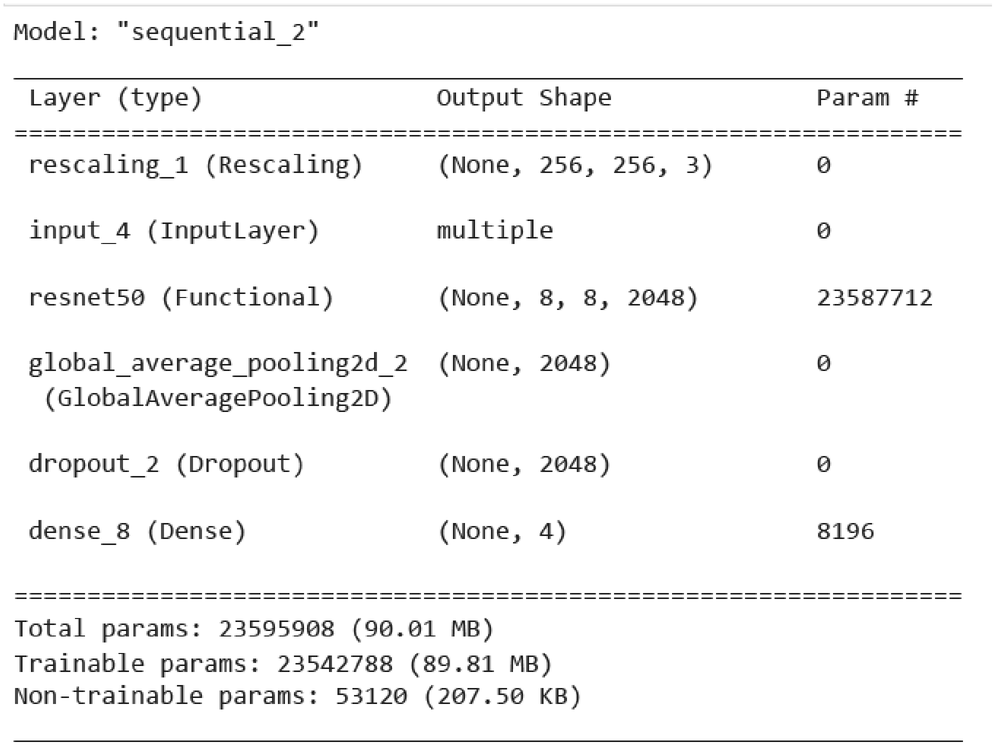


```
Actual Label: onion
Image Predicted as:  onion
```

Figure 3.18: Result from VGG-19 Model on a Random Datapoint

- **Pretrained Model: ResNet50**

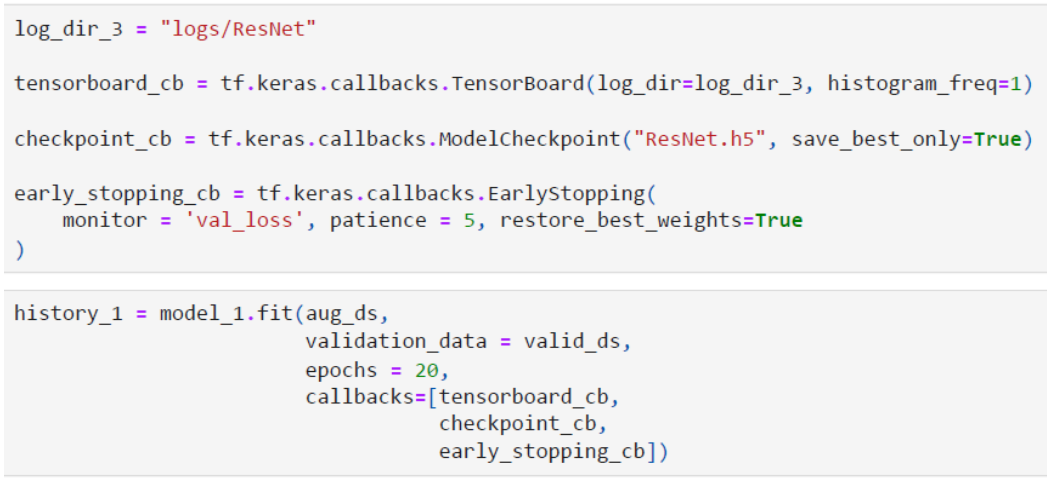
1. ResNet50 Model Summary



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 256, 256, 3)       0

 input_4 (InputLayer)        multiple                  0

 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d_2  (None, 2048)              0
  (GlobalAveragePooling2D)

 dropout_2 (Dropout)         (None, 2048)              0

 dense_8 (Dense)             (None, 4)                 8196

=================================================================
Total params: 23595908 (90.01 MB)
Trainable params: 23542788 (89.81 MB)
Non-trainable params: 53120 (207.50 KB)
_________________________________________________________________
```

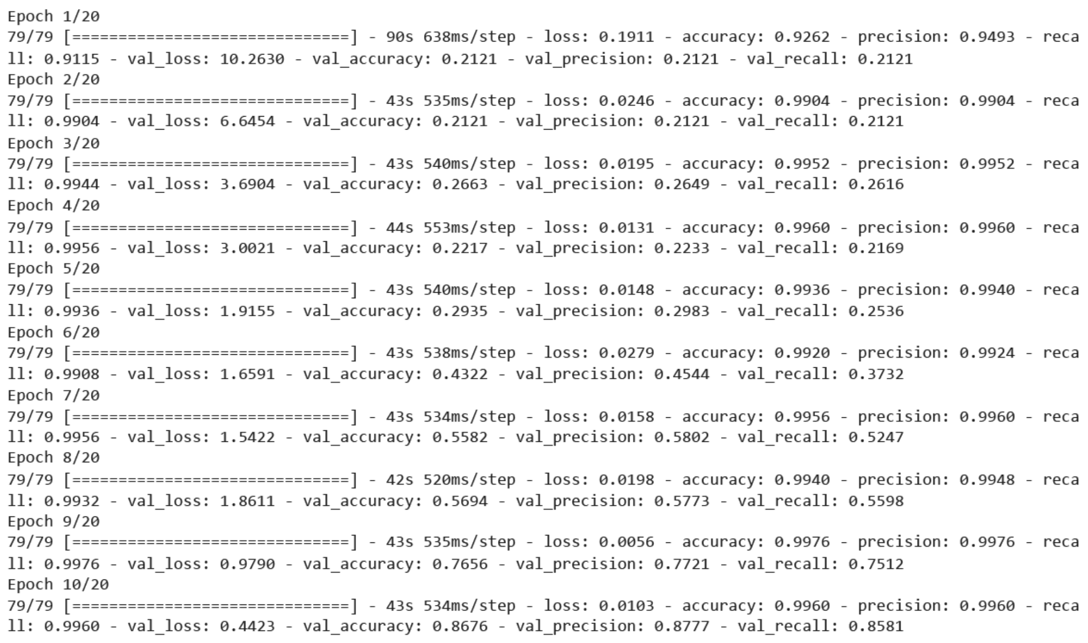
Figure 3.19: ResNet50 Model Summary

2. ResNet50 Model Training



```
log_dir_3 = "logs/ResNet"

tensorboard_cb = tf.keras.callbacks.TensorBoard(log_dir=log_dir_3, histogram_freq=1)

checkpoint_cb = tf.keras.callbacks.ModelCheckpoint("ResNet.h5", save_best_only=True)

early_stopping_cb = tf.keras.callbacks.EarlyStopping(
    monitor = 'val_loss', patience = 5, restore_best_weights=True
)
```

```
history_1 = model_1.fit(aug_ds,
                        validation_data = valid_ds,
                        epochs = 20,
                        callbacks=[tensorboard_cb,
                                   checkpoint_cb,
                                   early_stopping_cb])
```



```
Epoch 1/20
79/79 [==============================] - 90s 638ms/step - loss: 0.1911 - accuracy: 0.9262 - precision: 0.9493 - reca
ll: 0.9115 - val_loss: 10.2630 - val_accuracy: 0.2121 - val_precision: 0.2121 - val_recall: 0.2121
Epoch 2/20
79/79 [==============================] - 43s 535ms/step - loss: 0.0246 - accuracy: 0.9904 - precision: 0.9904 - reca
ll: 0.9904 - val_loss: 6.6454 - val_accuracy: 0.2121 - val_precision: 0.2121 - val_recall: 0.2121
Epoch 3/20
79/79 [==============================] - 43s 540ms/step - loss: 0.0195 - accuracy: 0.9952 - precision: 0.9952 - reca
ll: 0.9944 - val_loss: 3.6904 - val_accuracy: 0.2663 - val_precision: 0.2649 - val_recall: 0.2616
Epoch 4/20
79/79 [==============================] - 44s 553ms/step - loss: 0.0131 - accuracy: 0.9960 - precision: 0.9960 - reca
ll: 0.9956 - val_loss: 3.0021 - val_accuracy: 0.2217 - val_precision: 0.2233 - val_recall: 0.2169
Epoch 5/20
79/79 [==============================] - 43s 540ms/step - loss: 0.0148 - accuracy: 0.9936 - precision: 0.9940 - reca
ll: 0.9936 - val_loss: 1.9155 - val_accuracy: 0.2935 - val_precision: 0.2983 - val_recall: 0.2536
Epoch 6/20
79/79 [==============================] - 43s 538ms/step - loss: 0.0279 - accuracy: 0.9920 - precision: 0.9924 - reca
ll: 0.9908 - val_loss: 1.6591 - val_accuracy: 0.4322 - val_precision: 0.4544 - val_recall: 0.3732
Epoch 7/20
79/79 [==============================] - 43s 534ms/step - loss: 0.0158 - accuracy: 0.9956 - precision: 0.9960 - reca
ll: 0.9956 - val_loss: 1.5422 - val_accuracy: 0.5582 - val_precision: 0.5802 - val_recall: 0.5247
Epoch 8/20
79/79 [==============================] - 42s 520ms/step - loss: 0.0198 - accuracy: 0.9940 - precision: 0.9948 - reca
ll: 0.9932 - val_loss: 1.8611 - val_accuracy: 0.5694 - val_precision: 0.5773 - val_recall: 0.5598
Epoch 9/20
79/79 [==============================] - 43s 535ms/step - loss: 0.0056 - accuracy: 0.9976 - precision: 0.9976 - reca
ll: 0.9976 - val_loss: 0.9790 - val_accuracy: 0.7656 - val_precision: 0.7721 - val_recall: 0.7512
Epoch 10/20
79/79 [==============================] - 43s 534ms/step - loss: 0.0103 - accuracy: 0.9960 - precision: 0.9960 - reca
ll: 0.9960 - val_loss: 0.4423 - val_accuracy: 0.8676 - val_precision: 0.8777 - val_recall: 0.8581
```

Figure 3.20: ResNet50 Model Training

3. Accuracy vs. Epoch

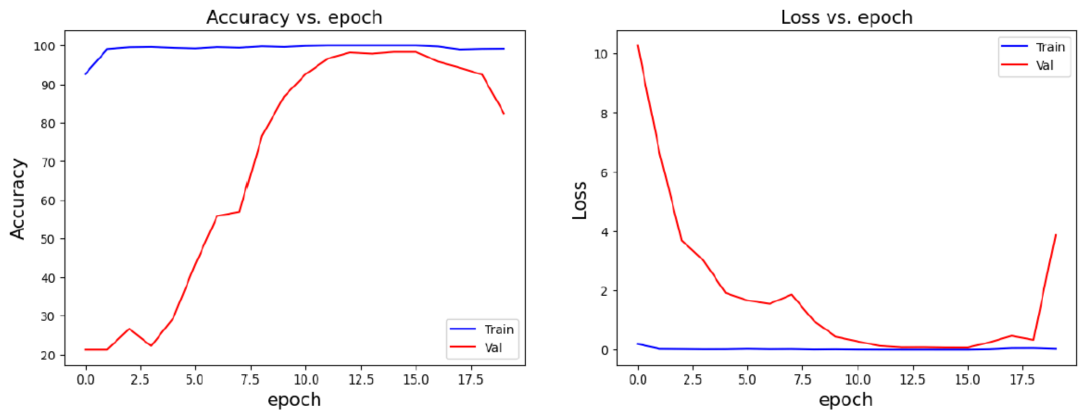


Figure 3.21: Plot for Accuracy and Loss vs. Epoch

4. Confusion Matrix

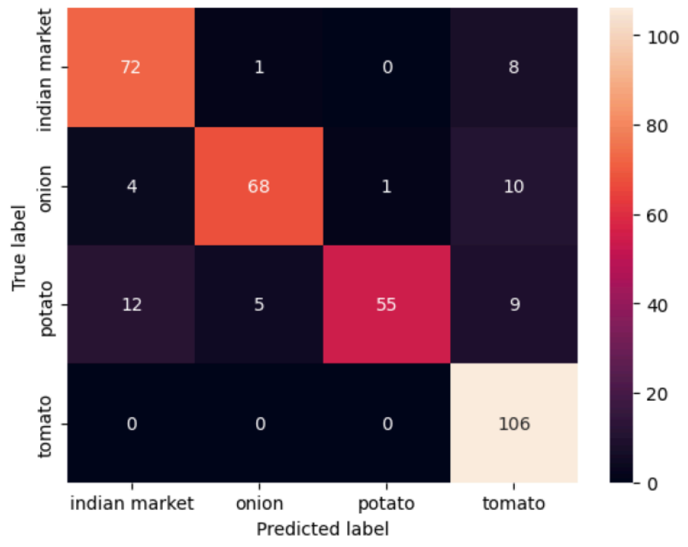
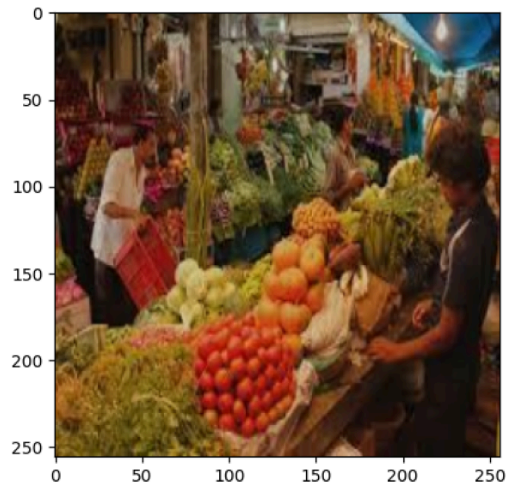


Figure 3.22: Confusion Matrix after VGG19 Classification

5. Model Parameters and Performance Metrics

```
Total params: 23,595,908         Test-accuracy: 0.9630
Trainable params: 23,542,788     Precision: 0.9628
Non-trainable params: 53,120     Recall: 0.9573
Loss : 0.1591
```

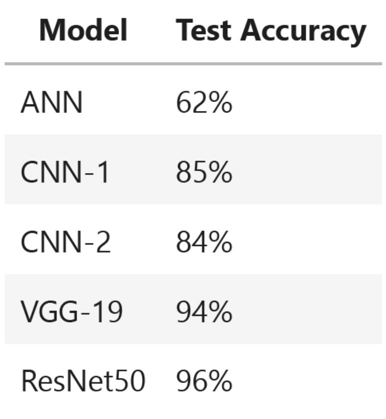
6. Result on Random Datapoint



```
Actual Label: onion
Image Predicted as:  tomato
```

Figure 3.23: Result from ResNet50Model on a Random Datapoint

- **Final Summary** across all Models



Model	Test Accuracy
ANN	62%
CNN-1	85%
CNN-2	84%
VGG-19	94%
ResNet50	96%

Figure 3.24: Summary of All Models

Insights and Recommendations

1. Tensorflow Functional API has been used to build the models (starting from generic ANN followed by Convolutional NeuralNetwork, along with Pre-trained VGG19, ResNet50).

2. To prepare the images for Training, Validation and Test, Keras **ImageDataGenerator** is used which would preprocess the data by resizing it and normalizing. It provides a quick and easy way to augment images. It provides a host of different augmentation techniques like Standardization, Rotation, Shifts, Flips, Brightness change, and many more. The main benefit of using the Keras ImageDataGenerator class is that it is designed to provide real-time data augmentation. Meaning it is generating augmented images on the fly while your model is still in the training stage.

3. Here we have built three models:

    a. **Convolution Neural Network** from scratch
    b. **VGG-19** model along with few more Convolution and Dense layers (which is nothing but Transfer learning).
    c. **ResNet50**, another deep Convolutional Neural Network (CNN). It is a variant of the popular ResNet architecture, which stands for 'Residual Network'. The '50' in the name refers to the number of layers in the network, which is 50 layers deep.It is a type of residual neural network, which has 48 Convolutional layers, one MaxPool layer, and one average Pool layer.

4. While training the model, various regularization techniques like dropouts and batch normalization helped in improving the model performance. The use of various callback techniques like:

   a. EarlyStopping
   b. LearningRateScheduler
   c. ModelCheckpoint
   d. TensorBoard

5. Out of all the above, Learning Rate Scheduler helped in boosting the model performance to a large extent.

6. The performance of the Convolution Neural Network from scratch got a lower Accuracy and F1 score. Whereas, the model in which Transfer learning has been used, could provide better results in a lesser number of epochs.

7. There was still a slight overfit in all the models.

8. Based on the Confusion Matrix, we can conclude that the model performed best on Tomato Dataset.

9. The model gets confused sometimes between Onion and Potato, and between Indian Market and Onion.

10. The model performance can be improved by improving the quality and quantity of data, along with more advanced architectures.

Chapter 4 : Business Case Study 4 — **Article Categorization using NLP**

Problem Description

The Gurugram-based company aims to revolutionize the way Indians perceive finance, business, and capital market investment, by giving it a boost through artificial intelligence (AI) and machine learning (ML). They're on a mission to reinvent financial literacy for Indians, where financial awareness is driven by smart information discovery and engagement with peers. Through their smart content discovery and contextual engagement, the company is simplifying business, finance, and investment for millennials and first-time investors.

The goal of this project is to use a bunch of news articles extracted from the companies' internal database and categorize them into several categories like politics, technology, sports, business and entertainment based on their content. We will use natural language processing to create & compare at least three different models.

Business Questions to be answered from Analysis

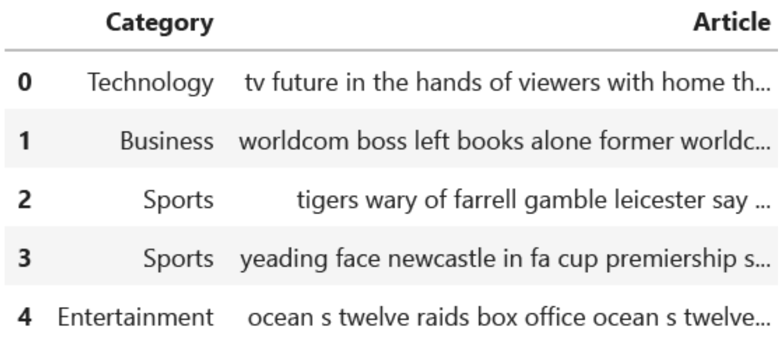
- **Impact on Financial Literacy:**
  - How can categorizing news articles into specific categories improve financial literacy and awareness among millennials and first-time investors?
  - What type of content (news category) is most engaging for users interested in finance and investment?
- **User Engagement and Experience:**
  - How does the categorization of news articles enhance user engagement and interaction on the platform?
  - Are there specific categories that drive more engagement or content consumption among users?
- **Content Personalization:**
  - How can the categorized news articles be used to provide personalized content recommendations to users based on their interests and past engagement?
  - What are the key factors that should be considered when developing personalized content feeds for different user segments?
- **Content Diversity and Inclusivity:**
  - How well does the system cover a diverse range of topics within each category, and does it include content from various perspectives?
  - Are there any biases in the categorization process that could limit the diversity of content presented to users?

- **Market Trends and Insights:**
  - How can the analysis of categorized news articles provide insights into current market trends and investor sentiment?
  - Can the system detect emerging topics or trends in the finance and business sectors that are relevant to users?
- **AI/ML Model Selection and Optimization:**
  - What are the trade-offs between different natural language processing (NLP) models used in this project, such as accuracy, computational efficiency, and scalability?
  - How can the selected model(s) be optimized for real-time content categorization and scaling as the volume of articles grows?
- **Integration and Deployment:**
  - How can the categorized news content be seamlessly integrated into the company's existing platforms and user interfaces?
  - What are the technical and operational challenges involved in deploying the NLP models and ensuring consistent performance?
- **Content Quality and Relevance:**
  - How does the system ensure the quality and relevance of the news articles being categorized and presented to users?
  - What measures are in place to update and refine the categorization process as new types of content and topics emerge?
- **Business Growth and Monetization:**
  - How can the categorized news content contribute to the company's growth, particularly in terms of user acquisition, retention, and monetization?
  - Are there opportunities to monetize the content categorization and personalization services, such as through premium subscriptions or partnerships with financial service providers?
- **Compliance and Ethical Considerations:**
  - What ethical considerations need to be addressed in the automated categorization of news content, particularly regarding misinformation and content bias?
  - How can the company ensure compliance with data privacy and content regulation laws while using AI/ML for content categorization?
- **Future Research and Development:**
  - What are the potential future directions for research and development in the area of AI-driven content categorization and personalization?
  - How can the company leverage advancements in AI/ML and NLP to enhance the accuracy and scope of its content discovery and engagement platform?

These questions aim to explore the practical applications, challenges, and opportunities associated with using AI and NLP for news article categorization, as well as their broader implications for user engagement, financial literacy, and business growth.

Analysis

The snapshot of the data shared by the respective News Article Categorizer Company looks like this:



	Category	Article
0	Technology	tv future in the hands of viewers with home th...
1	Business	worldcom boss left books alone former worldc...
2	Sports	tigers wary of farrell gamble leicester say ...
3	Sports	yeading face newcastle in fa cup premiership s...
4	Entertainment	ocean s twelve raids box office ocean s twelve...

Figure 4.1: Sample Data Shared by the Respective News Article Categorizer Company

- Basic Exploratory Data Analysis

    1. Distribution of News Categories

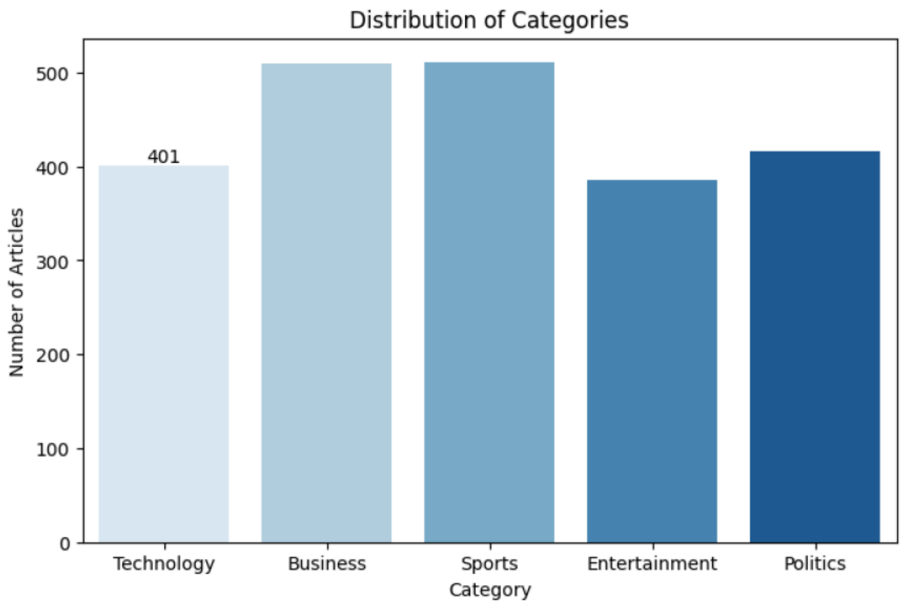


Figure 4.2: Barplot Showing Different News Category Across the No. of Articles

2. **Word Clouds:**

a. Sports

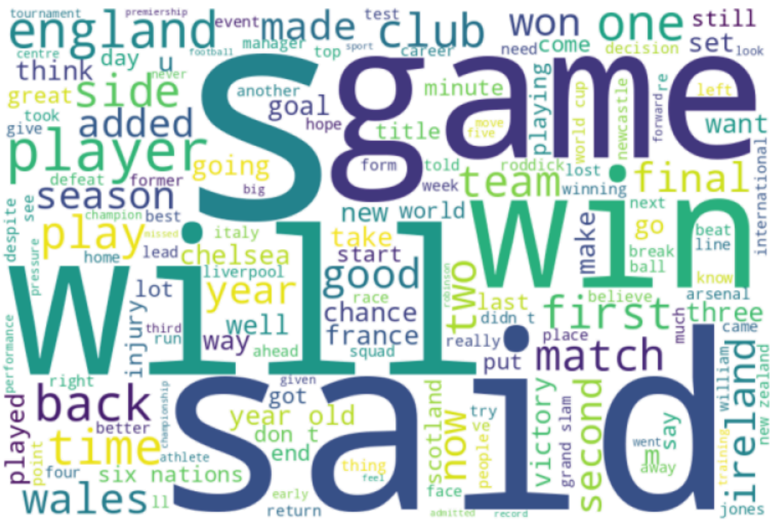


Figure 4.3: Word Cloud of the News Category 'Sports'

b. Business

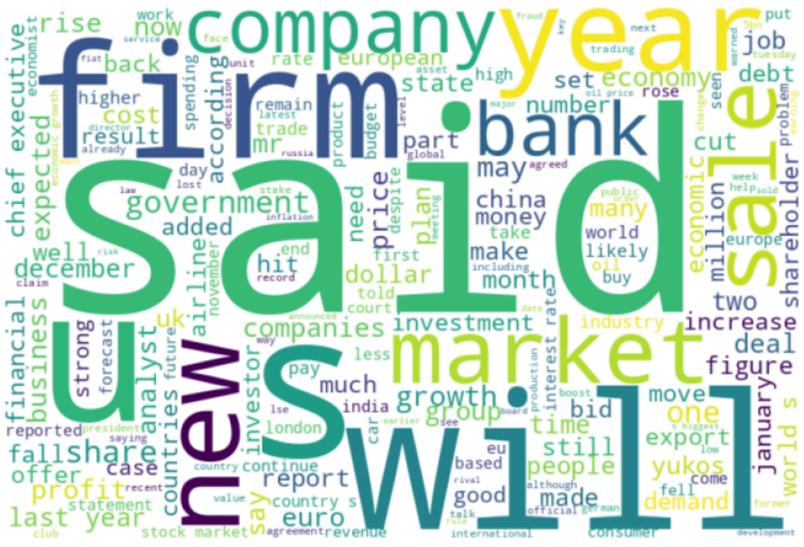


Figure 4.4: Word Cloud of the News Category 'Business'

c. Technology

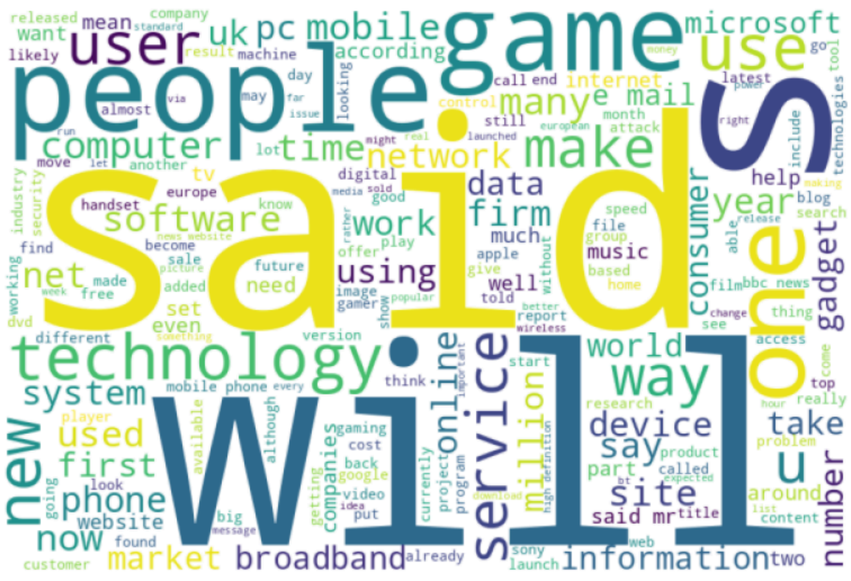


Figure 4.5: Word Cloud of the News Category 'Technology'

d. Entertainment

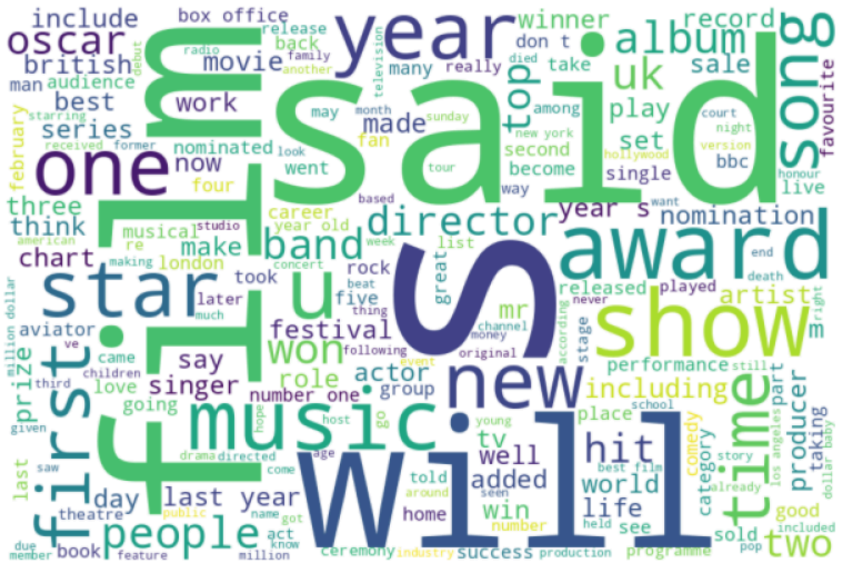


Figure 4.6: Word Cloud of the News Category 'Entertainment'

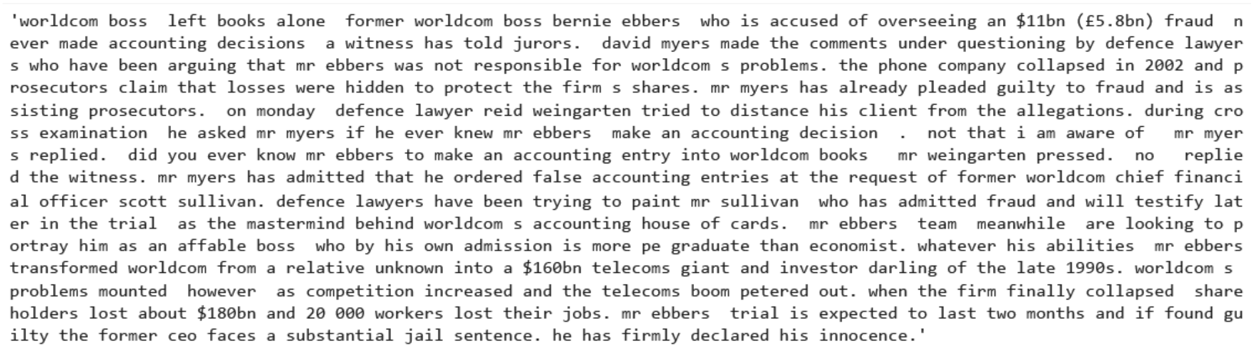
3. **Basic Article Preprocessing**

    a. Tokenization of the Words: Split text into individual tokens (words or subwords).

    b. Stopwords Removal: Remove common words (stop words) that may not carry significant meaning.

    c. Lemmatization: Reduce words to their base or dictionary form.

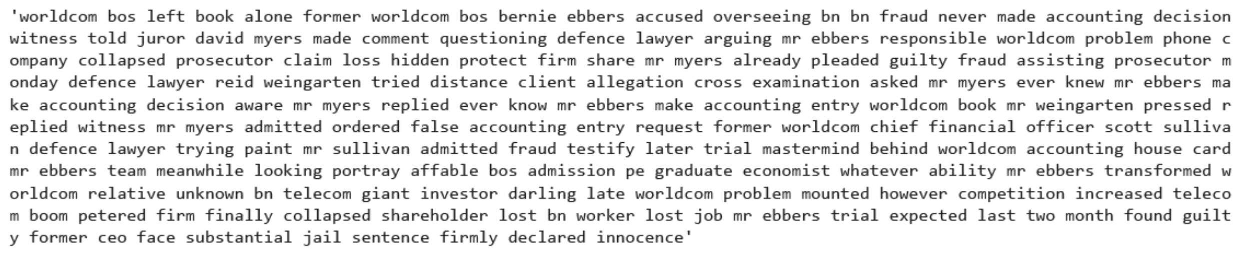
- **Before Preprocessing:**



```
'worldcom boss  left books alone  former worldcom boss bernie ebbers  who is accused of overseeing an $11bn (£5.8bn) fraud  n
ever made accounting decisions  a witness has told jurors.  david myers made the comments under questioning by defence lawyer
s who have been arguing that mr ebbers was not responsible for worldcom s problems. the phone company collapsed in 2002 and p
rosecutors claim that losses were hidden to protect the firm s shares. mr myers has already pleaded guilty to fraud and is as
sisting prosecutors.  on monday  defence lawyer reid weingarten tried to distance his client from the allegations. during cro
ss examination  he asked mr myers if he ever knew mr ebbers  make an accounting decision  .  not that i am aware of   mr myer
s replied.  did you ever know mr ebbers to make an accounting entry into worldcom books   mr weingarten pressed.  no   replie
d the witness. mr myers has admitted that he ordered false accounting entries at the request of former worldcom chief financi
al officer scott sullivan. defence lawyers have been trying to paint mr sullivan  who has admitted fraud and will testify lat
er in the trial  as the mastermind behind worldcom s accounting house of cards.  mr ebbers  team  meanwhile  are looking to p
ortray him as an affable boss  who by his own admission is more pe graduate than economist. whatever his abilities  mr ebbers
transformed worldcom from a relative unknown into a $160bn telecoms giant and investor darling of the late 1990s. worldcom s
problems mounted  however  as competition increased and the telecoms boom petered out. when the firm finally collapsed  share
holders lost about $180bn and 20 000 workers lost their jobs. mr ebbers  trial is expected to last two months and if found gu
ilty the former ceo faces a substantial jail sentence. he has firmly declared his innocence.'
```

Figure 4.7: Before Basic Text Preprocessing

- **After Preprocessing:**



```
'worldcom bos left book alone former worldcom bos bernie ebbers accused overseeing bn bn fraud never made accounting decision
witness told juror david myers made comment questioning defence lawyer arguing mr ebbers responsible worldcom problem phone c
ompany collapsed prosecutor claim loss hidden protect firm share mr myers already pleaded guilty fraud assisting prosecutor m
onday defence lawyer reid weingarten tried distance client allegation cross examination asked mr myers ever knew mr ebbers ma
ke accounting decision aware mr myers replied ever know mr ebbers make accounting entry worldcom book mr weingarten pressed r
eplied witness mr myers admitted ordered false accounting entry request former worldcom chief financial officer scott sulliva
n defence lawyer trying paint mr sullivan admitted fraud testify later trial mastermind behind worldcom accounting house card
mr ebbers team meanwhile looking portray affable bos admission pe graduate economist whatever ability mr ebbers transformed w
orldcom relative unknown bn telecom giant investor darling late worldcom problem mounted however competition increased teleco
m boom petered firm finally collapsed shareholder lost bn worker lost job mr ebbers trial expected last two month found guilt
y former ceo face substantial jail sentence firmly declared innocence'
```

Figure 4.8: After Basic Text Preprocessing

- Implementation of **Classical Machine Learning** Model

    1. Naive Bayes:

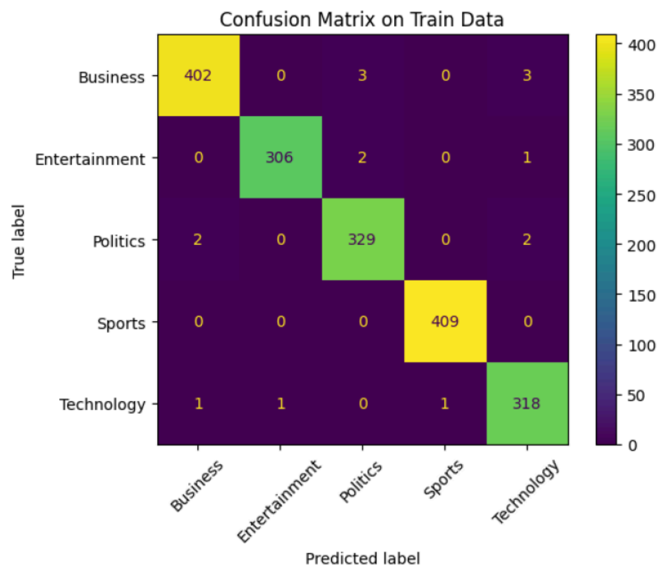


Figure 4.9: Confusion Matrix on Naive Bayes Training Data

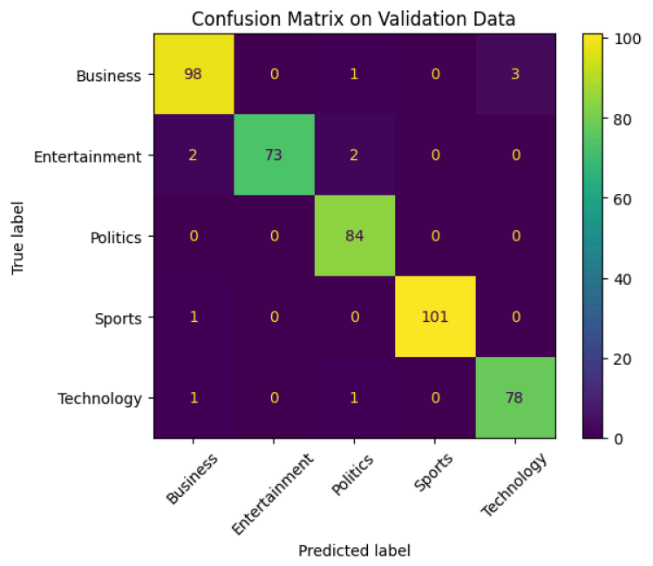


Figure 4.10: Confusion Matrix on Naive Bayes Validation Data

```
Accuracy: 0.97         Recall: 0.97

Precision: 0.97        F1-Score: 0.97
```

2. Logistic Regression:

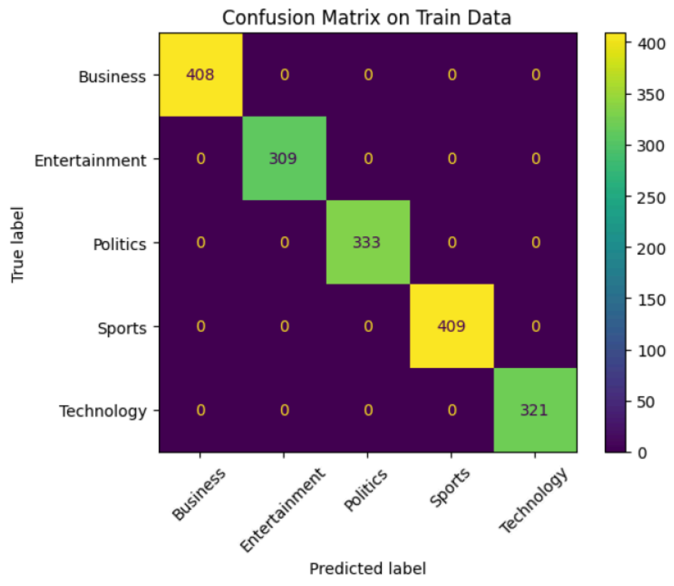


Figure 4.11: Confusion Matrix on Logistic Regression Training Data

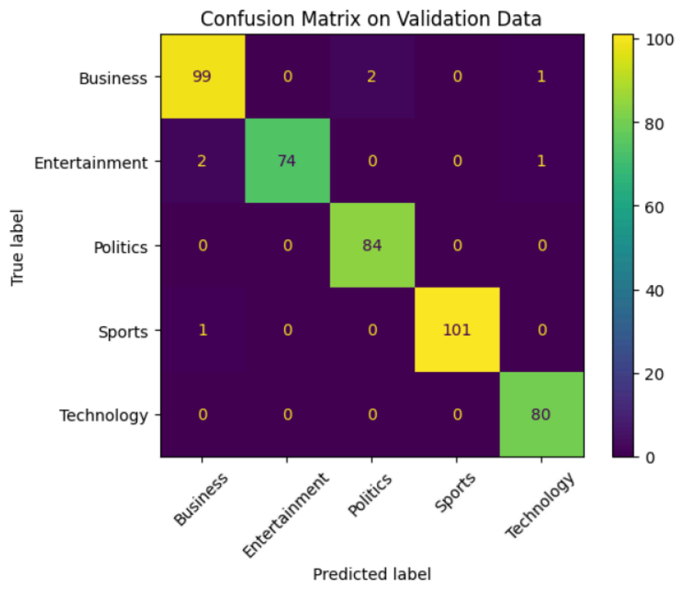


Figure 4.12: Confusion Matrix on Logistic Regression Validation Data

```
Accuracy: 0.98          Recall: 0.98

Precision: 0.98         F1-Score: 0.98
```

3. Decision Tree:

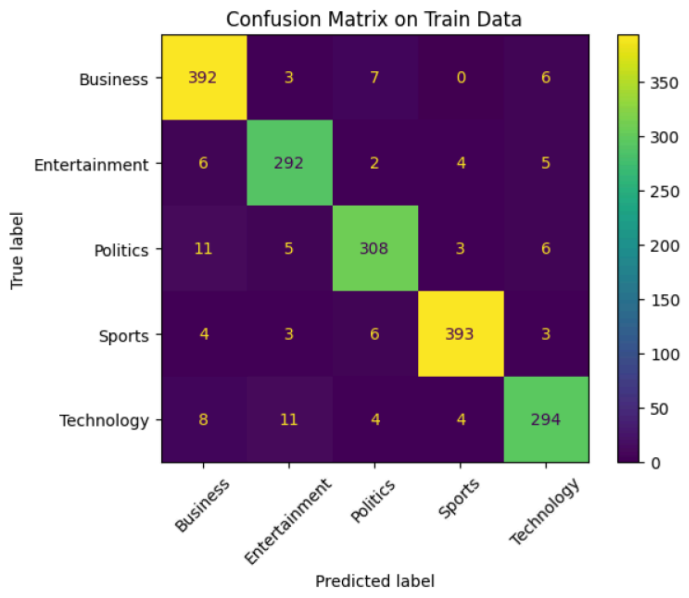


Figure 4.13: Confusion Matrix on Decision Tree Training Data

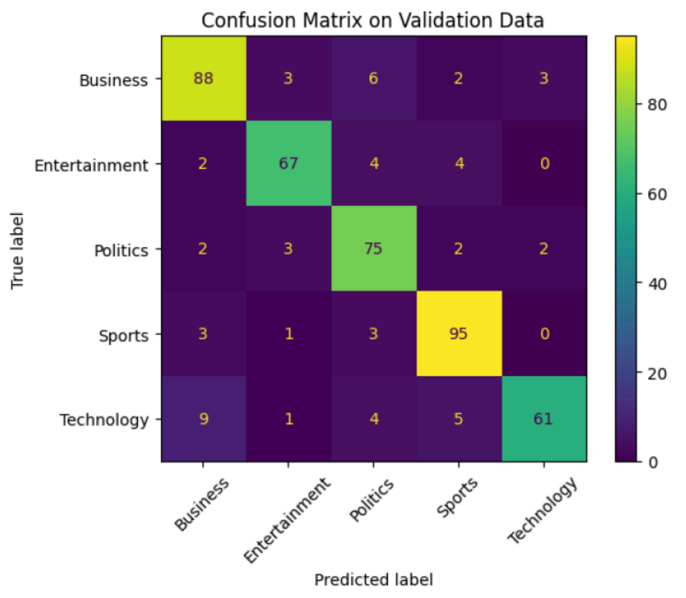


Figure 4.14: Confusion Matrix on Decision Tree Validation Data

```
Accuracy: 0.87          Recall: 0.87

Precision: 0.87         F1-Score: 0.87
```

4. K-Nearest Neighbours:

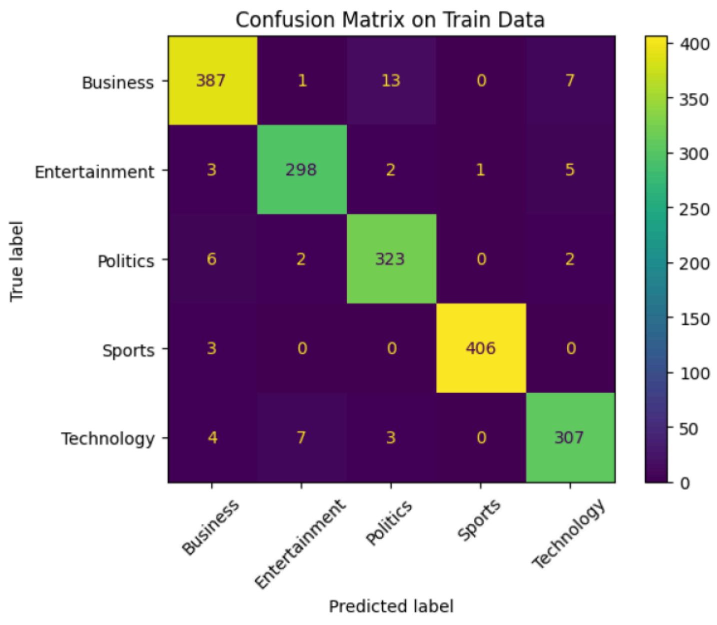


Figure 4.15: Confusion Matrix on K-Nearest Neighbours Training Data

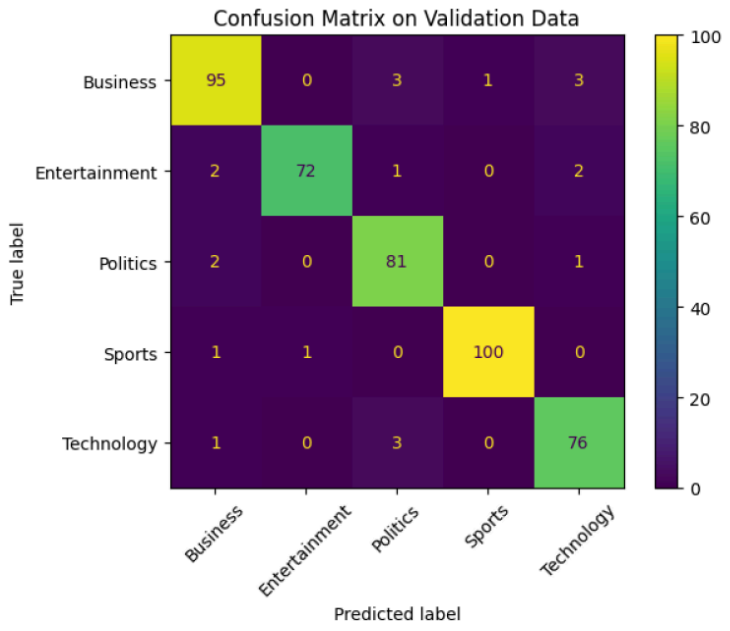


Figure 4.16: Confusion Matrix on K-Nearest Neighbours Validation Data

```
Accuracy: 0.95          Recall: 0.95

Precision: 0.95         F1-Score: 0.95
```

5. Random Forest Classifier:

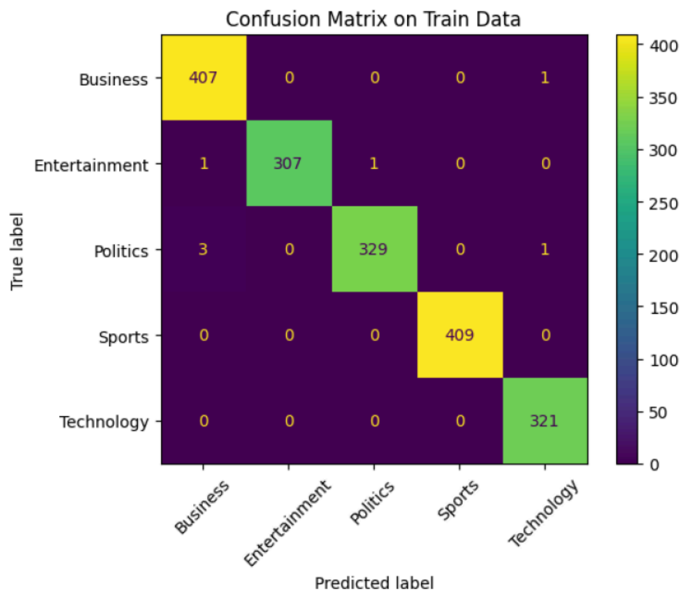


Figure 4.17: Confusion Matrix on Random Forest Classifier Training Data

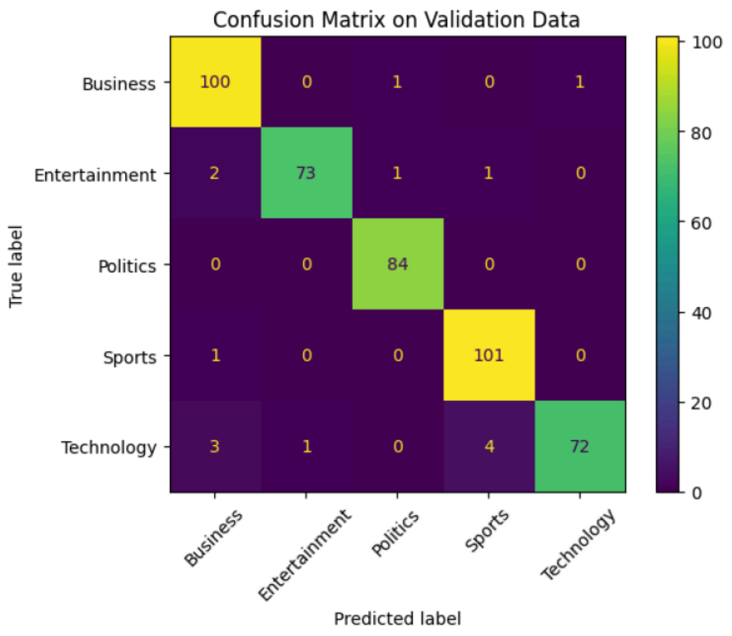


Figure 4.18: Confusion Matrix on Random Forest Classifier Validation Data

```
Accuracy: 0.97          Recall: 0.97

Precision: 0.97         F1-Score: 0.97
```

Insights and Recommendations

- **Business Insights**

1. **Article Distribution and Content Focus:**
   - The dataset contains 2225 news articles, with 2117 being unique, indicating a slight presence of duplicate or highly similar articles. The majority of these articles fall under the Sports category, suggesting a strong focus or high volume of content in this area. The Technology category, with 401 articles, also represents a significant portion of the dataset, reflecting substantial interest in tech-related news.
2. **Model Performance:**
   - The Naive Bayes Classifier has emerged as the best-performing model based on metrics such as train and test accuracy, precision, recall, and minimal overfitting. This suggests that Naive Bayes is particularly effective in distinguishing between the different news categories in the dataset. The Random Forest model, as a powerful ensemble technique, also performs well, indicating that complex relationships and interactions among features are being captured effectively.
3. **Data Quality and Diversity:**
   - The presence of 2117 unique articles out of 2225 total entries points to a relatively high data quality, with a low level of redundancy. However, the dominance of Sports articles may indicate a potential imbalance in content diversity, which could affect user engagement and the comprehensiveness of news coverage offered by the platform.

- **Recommendations**

1. **Content Diversification:**
    - **Increase Content Variety:** To provide a more balanced news experience, it is recommended to expand the dataset with additional articles from underrepresented categories, such as Technology, Business, Politics, and Entertainment. This will not only cater to a broader audience but also enhance the richness and diversity of content available on the platform.
    - **Address Content Imbalance:** Actively source or commission articles in less represented categories to ensure a more even distribution of content. This approach can help attract users with varied interests and reduce over-reliance on a single category, like Sports.
2. **Model Optimization and Deployment:**
    - **Utilize Naive Bayes Classifier:** Given its strong performance, the Naive Bayes Classifier should be prioritized for initial deployment in the news categorization system. Its efficiency and accuracy make it a practical choice for real-time content classification.
    - **Leverage Ensemble Techniques:** Although Naive Bayes performs well, consider integrating the Random Forest model as part of an ensemble approach. This can help improve classification robustness and accuracy by leveraging the strengths of multiple models. An ensemble method can also provide better handling of complex and non-linear relationships in the data.
3. **Data Quality Improvement:**
    - **Eliminate Duplicates:** Implement more rigorous checks to remove duplicate or highly similar articles from the dataset. This can enhance the uniqueness of content and prevent redundancy, leading to a more engaging user experience.
    - **Regular Data Refresh:** Establish a process for regularly updating the dataset with fresh articles, particularly focusing on emerging trends and popular topics across different categories. This will keep the platform's content current and relevant.
4. **User Engagement and Personalization:**
    - **Personalized Content Delivery:** Use the categorization results to offer personalized content recommendations to users. By analyzing user behavior and

preferences, the platform can deliver tailored news feeds that match individual interests, thereby increasing user engagement and retention.

- **Category-Based Notifications:** Consider implementing category-specific notifications or alerts to inform users about the latest articles in their preferred categories. This feature can enhance user experience and encourage more frequent platform visits.

5. **Future Model Exploration and Expansion:**

   - **Explore Additional Models:** While Naive Bayes and Random Forest have shown promise, exploring other models like Support Vector Machines (SVM), Gradient Boosting, or even deep learning approaches may further enhance classification accuracy, especially as the dataset grows in size and complexity.

   - **Multilingual Support:** If the platform aims to cater to a multilingual audience, consider extending the classification models to handle articles in multiple languages. This will involve incorporating language detection and translation capabilities, broadening the platform's reach and inclusivity.

By implementing these recommendations, the company can enhance the effectiveness of its news categorization system, diversify its content offerings, and provide a more engaging and personalized experience for its users. This will not only improve user satisfaction but also support the company's mission of promoting financial literacy and awareness through smart content discovery.

Chapter 5 : Business Case Study 5 — **Named Entity Recognition for Microblogging Site**

Problem Description

The company is a microblogging and social networking service on which users post and interact with messages. Every second, on average, around 6,000 microblogs are posted on it, corresponding to over 350,000 messages / microblogs sent per minute, 500 million microblogs per day. It wants to automatically tag and analyze tweets for better understanding of the trends and topics without being dependent on the hashtags that the users use. Many users do not use hashtags or sometimes use wrong or mis-spelled tags, so they want to completely remove this problem and create a system of recognizing important content of the messages.

Named Entity Recognition (NER) is an important subtask of information extraction that seeks to locate and recognise named entities. We need to train models that will be able to identify the various named entities. Dataset is annotated with 10 fine-grained NER categories: person, geo-location, company, facility, product,music artist, movie, sports team, TV show and other. Dataset was extracted from tweets and is structured in CoNLL format., in English language, containing in Text file format.

Business Questions to be answered from Analysis

Model Performance and Accuracy:

- How accurately can the NER models identify and categorize named entities into the specified categories (person, geo-location, company, facility, product, music artist, movie, sports team, TV show, and other)?
- Which NER model provides the best balance of precision, recall, and F1-score across all categories?

Trends and Topic Analysis:

- What are the most common entities (e.g., persons, companies, products) mentioned across the platform, and how do they vary over time?
- How can the identified entities help in understanding emerging trends, public sentiment, and popular topics on the platform?

User Engagement and Content Analysis:

- How does identifying named entities in microblogs contribute to understanding user interests and preferences?
- Can the NER system provide insights into user demographics and behavior based on the types of entities they frequently mention?

Content Moderation and Compliance:

- How can the NER system assist in identifying and flagging potentially sensitive or inappropriate content based on the recognized entities?
- What role can entity recognition play in ensuring compliance with regulations and content policies, especially concerning hate speech, misinformation, and privacy?

Advertising and Monetization Opportunities:

- How can the information extracted from NER be used to enhance targeted advertising and personalized marketing strategies?
- Are there specific categories (e.g., companies, products) that provide lucrative opportunities for partnerships and sponsorships based on the frequency and context of their mentions?

Enhancing User Experience:

- How can the identified entities be used to improve the search and discovery features on the platform, making it easier for users to find relevant content?
- Can the NER system enhance content recommendations and suggest relevant microblogs to users based on their interests and the entities they engage with?

Scalability and System Integration:

- How can the NER system handle the scale of processing 500 million microblogs per day while maintaining high accuracy and efficiency?
- What are the technical and infrastructure requirements for integrating the NER system into the platform’s existing architecture?

Language and Cultural Diversity:

- Although the dataset is in English, how can the system be adapted to recognize named entities in multiple languages, considering the platform's global user base?
- What challenges might arise in accurately identifying and categorizing entities from diverse linguistic and cultural backgrounds?

Data Quality and Annotation:

- What measures can be taken to ensure the quality and accuracy of the annotated dataset used for training the NER models?
- How can the system handle ambiguities and variations in entity names, such as nicknames, abbreviations, and common misspellings?

Future Development and Innovation:

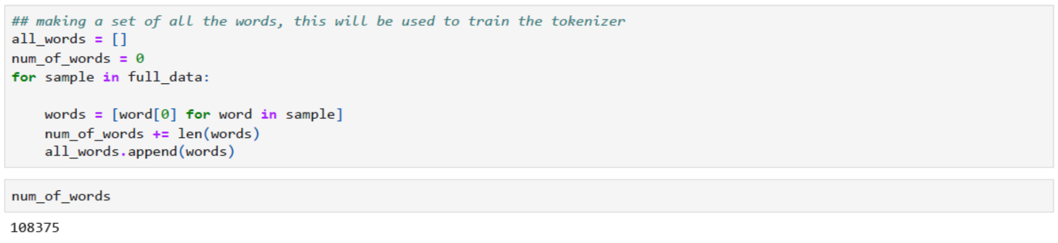
- What are the potential advancements in NER and related technologies that could further improve the system's capabilities and accuracy?
- How can the company leverage advancements in machine learning and NLP to explore new features and services that enhance user engagement and platform value?

Competitive Advantage:

- How does the implementation of an advanced NER system differentiate the platform from competitors in terms of content analysis and user engagement?
- What unique insights and analytics can the platform offer to users, advertisers, and partners as a result of the NER capabilities?

Analysis

- NER using CRF Bi-LSTM Model

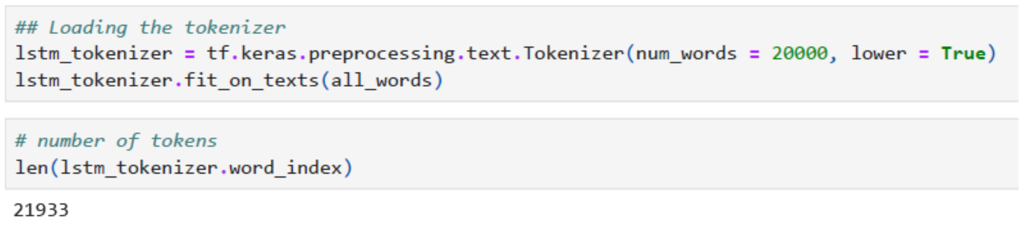


```
## making a set of all the words, this will be used to train the tokenizer
all_words = []
num_of_words = 0
for sample in full_data:

    words = [word[0] for word in sample]
    num_of_words += len(words)
    all_words.append(words)
```

```
num_of_words
```

```
108375
```

Figure 5.1: Basic Data Preprocessing

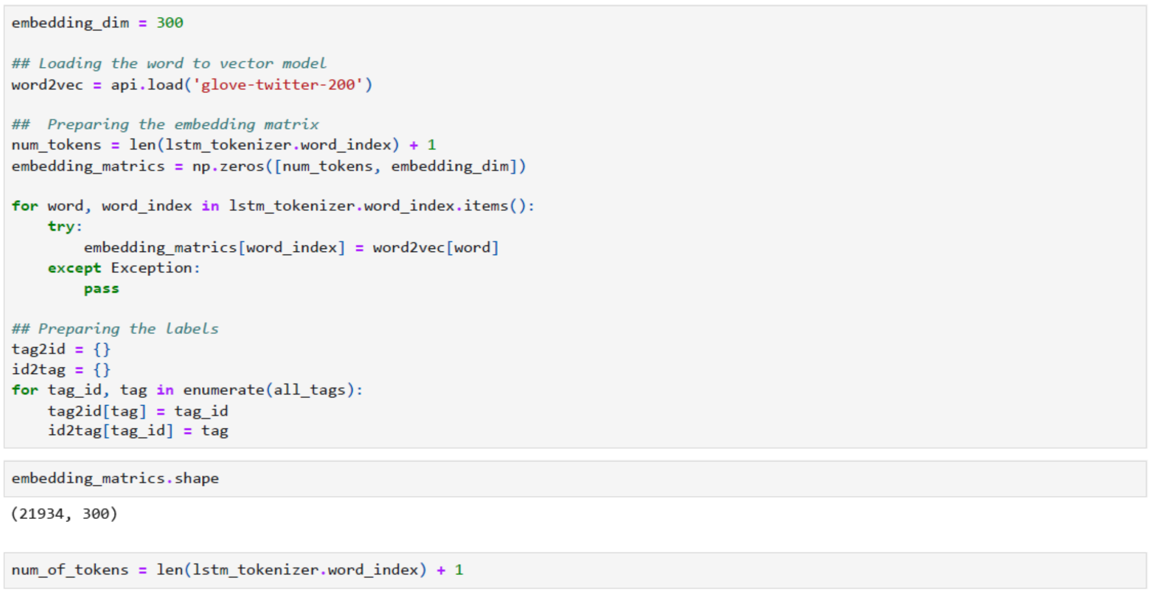


```
## Loading the tokenizer
lstm_tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words = 20000, lower = True)
lstm_tokenizer.fit_on_texts(all_words)
```

```
# number of tokens
len(lstm_tokenizer.word_index)
```

```
21933
```

Figure 5.2: Implementing Tokenizer on the Training Data



```
embedding_dim = 300

## Loading the word to vector model
word2vec = api.load('glove-twitter-200')

##  Preparing the embedding matrix
num_tokens = len(lstm_tokenizer.word_index) + 1
embedding_matrics = np.zeros([num_tokens, embedding_dim])

for word, word_index in lstm_tokenizer.word_index.items():
    try:
        embedding_matrics[word_index] = word2vec[word]
    except Exception:
        pass

## Preparing the labels
tag2id = {}
id2tag = {}
for tag_id, tag in enumerate(all_tags):
    tag2id[tag] = tag_id
    id2tag[tag_id] = tag
```

```
embedding_matrics.shape
```

```
(21934, 300)
```

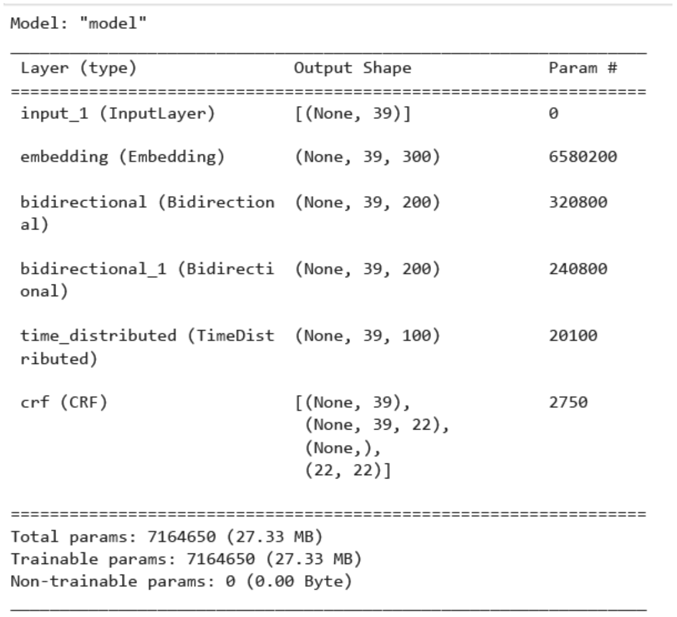
```
num_of_tokens = len(lstm_tokenizer.word_index) + 1
```

Figure 5.3: Using **word2vec** Model to Initialize Embeddings

- NER Model Training using LSTM

Hyperparameters Set 1:

LSTM Units = 100, weight_decay = 0.01, lr_rate = 0.001



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 39)]              0

 embedding (Embedding)       (None, 39, 300)           6580200

 bidirectional (Bidirection  (None, 39, 200)           320800
 al)

 bidirectional_1 (Bidirecti  (None, 39, 200)           240800
 onal)

 time_distributed (TimeDist  (None, 39, 100)           20100
 ributed)

 crf (CRF)                   [(None, 39),              2750
                              (None, 39, 22),
                              (None,),
                              (22, 22)]

=================================================================
Total params: 7164650 (27.33 MB)
Trainable params: 7164650 (27.33 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5.4: LSTM Model 1 Summary

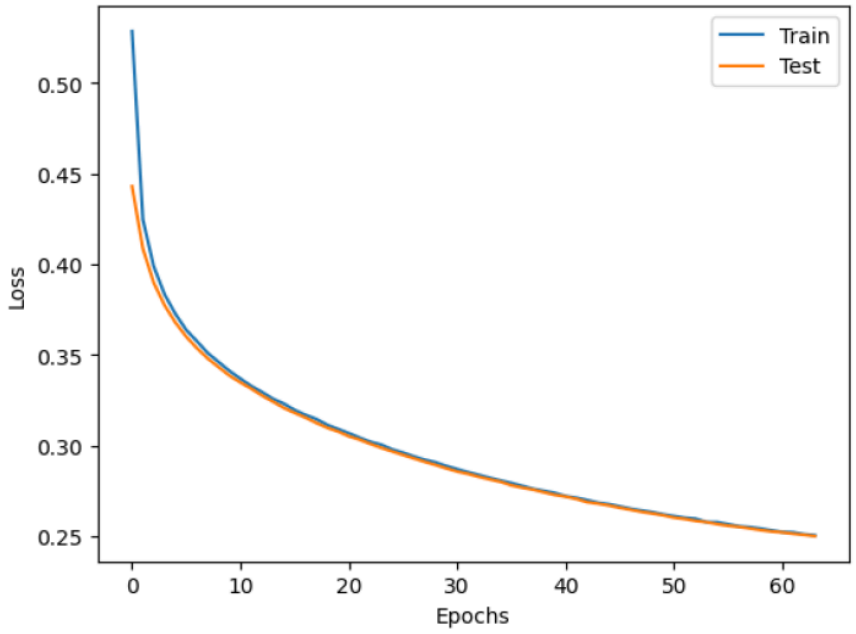
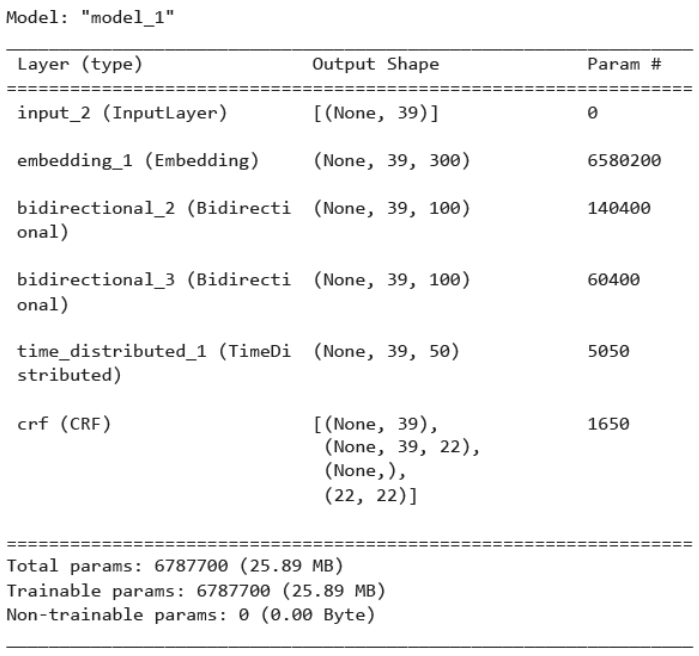


Figure 5.5: Loss Vs. Epoch Plot for Training and Test Data

Hyperparameters Set 2:

LSTM Units = 50, weight_decay = 0.01, lr_rate = 0.001



```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 39)]              0

 embedding_1 (Embedding)     (None, 39, 300)           6580200

 bidirectional_2 (Bidirecti  (None, 39, 100)           140400
 onal)

 bidirectional_3 (Bidirecti  (None, 39, 100)           60400
 onal)

 time_distributed_1 (TimeDi  (None, 39, 50)            5050
 stributed)

 crf (CRF)                   [(None, 39),              1650
                              (None, 39, 22),
                              (None,),
                              (22, 22)]

=================================================================
Total params: 6787700 (25.89 MB)
Trainable params: 6787700 (25.89 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 5.6: LSTM Model 2 Summary

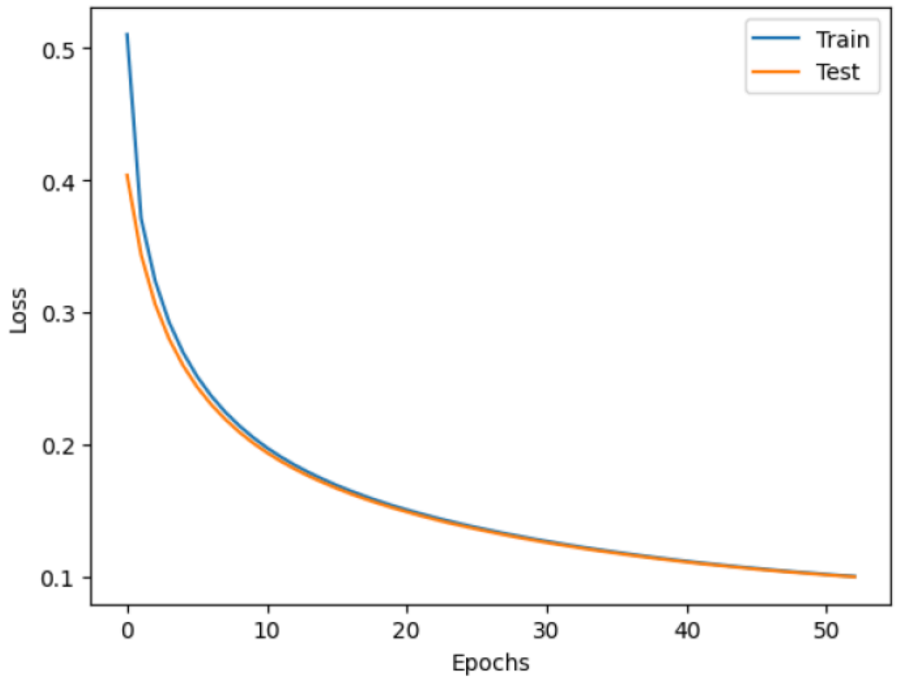
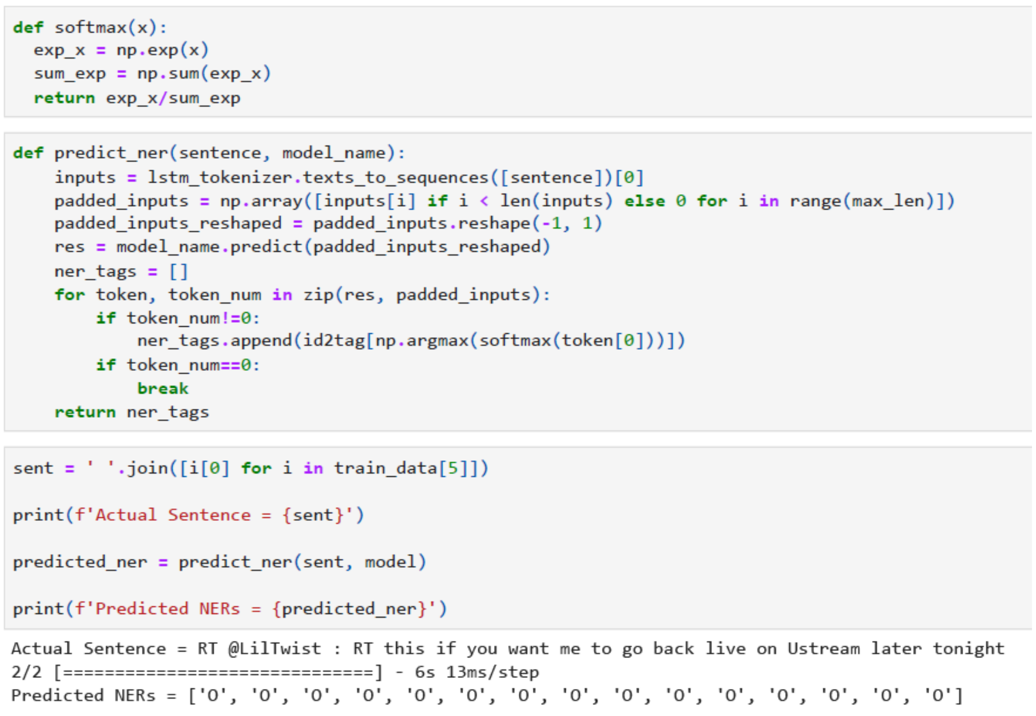


Figure 5.7: Loss Vs. Epoch Plot for Training and Test Data

Evaluation and Prediction from the Best Model:



```
def softmax(x):
  exp_x = np.exp(x)
  sum_exp = np.sum(exp_x)
  return exp_x/sum_exp
```

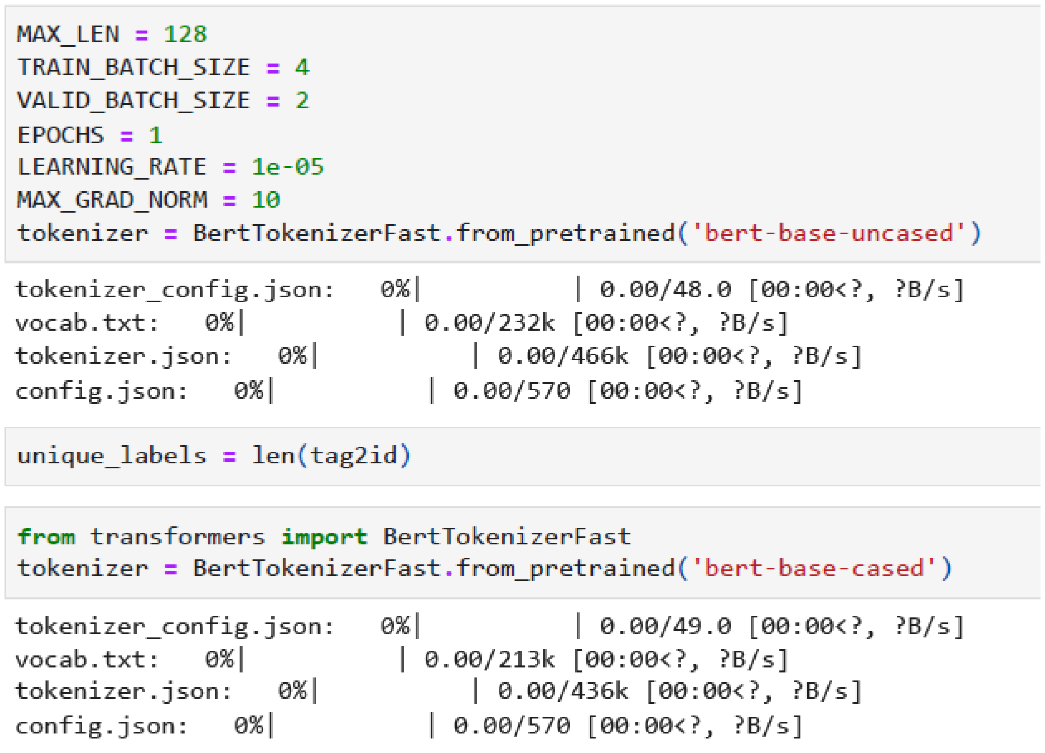
```
def predict_ner(sentence, model_name):
    inputs = lstm_tokenizer.texts_to_sequences([sentence])[0]
    padded_inputs = np.array([inputs[i] if i < len(inputs) else 0 for i in range(max_len)])
    padded_inputs_reshaped = padded_inputs.reshape(-1, 1)
    res = model_name.predict(padded_inputs_reshaped)
    ner_tags = []
    for token, token_num in zip(res, padded_inputs):
        if token_num!=0:
            ner_tags.append(id2tag[np.argmax(softmax(token[0]))])
        if token_num==0:
            break
    return ner_tags
```

```
sent = ' '.join([i[0] for i in train_data[5]])

print(f'Actual Sentence = {sent}')

predicted_ner = predict_ner(sent, model)

print(f'Predicted NERs = {predicted_ner}')
```

```
Actual Sentence = RT @LilTwist : RT this if you want me to go back live on Ustream later tonight
2/2 [==============================] - 6s 13ms/step
Predicted NERs = ['O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O']
```

Figure 5.8: Evaluation and Prediction from the Best LSTM Model

- NER Model Training using BERT



```
MAX_LEN = 128
TRAIN_BATCH_SIZE = 4
VALID_BATCH_SIZE = 2
EPOCHS = 1
LEARNING_RATE = 1e-05
MAX_GRAD_NORM = 10
tokenizer = BertTokenizerFast.from_pretrained('bert-base-uncased')
```

```
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

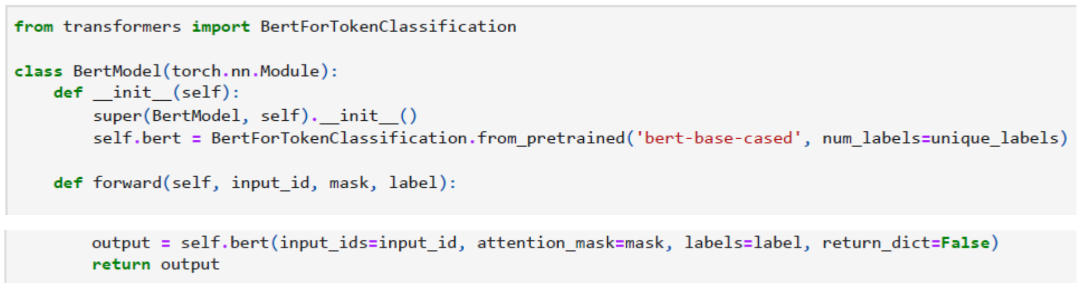
```
unique_labels = len(tag2id)
```

```
from transformers import BertTokenizerFast
tokenizer = BertTokenizerFast.from_pretrained('bert-base-cased')
```

```
tokenizer_config.json:   0%|          | 0.00/49.0 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]
config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]
```

Figure 5.9: Downloading the Tokenizer for BERT

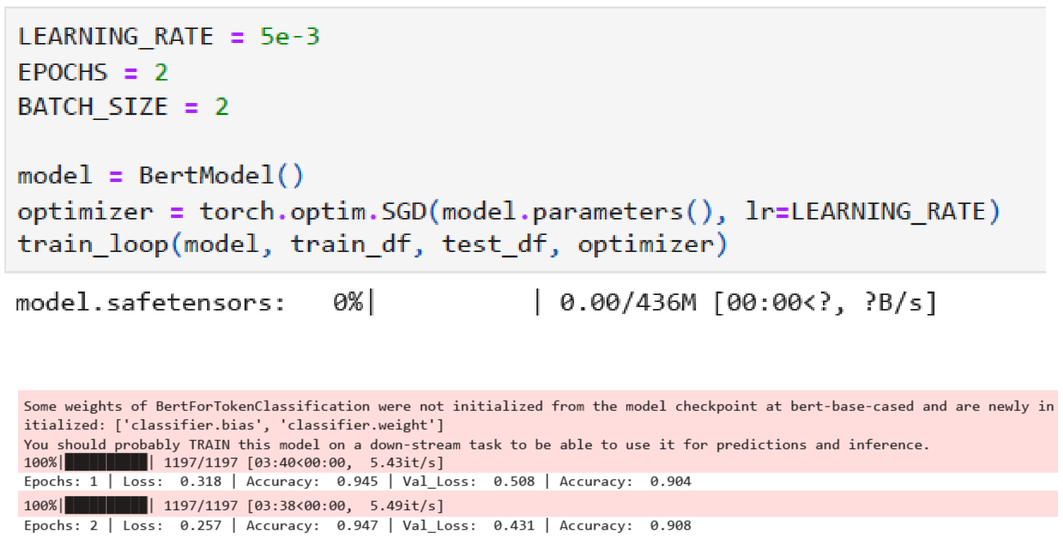
Model:



```
from transformers import BertForTokenClassification

class BertModel(torch.nn.Module):
    def __init__(self):
        super(BertModel, self).__init__()
        self.bert = BertForTokenClassification.from_pretrained('bert-base-cased', num_labels=unique_labels)

    def forward(self, input_id, mask, label):

        output = self.bert(input_ids=input_id, attention_mask=mask, labels=label, return_dict=False)
        return output
```

Figure 5.10: BERT Model Initialization

Hyperparameters Set 1:



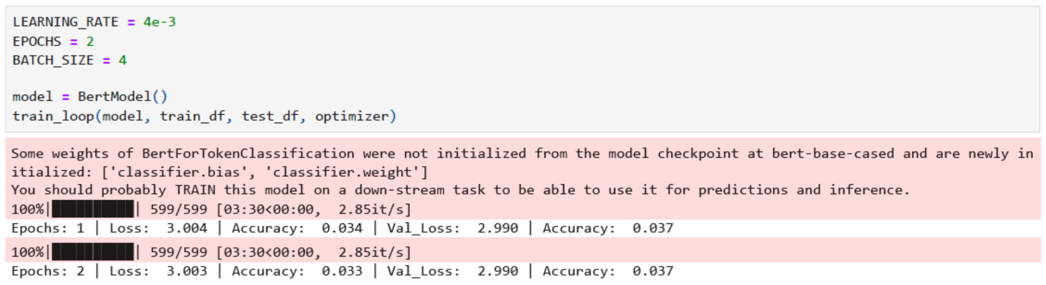
```
LEARNING_RATE = 5e-3
EPOCHS = 2
BATCH_SIZE = 2

model = BertModel()
optimizer = torch.optim.SGD(model.parameters(), lr=LEARNING_RATE)
train_loop(model, train_df, test_df, optimizer)
```

```
model.safetensors:   0%|          | 0.00/436M [00:00<?, ?B/s]
```

```
Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly in
itialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
100%|██████████| 1197/1197 [03:40<00:00,  5.43it/s]
Epochs: 1 | Loss:  0.318 | Accuracy:  0.945 | Val_Loss:  0.508 | Accuracy:  0.904
100%|██████████| 1197/1197 [03:38<00:00,  5.49it/s]
Epochs: 2 | Loss:  0.257 | Accuracy:  0.947 | Val_Loss:  0.431 | Accuracy:  0.908
```

Figure 5.11: BERT Model with Hyperparameter Set 1

Hyperparameters Set 2:

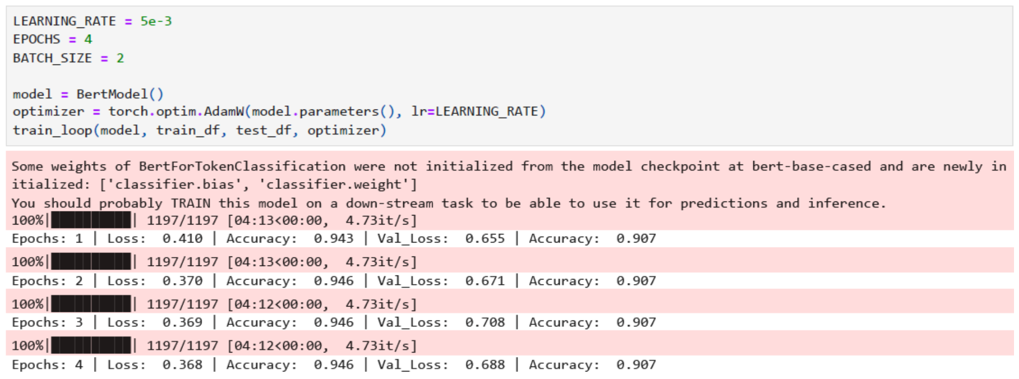


```
LEARNING_RATE = 4e-3
EPOCHS = 2
BATCH_SIZE = 4

model = BertModel()
train_loop(model, train_df, test_df, optimizer)
```

```
Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly in
itialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
100%|██████████| 599/599 [03:30<00:00,  2.85it/s]
Epochs: 1 | Loss:  3.004 | Accuracy:  0.034 | Val_Loss:  2.990 | Accuracy:  0.037
100%|██████████| 599/599 [03:30<00:00,  2.85it/s]
Epochs: 2 | Loss:  3.003 | Accuracy:  0.033 | Val_Loss:  2.990 | Accuracy:  0.037
```

Figure 5.12: BERT Model with Hyperparameter Set 2

Hyperparameters Set 3:

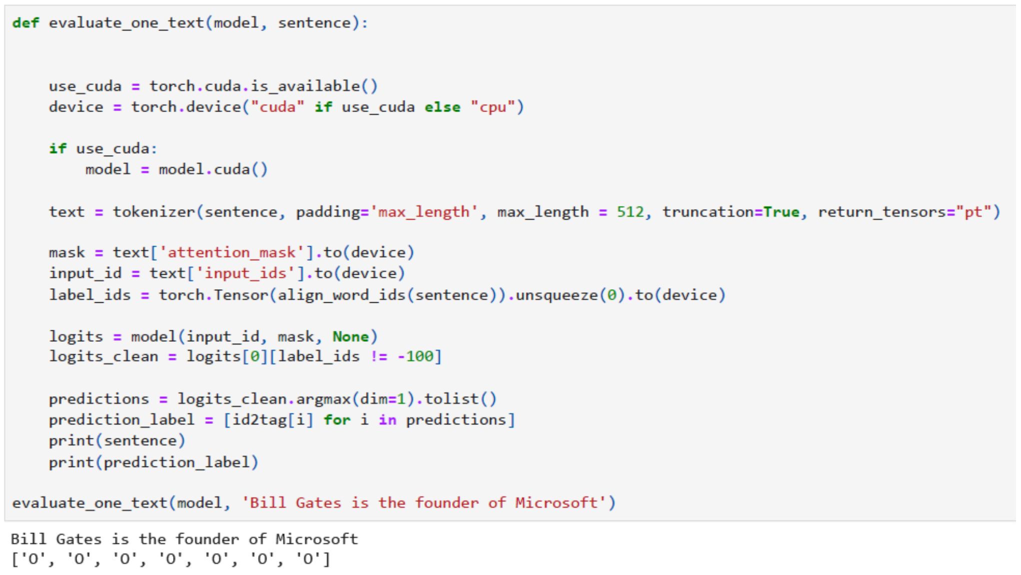


```
LEARNING_RATE = 5e-3
EPOCHS = 4
BATCH_SIZE = 2

model = BertModel()
optimizer = torch.optim.AdamW(model.parameters(), lr=LEARNING_RATE)
train_loop(model, train_df, test_df, optimizer)
```

```
Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly in
itialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
100%|██████████| 1197/1197 [04:13<00:00,  4.73it/s]
Epochs: 1 | Loss:  0.410 | Accuracy:  0.943 | Val_Loss:  0.655 | Accuracy:  0.907
100%|██████████| 1197/1197 [04:13<00:00,  4.73it/s]
Epochs: 2 | Loss:  0.370 | Accuracy:  0.946 | Val_Loss:  0.671 | Accuracy:  0.907
100%|██████████| 1197/1197 [04:12<00:00,  4.73it/s]
Epochs: 3 | Loss:  0.369 | Accuracy:  0.946 | Val_Loss:  0.708 | Accuracy:  0.907
100%|██████████| 1197/1197 [04:12<00:00,  4.73it/s]
Epochs: 4 | Loss:  0.368 | Accuracy:  0.946 | Val_Loss:  0.688 | Accuracy:  0.907
```

Figure 5.13: BERT Model with Hyperparameter Set 3

Predicting the Token for Next Sentence:



```
def evaluate_one_text(model, sentence):


    use_cuda = torch.cuda.is_available()
    device = torch.device("cuda" if use_cuda else "cpu")

    if use_cuda:
        model = model.cuda()

    text = tokenizer(sentence, padding='max_length', max_length = 512, truncation=True, return_tensors="pt")

    mask = text['attention_mask'].to(device)
    input_id = text['input_ids'].to(device)
    label_ids = torch.Tensor(align_word_ids(sentence)).unsqueeze(0).to(device)

    logits = model(input_id, mask, None)
    logits_clean = logits[0][label_ids != -100]

    predictions = logits_clean.argmax(dim=1).tolist()
    prediction_label = [id2tag[i] for i in predictions]
    print(sentence)
    print(prediction_label)

evaluate_one_text(model, 'Bill Gates is the founder of Microsoft')
```

```
Bill Gates is the founder of Microsoft
['O', 'O', 'O', 'O', 'O', 'O', 'O']
```

Figure 5.14: Predicting the Token for Next Sentence

Insights and Recommendations

- **Business Insights**

1. **Understanding Trends Beyond Hashtags:**
    - The microblogging site's initiative to use Named Entity Recognition (NER) to automatically tag and analyze tweets reflects a move towards a deeper understanding of user-generated content. By bypassing the limitations of user-defined hashtags, the social media platform can gain insights into the underlying entities and topics, which may not always be explicitly tagged by users.

2. **Data Annotation and Tokenization:**
    - The use of the CONLL bio format for data annotation highlights the structured approach in representing entities within sentences, facilitating clear differentiation between words and their corresponding tags. Tokenization is crucial in this context as it converts unstructured text into a machine-readable format, allowing the NER models to accurately identify and classify entities.

3. **Diverse NER Data Annotation Formats:**
   - The availability of various data annotation formats like IO, JSON, and XML indicates the flexibility needed in handling different data structures. These formats serve different purposes, such as ease of data manipulation and integration with other systems, which can be crucial for various applications, including biomedical data analysis and beyond.
4. **Model Selection and Performance:**
   - The selection of models such as Conditional Random Fields (CRF), Bidirectional LSTM-CRF, BERT, Spacy NER, Flair, and ULMFiT shows a comprehensive approach to finding the best fit for NER tasks. BERT, with its focus on capturing long-range dependencies through self-attention, is particularly well-suited for complex language tasks. Early stopping is effectively used to prevent overfitting, ensuring that models generalize well to new data.
5. **Advantages of Attention-Based Models:**
   - The preference for attention-based models like BERT over recurrent-based models underscores the importance of capturing relationships between distant words in a sentence. This capability is critical for understanding context in tweets, which often contain non-linear structures and references.

- **Recommendations**

1. **Leverage NER for Enhanced Content Understanding:**
   - Implement NER across the platform to automatically tag entities in tweets. This can help in categorizing content, understanding emerging trends, and identifying key topics and entities that drive user engagement. This system should be designed to continuously update and adapt to new terms and entities as they emerge.
2. **Adopt Multiple Annotation Formats for Versatility:**
   - Consider using a combination of data annotation formats, depending on the specific use case. For instance, JSON format can be used for integration with web-based APIs, while CONLL bio format can be maintained for internal training datasets. This multi-format approach can facilitate interoperability with different data systems and applications.

3. **Implement Advanced Tokenization and Preprocessing:**
    - Enhance tokenization processes to handle the nuances of social media text, such as abbreviations, emojis, and slang. Preprocessing steps should include normalization and handling of special characters to improve the accuracy of NER models.
4. **Optimize Model Selection and Training:**
    - While BERT offers robust performance for NER, it continues experimenting with other models like Flair and ULMFiT to explore potential gains in specific contexts. Utilize transfer learning techniques where applicable, and refine models through hyperparameter tuning and early stopping to achieve optimal performance.
5. **Explore Contextual Analysis for Richer Insights:**
    - Beyond identifying named entities, develop mechanisms to analyze the context in which entities are mentioned. This can provide deeper insights into user sentiments and opinions. For example, distinguishing between positive and negative mentions of a company can inform brand sentiment analysis.
6. **Expand Beyond English Language:**
    - As the platform has a global user base, consider extending NER capabilities to support multiple languages. This will involve training models on multilingual datasets and ensuring that the system can accurately tag entities in diverse linguistic contexts.
7. **Utilize NER for Targeted Advertising and Content Recommendations:**
    - Use the data from NER to enhance targeted advertising and content recommendation systems. By understanding the entities and topics users are interested in, this microblogging site can deliver more relevant ads and suggested content, improving user experience and engagement.
8. **Monitor and Update Entity Recognition Systems:**
    - Regularly update the NER system to account for new entities, slang, and emerging trends. Establish a feedback loop where user interactions and feedback are used to refine the models and improve accuracy over time.

9. **Ethical Considerations and Transparency:**
    - Ensure transparency in how entity recognition data is used, especially in content moderation and user profiling. Establish clear policies and guidelines to address ethical considerations, such as avoiding biases in entity tagging and ensuring user privacy.

By implementing these recommendations, the microblogging and social media platform can enhance its ability to understand and analyze the vast amounts of content generated on the platform. This will not only improve content discovery and user experience but also provide valuable insights into trends and public sentiment.

CONCLUSION

Key Takeaways:

1. **Digital Advertising Optimization:** Accurate forecasting of page views allows for strategic ad placement, significantly improving ad performance and cost efficiency.
2. **Logistics Efficiency:** Precise delivery time estimations enhance customer satisfaction by providing reliable service timelines, crucial for retaining and attracting customers in the competitive logistics market.
3. **Automated Classification in Supply Chains:** Automated image classification in fresh produce supply chains ensures quality control and operational efficiency, reducing manual effort and potential errors.
4. **Enhancing Financial Literacy:** Categorizing news articles helps users easily access relevant financial information, thereby enhancing their understanding and engagement with financial markets.
5. **Advanced Content Analysis:** The application of NER in microblogging platforms provides deep insights into trends and topics, improving content discovery and user engagement.

Practical Applications:

- The methodologies applied in these projects have real-world implications across various industries. From improving advertising strategies and logistics operations to automating quality control and enhancing content delivery, the use of AI and ML tools can drive significant business growth and customer satisfaction.

Limitations and Suggestions for Improvement:

- While these studies demonstrate the potential of AI and machine learning, there are limitations such as data quality, model interpretability, and scalability. The accuracy of predictive models can be affected by the quality and quantity of data available.

Additionally, complex models like deep learning can be difficult to interpret, making it challenging to understand the decision-making process. Future work should focus on improving data collection methods, developing more interpretable models, and ensuring scalability of solutions to handle large datasets efficiently.

In conclusion, these case studies underscore the importance of leveraging advanced analytics and AI technologies to solve business problems, enhance operational efficiency, and deliver superior customer experiences. As these technologies continue to evolve, they offer exciting opportunities for further innovation and optimization across various business domains.

References

- **Websites:**

1. Time Series Forecasting — A Complete Guide | by Puja P. Pathak | Analytics Vidhya | Medium
2. What Is Time Series Forecasting? - MachineLearningMastery.com
3. Prophet | Forecasting at scale. (facebook.github.io)
4. Neural Networks: Forward pass and Backpropagation | by Ritwick Roy | Towards Data Science
5. Building a Neural Network with Tensorflow | by Muktha Sai Ajay | Towards Data Science
6. CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more… | by Siddharth Das | Analytics Vidhya | Medium
7. GitHub - Adithia88/Image-Classification-using-VGG19-and-Resnet
8. Data Preprocessing Steps for NLP. The Complete NLP Guide: Text to Context… | by Merve Bayram Durna | Medium
9. What is named entity recognition (NER) and how can I use it? | by Christopher Marshall | super.AI | Medium
10. LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras | by Ryan T. J. J. | Analytics Vidhya | Medium
11. [1706.03762] Attention Is All You Need (arxiv.org)
12. BERT Explained: What it is and how does it work? | Towards Data Science

- **Research Publications:**

1. Forecasting Time Series Data with Prophet: Build, improve, and optimize time series forecasting models using Meta's advanced forecasting tool | Packt Publishing books | IEEE Xplore
2. Fundamentals of Artificial Neural Networks and Deep Learning | SpringerLink
3. A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification | IEEE Conference Publication | IEEE Xplore
4. ResNet-50 vs VGG-19 vs training from scratch: A comparative analysis of the segmentation and classification of Pneumonia from chest X-ray images - ScienceDirect
5. researchgate.net/publication/13853244_Long_Short-term_Memory
6. [1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (arxiv.org)