

A SHORT-TERM INTERNSHIP REPORT ON
ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

BY

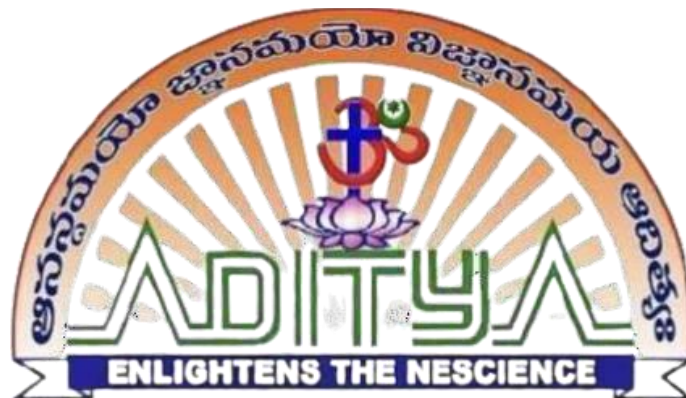
NARAMANI SRAVANI

III BCA

Under the Esteemed Guidance of

Mr. G.V.S.S PRASANTH SIR

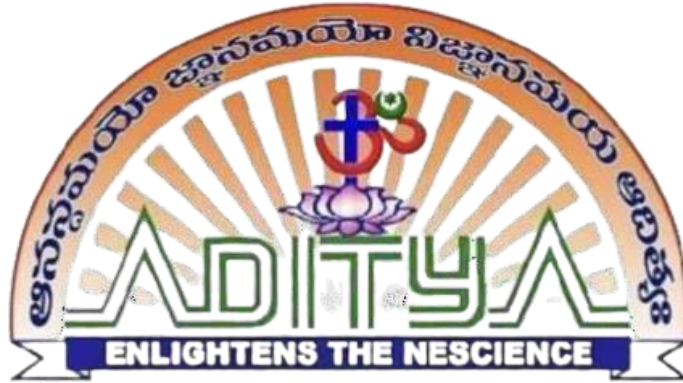
Tutor of Artificial Intelligence & Machine Learning



SRI ADITYA DEGREE COLLEGE,
Bhimavaram

(Affiliated to ADIKAVI NANNAYA UNIVERSITY)
BHIMAVARAM-534201, WEST GODAVARI DISTRICT,
ANDHRA PRADESH, 2022-2025

SRI ADITYA DEGREE COLLEGE



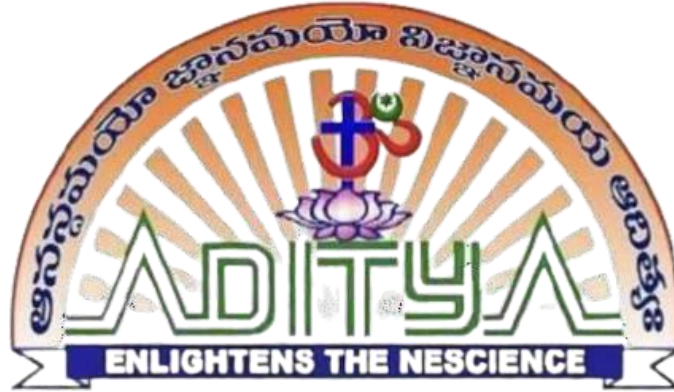
DECLARATION BY THE STUDENT

I hereby declare that the work described in this Short-term Internship, entitled “**Artificial Intelligence & Machine Learning**” which is being submitted by me in partial fulfilment of the requirements for the award of degree of **Bachelor of Computer Applications** from the Department of computers Science to Sri Aditya Degree College, Bhimavaram under the guidance of **Mr. G.V.S.S. PRASANTH Sir** tutor of **Artificial Intelligence & Machine Learning** in Sri Aditya Degree College, Bhimavaram.

Place: Bhimavaram

(Naramani Sravani) Date:

SRI ADITYA DEGREE COLLEGE



CERTIFICATE FROM THE SUPERVISOR

This is to certify that the Short-Term Internship entitled,”
ARTIFICIAL INTELLIGENCE & MACHINE LEARNING”, that is being
submitted by **Naramani Sravani** bearing **224107400038** of **III BCA**, which
is being submitted by us in partial fulfilment of the requirements for the
award of degree of **Bachelor of Computer Applications** from the
Department of **Computer Science** to Sri Aditya Degree College, bonified
work carried out by him under my guidance and Supervision.

(Mr. G.V.S.S PRASANTH SIR)

ACKNOWLEDGEMENT

No endeavour is completed without the valuable support of others.

I would like to take this opportunity to extend my sincere gratitude to all those who have contributed to the successful completion of this Short - Term

Internship Project Report. I express my deep sense of gratitude to

Mrs. A. Anuradha mam, Principal, for her Efforts and for giving us permission for carrying out this Short-Term Internship. I feel deeply honoured in expressing my sincere thanks to Mr. G.V.S.S Prashanth Sir tutor of ULearn Visakhapatnam for making the resources available at right time and providing valuable insights leading to the successful

completion of my short-Term Internship Project Report.

Finally, I thank all the faculty members of our department who contributed their valuable suggestions in completion of Short-Term Internship report and I also put my sincere thanks to My Parents who stood with me during the whole Short-Term Internship.

(Naramani Sravani)

CONTENTS

- **Introduction to AI AND ML using python**
- **ML and types of ML**
- **Applications of ML**
- **Deep Learning**
- **ANN, NLP, CC**
- **AI tools, we use in our daily life**
- **Back Propagation**
- **Difference between Neural networks and Deep Neural networks**
- **Difference between Chatgpt and Google**
- **POS Tagging**
- **Object Detection**
- **CNN Algorithm**
- **Deep fake, deep dream**
- **GAN model and Architecture**
- **Data Augmentation**
- **Parameter sharing and typing**
- **Ensemble methods**
- **Bayes theorem**
- **LSTM – Long short term Memory**
- **Restricted Boltzmann machine – RBM**
- **RNN – Recurrent Neural Network**
- **Auto encoders and types**
- **VGG Net and Architecture**
- **Google Net and its Architecture**
- **Programming languages**
- **Python and it's uses**
- **Python and pycharm, anaconda installations**
- **Datatypes**
- **Arithmetic operations**
- **Declaration of comments and variables**
- **Reserved words**
- **Control statements**
- **Prime numbers, factors, reverse number, duplicates elimination**

INTRODUCTION

A supermarket is a self-service shop offering a wide variety of food, beverages and household products, organized into sections. This kind of store is larger and has a wider selection than earlier grocery stores, but is smaller and more limited in the range of merchandise than a hypermarket or big-box market. In everyday United States usage, however, "grocery store" is often used to mean "supermarket". The supermarket typically has places for fresh meat, fresh produce, dairy, deli items, baked goods, and similar foodstuffs

Shelf space is also reserved for canned and packaged goods and for various non-food items such as kitchenware, household cleaners, pharmacy products and pet supplies. Some supermarkets also sell other household products that are consumed regularly, such as alcohol (where permitted), medicine, and clothing, and some sell a much wider range of non-food products: DVDs, sporting equipment, board games, and seasonal items (e.g., Christmas wrapping paper, Easter eggs, school uniforms, Valentine's Day themed gifts, Mother's Day gifts, Father's Day gifts and Halloween).

A larger full-service supermarket combined with a department store is sometimes known as a hypermarket. Other services may include those of banks, cafés, childcare centers/creches, insurance (and other financial services), mobile phone sales, photo processing, video rentals, pharmacies, and gas stations.

If the eatery in a supermarket is substantial enough, the facility may be called a "grocerant", a portmanteau of "grocery" and "restaurant". The traditional supermarket occupies a large amount of floor space, usually on a single level. It is usually situated near a residential area in order to be convenient to consumers. The basic appeal is the availability of a broad selection of goods under a single roof, at relatively low prices.

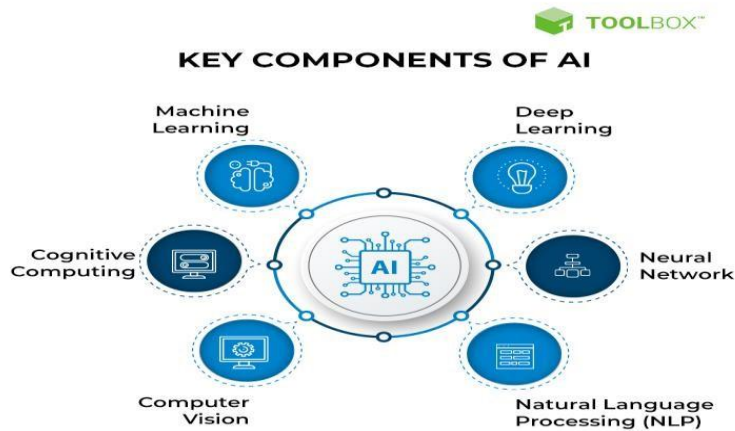
Other advantages include ease of parking and frequently the convenience of shopping hours that extend into the evening or even 24 hours of the day. Supermarkets usually allocate large budgets to advertising, typically through newspapers and television. They also present elaborate in-shop displays of products.

Supermarkets typically are chain stores, supplied by the distribution centers of their parent companies, thus increasing opportunities for economies of scale. Supermarkets usually offer products at relatively low prices by using their buying power to buy goods from manufacturers at lower prices than smaller stores can. They also minimize financing costs by paying for goods at least 30 days after receipt and some extract credit terms of 90 days or more from vendors. Certain products (typically staple foods such as bread, milk and sugar) are very occasionally sold as loss leaders so as to attract shoppers to their store. Supermarkets make up for their low margins by a high volume of sales, and with of higher-margin items bought by the customers. Self-service with shopping carts (trolleys) or baskets reduces labor costs, and many supermarket chains are attempting further reduction by shifting to self-service check-outs.

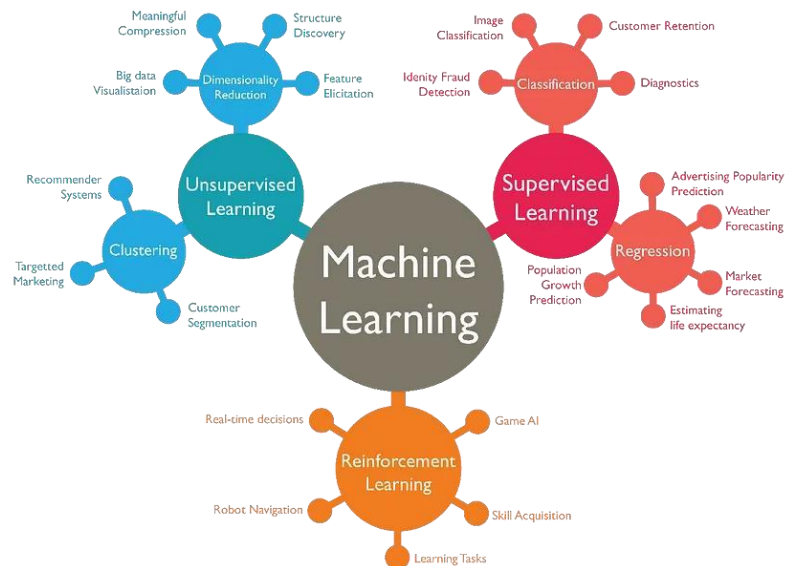
LEARNING OUTCOME OF SHORT-TERM INTERNSHIP

Introduction to Artificial Intelligence and Machine Learning

Artificial Intelligence (AI) is a branch of computer science focused on creating systems capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, and language understanding. AI aims to develop machines that can simulate human intelligence and execute complex tasks autonomously.



Machine Learning (ML) is a subset of AI that provides systems with the ability to learn and improve from experience without being explicitly programmed. ML algorithms enable computers to analyze large datasets, detect patterns, and make data-driven decisions. This capability is pivotal in various applications, from predictive analytics to autonomous systems.



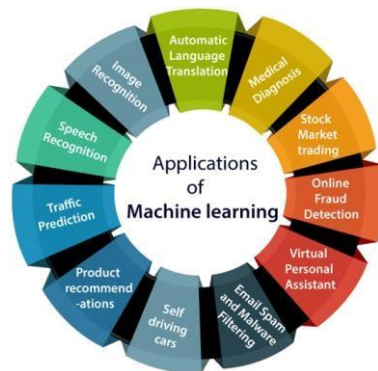
Types of Machine Learning

- **Supervised Learning:** In supervised learning, the algorithm learns from labeled training data to predict outcomes for new data. It involves training the model on input-output pairs and is used for tasks like classification and regression.
- **Unsupervised Learning:** Unsupervised learning deals with unlabeled data where the algorithm discovers patterns and structures on its own. Clustering and association are common tasks in unsupervised learning.
- **Reinforcement Learning:** Reinforcement learning involves an agent learning to make decisions in an environment to maximize cumulative rewards. The agent learns through trial and error, receiving feedback from the environment.
- **Semi-supervised Learning:** This type combines both labeled and unlabeled data for training. It leverages small amounts of labeled data with a large amount of unlabeled data to improve learning accuracy.

Applications of Machine Learning

Machine Learning finds applications across various domains due to its ability to analyze data and make predictions. Some prominent applications include Predictive Analytics: ML models predict outcomes based on historical data, used in finance for stock market prediction, and in healthcare for disease diagnosis.

Natural Language Processing (NLP): NLP enables machines to understand, interpret, and generate human language. Applications include sentiment analysis, machine translation, and chatbots.



Computer Vision (CV): CV involves teaching machines to interpret and understand visual information from the world. It's used in facial recognition, object detection, autonomous vehicles, and medical image analysis.

Recommendation Systems: ML powers recommendation engines that suggest products, movies, or content based on user preferences and behaviors.

Fraud Detection: ML algorithms detect fraudulent activities in banking transactions and online transactions by analyzing patterns and anomalies.

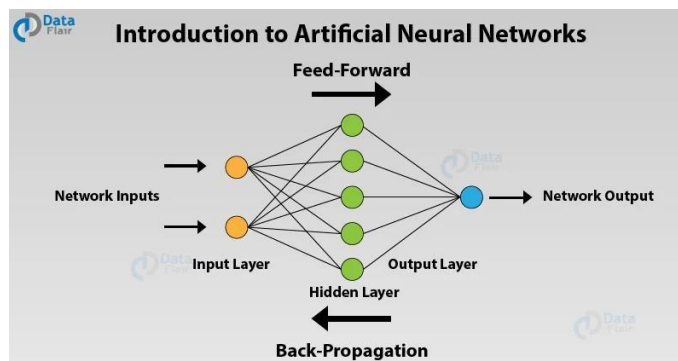
Deep Learning :

Deep Learning is a subset of ML inspired by the structure and function of the human brain. It uses deep neural networks with multiple layers (deep networks) to learn hierarchical representations of data.

Artificial Neural Networks (ANN)

ANNs are computational models inspired by biological neural networks. They consist of interconnected nodes (neurons) organized in layers. ANNs are capable of learning complex patterns and relationships in data.

Natural Language Processing (NLP): NLP uses deep learning models like recurrent neural networks (RNNs) and transformers to process and understand human language. Applications range from machine translation to sentiment analysis and chatbots.



Computer Vision (CV): Deep learning models such as Convolutional Neural Networks (CNNs) are used in CV tasks like image classification, object detection, and image segmentation. CNNs learn to extract features from images and make predictions.

Generative Adversarial Networks (GANs): GANs are a type of deep learning framework where two neural networks compete against each other. GANs are used for generating realistic images, videos, and audio.

Modern AI Tools

Artificial Intelligence (AI) has become increasingly integrated into our daily lives, impacting various aspects from communication and entertainment to productivity and healthcare. Here's an exploration of AI tools that are commonly used and have become indispensable in modern society:

1. Virtual Assistants

Examples: Siri (Apple), Google Assistant, Amazon Alexa, Microsoft Cortana



Description: Virtual assistants use AI to understand voice commands and perform tasks such as setting reminders, answering questions, providing weather updates, and controlling smart home devices. They leverage Natural Language Processing (NLP) and Machine Learning to improve accuracy and responsiveness over time.

Impact: Virtual assistants enhance convenience by allowing hands-free interaction with technology, streamlining daily tasks, and providing personalized information and recommendations.

2. Recommendation Systems

Examples:

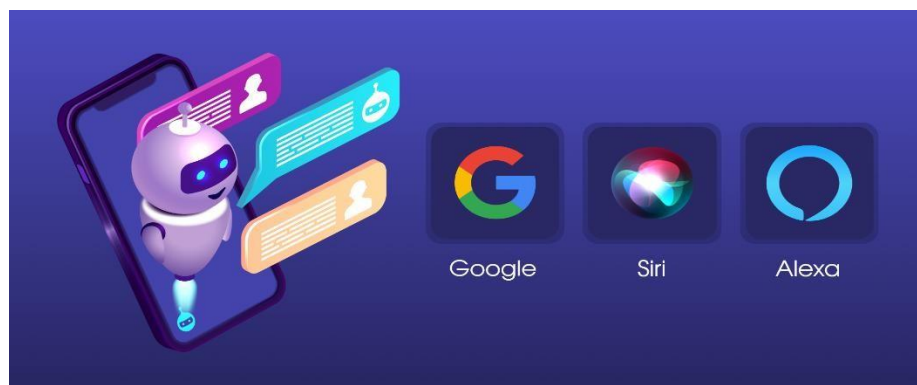
Netflix ,recommendation engine, Amazon product recommendations, Spotify Discover Weekly

Description: Recommendation systems use AI algorithms to analyze user preferences and behavior. They suggest content, products, or services that users are likely to find relevant based on past interactions and similarities with other users.

Impact: These systems enhance user experience by personalizing content consumption, increasing engagement, and driving sales through targeted recommendations.

3. Image and Speech Recognition

Examples: Face recognition on smartphones, Google Photos, speech-to-text applications



Description: AI-powered image recognition identifies objects, faces, and scenes in photos and videos. Speech recognition converts spoken language into text, enabling hands-free typing and voice commands.

Impact: These technologies improve accessibility, automate data entry, and enhance security (e.g., facial recognition for unlocking devices). They also facilitate applications in healthcare (medical imaging analysis) and surveillance.

4. Spam Filters and Email Classification

Examples: Gmail spam filter, Outlook categorization

Description: AI algorithms classify incoming emails based on content and sender behavior to filter out spam and categorize messages into primary, social, or promotional folders.

Impact: These tools improve productivity by reducing the time spent managing email and prioritizing important communications. They also enhance cybersecurity by detecting phishing attempts and malicious content.

5. Autonomous Vehicles

Examples: Tesla Autopilot, Waymo self-driving cars

Description: AI is used in autonomous vehicles to perceive surroundings through sensors (radar, lidar, cameras), interpret data in real-time, and make decisions such as steering, accelerating, and braking.

Impact: Autonomous vehicles promise to improve road safety, reduce traffic congestion, and provide mobility solutions for elderly and disabled individuals. They represent a significant advancement in transportation technology.

6. Predictive Analytics in Healthcare

Examples: Predicting disease outbreaks, personalized medicine

Description: AI analyzes large datasets of medical records, genomic data, and clinical trials to identify patterns and predict outcomes. This aids in early disease detection, treatment planning, and drug discovery.

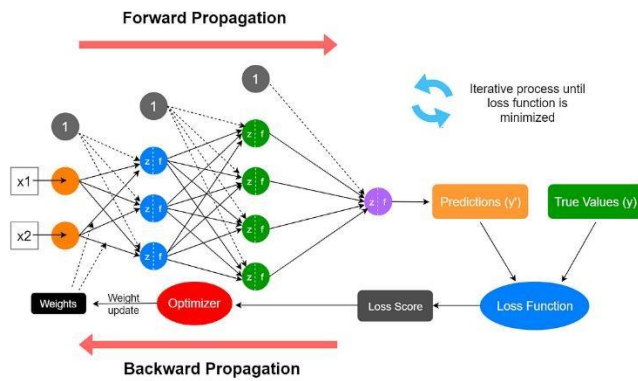
Impact: Predictive analytics improve patient outcomes by enabling proactive healthcare interventions, optimizing resource allocation in hospitals, and supporting medical research and development.

Understanding Back propagation

1. Neural Network Basics:

Neurons: Artificial neurons in a neural network receive inputs, apply weights, and pass the result through an activation function to produce an output.

Layers: Networks are organized in layers—input, hidden, and output—with connections (weights) between neurons.



2. Forward Propagation:

Process: During forward propagation, input data is fed into the network. Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the result to the next layer.

Output: The final layer produces predictions or outputs based on the input data.

3. Backpropagation Algorithm:

Purpose: Backpropagation calculates gradients of the loss function with respect to each weight in the network. It enables the network to learn from errors and adjust weights accordingly to improve prediction accuracy.

Steps:

Compute Loss: Measure the difference between predicted and actual outputs using a loss function (e.g., mean squared error).

Backward Pass: Propagate the error backward through the network to calculate gradients.

Gradient Descent: Update weights and biases iteratively to minimize the loss function using optimization algorithms like stochastic gradient descent (SGD).

4. Chain Rule of Calculus:

Mathematical Foundation: Backpropagation utilizes the chain rule to compute gradients efficiently across multiple layers of a neural network.

Error Attribution: Errors are attributed backward through the network, layer by layer, adjusting weights based on how much they contributed to the overall error.

5. Activation Functions:

Role: Activation functions introduce non-linearity to neural networks, enabling them to learn complex patterns in data.

Types: Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax (for multi-class classification).

6 Training Process:

Iterations: Backpropagation involves multiple iterations (epochs) where the network learns from training data, adjusts weights, and improves prediction accuracy over time.

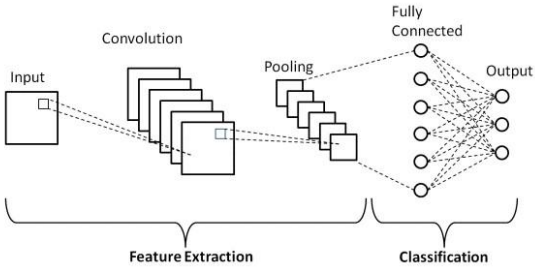
Overfitting: Techniques like regularization (e.g., L1, L2 regularization) help prevent overfitting by penalizing large weights.

Practical Applications of Backpropagation

1Image and Speech Recognition:

Aspect	Neural Network	Deep Neural Network
--------	----------------	---------------------

Convolutional Neural Networks (CNNs): Backpropagation is crucial for training CNNs to classify images and recognize objects.



Recurrent Neural Networks (RNNs): Used in speech recognition and natural language processing tasks where sequences of data are processed.

Financial Forecasting and Predictive Analytics:

Time Series Analysis: Backpropagation aids in forecasting stock prices, economic trends, and customer behavior based on historical data.

Anomaly Detection: Identifying unusual patterns in financial transactions or network traffic.

Healthcare and Medical Diagnosis:

Medical Imaging Analysis: Backpropagation helps analyze MRI, CT scan images for disease detection and diagnosis.

Drug Discovery: Predicting drug interactions and designing new pharmaceutical compounds.

Challenges and Future Directions

Computational Resources: Training deep neural networks with backpropagation requires significant computational power and memory.

Exploding and Vanishing Gradients: Gradient issues can affect training stability, addressed by techniques like gradient clipping and batch normalization.

Advances in Optimization: Research continues on improving optimization algorithms for faster convergence and better generalization.

Structure	Typically consists of a few hidden layers	Consists of many hidden layers (typically more than 3)
Complexity	Relatively simpler and shallower	More complex and deeper
Representation Learning	Limited ability to learn complex representations	Capable of learning hierarchical representations
Feature Extraction	Basic feature extraction capabilities	Enhanced feature extraction through multiple layers
Performance	May struggle with complex tasks and large datasets	Better suited for complex tasks and big data
Training Time	Faster training times	Longer training times due to more layers and parameters
Computational Requirements	Lower computational requirements	Higher computational requirements
Applications	Suitable for simpler tasks like basic image recognition or regression	Used for advanced applications like natural language processing, computer vision, and deep reinforcement learning

Difference between Chatbot and Google (Search Engine)		
Aspect	Chatbot	Google Search Engine
Interaction	Interactive, interface conversational	Query-based, noninteractive search

Difference between Neural Network and Deep Neural Network

Purpose	Provides personalized assistance or information based on user queries	Retrieves relevant information from the web based on keywords
Capabilities	Can engage in dialogue, answer questions, provide recommendations	Crawls and indexes web pages, ranks results based on relevance
Scope	Limited to specific tasks or domains	Accesses vast information across the entire web
User Engagement	Emphasizes user interaction and engagement	Focuses on retrieving information efficiently
Examples	Customer service bots, virtual assistants	Google Search, Google Assistant

Part-of-Speech (POS) Tagging :

Part-of-Speech (POS) tagging is a fundamental task in Natural Language Processing (NLP) that involves assigning grammatical tags (such as noun, verb, adjective, etc.) to words in a text based on their syntactic context.

Here's an overview:

Purpose: POS tagging helps in understanding the grammatical structure of sentences, which is crucial for many downstream NLP tasks like parsing, machine translation, and sentiment analysis.

Techniques: POS tagging can be done using rule-based approaches, statistical models, or more advanced deep learning methods like recurrent neural networks (RNNs) or transformer models.

Applications:

Text Processing: Helps in information retrieval, text summarization, and keyword extraction.

Language Understanding: Aids in disambiguating word meanings and improving machine translation accuracy.

Grammar Checking: Used in grammar and syntax checking tools to enhance writing quality.

Challenges: Ambiguity in language, especially in homographs (words with multiple meanings), and handling of out-of-vocabulary (OOV) words are common challenges in POS tagging.

Object Detection :

Object detection is a computer vision task that involves identifying and localizing objects of interest within an image or video frame. Here's an overview:

Purpose: Object detection enables machines to recognize and locate multiple objects in an image, which is essential for applications like autonomous vehicles, surveillance, and augmented reality.

R-CNN (Region-based Convolutional Neural Networks)

It Uses region proposal networks to localize objects.

YOLO (You Only Look Once): A single-stage object detection model that predicts bounding boxes and class probabilities directly.

SSD (Single Shot MultiBox Detector): Combines efficiency and accuracy by predicting object classes and locations in a single pass.

Applications:

Autonomous Driving: Detecting pedestrians, vehicles, and traffic signs for safe navigation.

Security and Surveillance: Identifying intruders or suspicious activities in real-time.

Retail and Inventory Management: Monitoring product placement and inventory levels.

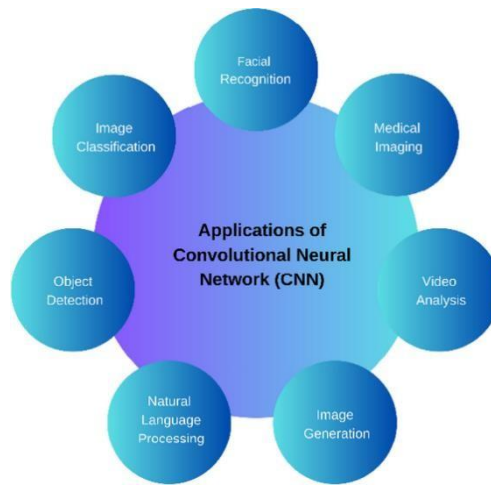
Challenges: Object detection faces challenges like handling occlusions (when objects overlap), scale variations, and robustness to different lighting conditions and viewpoints.

Convolutional Neural Network (CNN) Algorithm :

1. Introduction to CNNs

Purpose: CNNs are designed to process and analyze visual data, such as images and videos, by mimicking the human visual system's ability to extract hierarchical features.

Architecture: CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which learn increasingly complex representations of the input data.



2. Key Components of CNNs

Convolutional Layers:

Convolution Operation: Involves sliding a filter (kernel) over the input image to compute dot products, capturing spatial hierarchies of features (edges, textures, patterns).

Feature Maps: Output of convolutions, capturing different aspects of the input through learned filters.

Pooling Layers:

Pooling Operation: Reduces the spatial dimensions (width and height) of each feature map while retaining important information.

Types: Max pooling (retains maximum value), average pooling (computes average value), etc.

Activation Functions:

ReLU (Rectified Linear Unit): Commonly used activation function in CNNs to introduce non-linearity and improve convergence.

Others: Sigmoid and tanh functions are also used in certain contexts, especially in the output layers for binary or multi-class classification.

Fully Connected Layers:

Role: Typically found at the end of the network, these layers connect every neuron from one layer to every neuron in the next layer.

Classification: Used for making final predictions based on learned features extracted by earlier layers.

3. Training Process

Forward Propagation: Input data is passed through the network layer by layer, with each layer performing operations (convolutions, activations, pooling).

Loss Calculation: Compares the predicted output with the actual labels using a loss function (e.g., cross-entropy for classification).

Backpropagation: Error gradients are computed backward through the network, adjusting weights using optimization algorithms (e.g., SGD, Adam) to minimize the loss function.

4. Applications of CNNs :

Image Classification: Identifying objects in images and assigning them to specific categories (e.g., identifying cats vs. dogs).

Object Detection: Locating objects within an image and drawing bounding boxes around them.

Segmentation: Dividing an image into segments for detailed analysis.

Face Recognition: Recognizing faces in images or videos.

Medical Imaging: Analyzing medical images for diagnostic purposes.

5. Advanced CNN Architectures

LeNet-5: One of the earliest CNN architectures developed by Yann LeCun for handwritten digit recognition.

AlexNet: Introduced in 2012, significantly advanced the field with deeper layers and GPU acceleration.

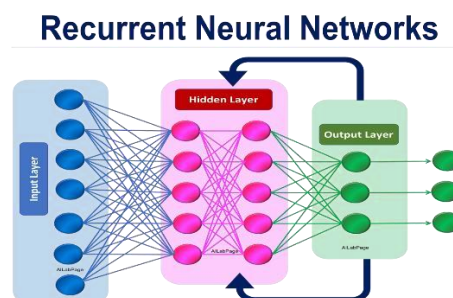
VGGNet: Known for its simplicity with multiple convolutional layers.

ResNet: Introduced residual connections to address vanishing gradient problem, enabling training of extremely deep networks.

Inception (GoogLeNet): Utilizes inception modules for efficient computation and feature extraction.

Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a type of neural network designed to process sequential data where dependencies exist between successive inputs. Unlike feedforward neural networks, RNNs have connections that form cycles, allowing them to maintain a state or memory of previous inputs.



Purpose: RNNs are suited for tasks like natural language processing (NLP), speech recognition, and time series prediction, where context and temporal relationships are crucial.

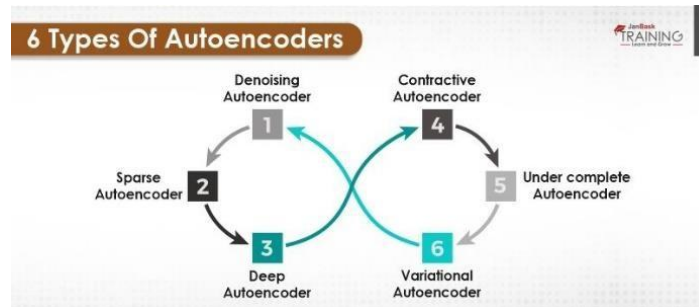
Architecture: RNNs consist of recurrent connections that enable information to persist. They process input sequences step-by-step, updating their internal state with each new input.

Applications: Used in machine translation (e.g., Google Translate), sentiment analysis, handwriting recognition, and generating text (e.g., predictive text in smartphones).

Challenges: RNNs can suffer from the vanishing gradient problem, where gradients diminish as they propagate back through time, affecting long-term dependencies.

Autoencoders and Types :

Autoencoders are a type of artificial neural network used for unsupervised learning, particularly in dimensionality reduction and data compression tasks. They consist of an encoder network that compresses the input data into a latent-space representation and a decoder network that reconstructs the original input from this representation.



Types:

Undercomplete Autoencoder: Restricts the size of the latent space, forcing the network to learn a compressed representation of the input data.

Overcomplete Autoencoder: Allows the latent space to have a larger dimensionality than the input, potentially learning more complex representations.

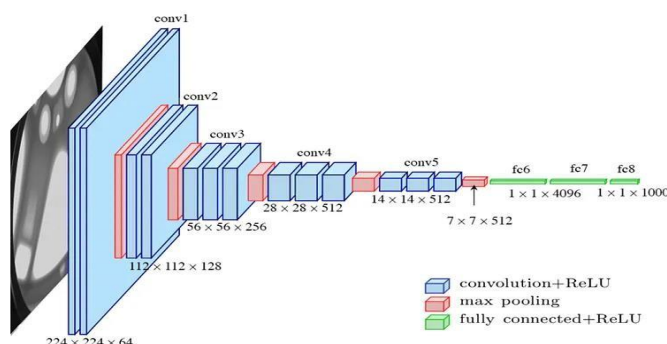
Denoising Autoencoder: Trained to recover the original input from a corrupted version, improving robustness to noise in data.

Variational Autoencoder (VAE): Incorporates probabilistic techniques to generate new data points similar to those in the training data, useful for tasks like image generation.

Applications: Used for feature extraction, anomaly detection (detecting unusual data points), image denoising, and generating synthetic data.

VGGNet and Architecture :

VGGNet (Visual Geometry Group Network) is a convolutional neural network architecture known for its simplicity and effectiveness in image recognition tasks. It was developed by the Visual Geometry Group at the University of Oxford.



Architecture:

Depth: VGGNet is characterized by its depth, consisting of 16 or 19 weight layers (convolutional and fully connected layers).

Convolutional Layers: Use small 3x3 convolutional filters with a stride of 1 and padding to maintain spatial resolution.

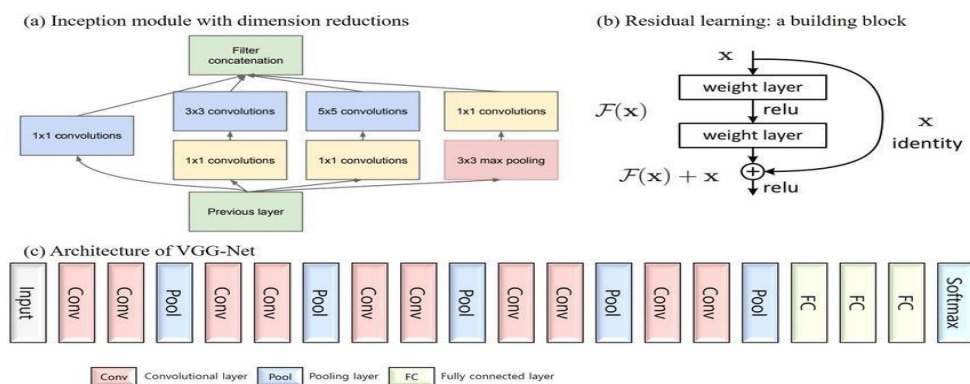
Pooling Layers: Employ max-pooling layers with 2x2 filters and a stride of 2 to reduce spatial dimensions.

Fully Connected Layers: Conclude the network with multiple fully connected layers for classification.

Advantages: VGGNet's uniform architecture and smaller filter sizes make it easier to understand and modify. It achieved competitive performance in image classification tasks, winning the ImageNet Large Scale Visual Recognition Challenge in 2014.

GoogleNet (Inception) and its Architecture :

GoogleNet, also known as Inception v1, is a deep convolutional neural network architecture developed by researchers at Google. It was designed to improve computational efficiency and accuracy in image classification and object detection tasks.



Architecture:

Inception Modules: Key feature of GoogleNet, consisting of multiple parallel convolutional layers of different filter sizes (1x1, 3x3, 5x5) and pooling operations.

Computational Efficiency: Uses global average pooling and 1x1 convolutions to reduce the number of parameters and computational cost.

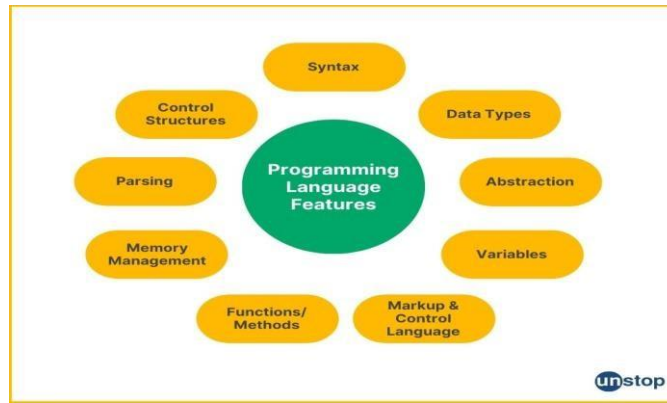
Depth: GoogleNet is deeper than previous architectures, with approximately 22 layers.

Auxiliary Classifiers: Includes auxiliary classifiers at intermediate layers to combat the vanishing gradient problem during training.

Advantages: GoogleNet achieved state-of-the-art performance on the ImageNet dataset in 2014 by balancing depth and computational efficiency, paving the way for subsequent versions like Inception v2, v3, and v4.

Programming Language :

A programming language is a formal set of instructions that a computer can interpret and execute to perform specific tasks or manipulate data. It serves as a medium for humans to communicate with computers, enabling the development of software applications, algorithms, and systems.



Key Aspects of Programming Languages:

Syntax and Semantics:

Syntax: Rules that dictate how instructions are structured and written in the language.

Semantics: Meaning behind the syntax, defining the behavior and operations the language can perform.

Types and Variables:

Types: Define the kind of data a variable can hold (e.g., integer, float, string, boolean).

Variables: Named containers that store data values, which can be manipulated and updated during program execution.

Control Structures:

Conditional Statements: Control program flow based on conditions (e.g., ifelse statements).

Loops: Repeat a block of code multiple times until a condition is met (e.g., for loops, while loops).

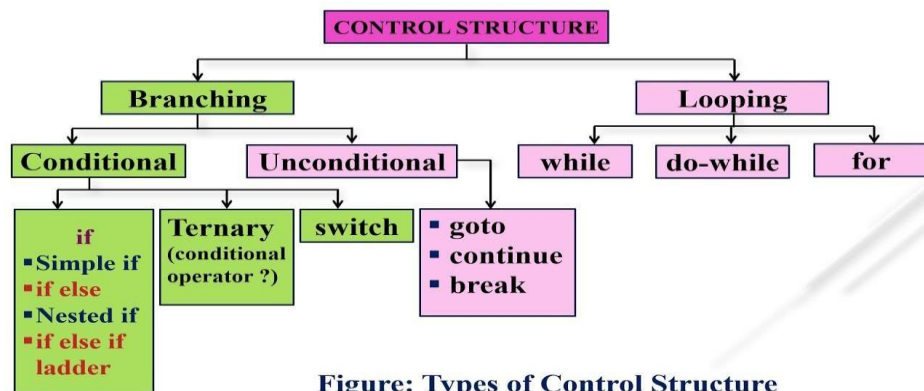


Figure: Types of Control Structure

Functions and Procedures:

Functions: Modular blocks of code that perform a specific task and can be reused throughout the program.

Procedures: Similar to functions but do not return a value (void functions).

Object-Oriented Programming (OOP):

Classes and Objects: Encapsulate data (attributes) and behaviors (methods) into objects.

Inheritance: Enables classes to inherit attributes and behaviors from other classes.

Polymorphism: Allows objects to be treated as instances of their parent class or their own class.

Libraries and Frameworks:

Libraries: Collections of pre-written code modules that extend the language's functionality.

Frameworks: Provide a structured way to build applications by offering reusable code, libraries, and tools.

Examples of Programming Languages:

High-Level Languages: Python, Java, C++, JavaScript, Ruby, Swift.

Scripting Languages: Perl, PHP, PowerShell, Bash.

Specialized Languages: SQL (for database management), R (for statistical computing), MATLAB (for numerical computing).

Python and Uses of Python

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Uses of Python:

Web Development: Frameworks like Django and Flask for building web applications.

Data Science: Analyzing data with libraries like NumPy, Pandas, and SciPy.

Machine Learning: Implementing algorithms with libraries such as TensorFlow and PyTorch.

Automation: Writing scripts for tasks like file handling, system management, and testing.

AI and Natural Language Processing: Developing applications for AI and NLP using libraries like NLTK and spaCy.

Data

Python supports several built-in data types:

Numeric Types: Integers (int), floating-point numbers (float), and complex numbers (complex).

Sequence Types: Lists (list), tuples (tuple), and range objects (range).

Text Type: Strings (str).

Mapping Type: Dictionaries (dict).

Set Types: Sets (set) and frozensets (frozenset).

Boolean Type: Boolean values (bool).

Arithmetic Operations

Python supports standard arithmetic operations:

Addition (+), Subtraction (-), Multiplication (*), Division (/).

Integer Division (//): Returns the quotient without the remainder.

Exponentiation (**): Raises a number to the power of another.

Modulo (%): Returns the remainder of division.

Reserved Words

Python has reserved words that cannot be used as identifiers (variable names or function names) because they are used for specific purposes within the language:

Examples: if, else, for, while, class, def, import, try, except, finally, global, nonlocal, return, True, False, None.

These reserved words have special meanings and cannot be redefined or used for other purposes in Python code.

Python, control statements are used to control the flow of execution in a program. They include conditional statements (if, elif, else) and loop statements (for, while). Here are very simple examples of each:

Conditional Statements 1.

if Statement

The if statement allows you to execute a block of code only if a specified condition is true.

x = 10 if x > 5:

```
print("x is greater than 5")
```

2. if-else Statement

The if-else statement executes one block of code if the condition is true, and another block if it is false.

x = 3 if x > 5: print("x is greater than 5") else:

```
print("x is not greater than 5")
```

3. if-elif-else Statement

The if-elif-else statement allows you to check multiple conditions and execute a block of code corresponding to the first true condition.

x = 7 if x > 10:

```
print("x is greater than 10") elif x > 5: print("x is greater than 5 but not greater than 10") else:
print("x is 5 or less")
```

Loop Statements 1.

for Loop

The for loop is used to iterate over a sequence (like lists, tuples, strings) or other iterable objects.

```
fruits = ["apple", "banana", "cherry"] for fruit in fruits: print(fruit)
```

2. while Loop

The while loop executes a block of code as long as a specified condition is true.

```
count = 0 while count < 5: print(count) count += 1
```

Break and Continue Statements 1.

break Statement

The break statement terminates the loop and transfers execution to the statement immediately following the loop.

```
fruits = ["apple", "banana", "cherry"] for fruit in fruits:  
print(fruit) if fruit == "banana": break
```

2. continue Statement

The continue statement skips the remaining code inside the loop for the current iteration and moves to the next iteration.

```
fruits = ["apple", "banana", "cherry"] for fruit in fruits: if fruit == "banana": continue print(fruit)
```

Example programs in python:

1. Prime Number

A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

```
def is_prime(n): if n <= 1: return False for i in range(2, int(n**0.5) + 1): if n % i == 0:  
return False return True  
  
number = 17 if is_prime(number): print(f"{number} is a prime number.") else: print(f"{number} is not  
a prime number.")
```

2. Factors of a Number

Factors of a number are integers that divide the number without leaving a remainder.

```
def find_factors(n):    factors = []    for i in range(1, n + 1):        if n % i == 0:            factors.append(i)    return factors

number = 24    print(f"The factors of {number} are: {find_factors(number)}")
```

3. Reverse a Number

Reversing a number involves reversing its digits.

```
def reverse_number(n):    rev = 0    while n > 0:        remainder = n % 10        rev = rev * 10 + remainder        n = n // 10    return rev

number = 12345    print(f"The reverse of {number} is: {reverse_number(number)}")
```

Problem Statement and Explanation

Problem Statement:

Exploratory Data Analysis (EDA) on Global Air Pollution **Introduction:**

Air pollution is a critical global challenge impacting public health and the environment. With rising industrialization and urbanization, pollutants like PM2.5, PM10, NO2, and SO2 have escalated, necessitating a thorough understanding through Exploratory Data Analysis (EDA).

Objective:

Methodology:

<ul style="list-style-type: none">• <p>Aggregate and preprocess data to ensure consistency and handle missing values.</p>	Data Collection and Cleaning:
<ul style="list-style-type: none">• <p>Conduct descriptive statistics to:</p> <ul style="list-style-type: none">• pollution datasets• pollutant concentrations.• demographic and industrial factors.• environmental policies and interventions. <p>Summarize pollutant levels and distributions. Visualize temporal and spatial trends using charts and maps.</p>	Exploratory Data Analysis: This EDA aims to analyze global air Identify trends and spatial patterns in Explore correlations with Provide insights for effective
<ul style="list-style-type: none">• <p>Identify regions most affected by air pollution and assess improvement or deterioration trends.</p> <p>Propose actionable recommendations for policymakers and stakeholders based on findings.</p> <p>Summary: This concise EDA on global air pollution seeks to uncover patterns and correlations in pollutant concentrations, offering insights to support effective environmental policies and initiatives aimed at improving air quality globally.</p>	Insights and Recommendations:

PROBLEM STATEMENT

AND

EXPLANATION

Problem Statement:

Develop a machine learning model to accurately predict the species of iris flowers based on their morphological features: sepal length, sepal width, petal length, and petal width. The species to be predicted are Iris setosa, Iris versicolor, and Iris virginica.

Explanation:

The Iris flower dataset is a benchmark dataset in machine learning, introduced by Ronald Fisher in 1936. It comprises 150 samples, each with four numerical features: sepal length, sepal width, petal length, and petal width. These features are used to classify the samples into one of three species of iris flowers: Iris setosa, Iris versicolor, and Iris virginica.

The task is to build a classification model that can predict the species of an iris flower given its morphological measurements. This involves several steps, including data preprocessing, feature analysis, model selection, training, and evaluation. The main challenges include dealing with class imbalance, understanding the relationships between features, and selecting the most appropriate algorithm to achieve high accuracy and robustness.

A successful model will accurately classify new, unseen data, demonstrating strong generalization capabilities. The model's performance will be assessed using various metrics:

- **Accuracy** measures the overall correctness of the predictions.
- **Confusion Matrix** provides a detailed breakdown of true vs. predicted classifications, highlighting misclassifications.
- **Precision, Recall, and F1-Score** offer insights into the model's performance for each class, crucial for evaluating how well the model handles class imbalance.

By leveraging techniques such as cross-validation and hyperparameter tuning, the model can be optimized to provide reliable and accurate predictions. The ultimate goal is to create a robust classifier that can effectively distinguish between the three species of iris flowers based on their morphological characteristics, aiding in botanical studies and applications where species identification is necessary.

-SOURCE CODE AND OUTPUT

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
[2]: data = pd.read_csv('supermarket_sales - Sheet1.csv')
```

```
[3]: data.head(5)
```

```
[3]:
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income	Rating
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	1/5/2019	13:08	Ewallet	522.83	4.761905	26.1415	9.1
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	3/8/2019	10:29	Cash	76.40	4.761905	3.8200	9.6
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	3/3/2019	13:23	Credit card	324.31	4.761905	16.2155	7.4
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880	8.4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	2/8/2019	10:37	Ewallet	604.17	4.761905	30.2085	5.3


jupyter
SUPERMARKETSALES TEAM2 BCA SADCBVM
Last Checkpoint: 13 minutes ago


File Edit View Run Kernel Settings Help

Python 3 (ipykernel)



```

[4]: data.shape
[4]: (1000, 17)

[5]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Invoice ID             1000 non-null   object
1   Branch                1000 non-null   object
2   City                  1000 non-null   object
3   Customer type         1000 non-null   object
4   Gender                1000 non-null   object
5   Product line          1000 non-null   object
6   Unit price            1000 non-null   float64
7   Quantity              1000 non-null   int64
8   Tax 5%                1000 non-null   float64
9   Total                 1000 non-null   float64
10  Date                  1000 non-null   object
11  Time                  1000 non-null   object
12  Payment               1000 non-null   object
13  cogs                  1000 non-null   float64
14  gross margin percentage 1000 non-null   float64
15  gross income          1000 non-null   float64
16  Rating                1000 non-null   float64
dtypes: float64(7), int64(1), object(9)
memory usage: 132.9+ KB

```


jupyter
SUPERMARKETSALES TEAM2 BCA SADCBVM
Last Checkpoint: 23 minutes ago


File Edit View Run Kernel Settings Help

Python 3 (ipykernel)

```

[6]: data.describe()

```

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income	Rating
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000
mean	55.672130	5.510000	15.379369	322.966749	307.58738	4.761905e+00	15.379369	6.97270
std	26.494628	2.923431	11.708825	245.885335	234.17651	6.131498e-14	11.708825	1.71858
min	10.080000	1.000000	0.508500	10.678500	10.17000	4.761905e+00	0.508500	4.00000
25%	32.875000	3.000000	5.924875	124.422375	118.49750	4.761905e+00	5.924875	5.50000
50%	55.230000	5.000000	12.088000	253.848000	241.76000	4.761905e+00	12.088000	7.00000
75%	77.935000	8.000000	22.445250	471.350250	448.90500	4.761905e+00	22.445250	8.50000
max	99.960000	10.000000	49.650000	1042.650000	993.00000	4.761905e+00	49.650000	10.00000



```
[7]: data.isna().sum()
```

```
[7]: Invoice ID      0
      Branch        0
      City          0
      Customer type 0
      Gender         0
      Product line   0
      Unit price     0
      Quantity       0
      Tax 5%         0
      Total          0
      Date           0
      Time           0
      Payment        0
      cogs           0
      gross margin percentage 0
      gross income   0
      Rating         0
      dtype: int64
```

```
[8]: data.duplicated().sum()
```

```
[8]: 0
```



```
[9]: data.columns
```

```
[9]: Index(['Invoice ID', 'Branch', 'City', 'Customer type', 'Gender',
         'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Date',
         'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income',
         'Rating'],
         dtype='object')
```

```
[10]: data.dtypes
```

```
[10]: Invoice ID      object
      Branch        object
      City          object
      Customer type  object
      Gender         object
      Product line   object
      Unit price     float64
      Quantity       int64
      Tax 5%         float64
      Total          float64
      Date           object
      Time           object
      Payment        object
      cogs           float64
      gross margin percentage float64
      gross income   float64
      Rating         float64
      dtype: object
```

```
[11]: def display_pie_chart(values, labels, title, chart_type = 'pie'):
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_subplot()
    if chart_type == 'doughnat':
        explode = np.full(len(labels), 0.05)
        ax.pie(values, labels=labels, autopct='%1.1f%%', startangle=90, explode = explode, pctdistance=0.85)
        centre_circle = plt.Circle((0, 0), 0.70, fc='white')
        fig = plt.gcf()
        fig.gca().add_artist(centre_circle)
    else:
        ax.pie(values, labels=labels, autopct='%1.1f%%', startangle=90)
    ax.axis('equal')
    ax.set_title(title)

    plt.savefig(f'{labels[0]}', bbox_inches='tight')
    plt.show()

[12]: def group_by_column(df, col, function):
    grouped_df = df.groupby(col).agg(function).reset_index()
    grouped_df = grouped_df.set_index(col)

    return grouped_df

[13]: gender_and_total = data[['Gender', 'Total']]

grouped_by_gender_df = group_by_column(gender_and_total, 'Gender', 'sum')

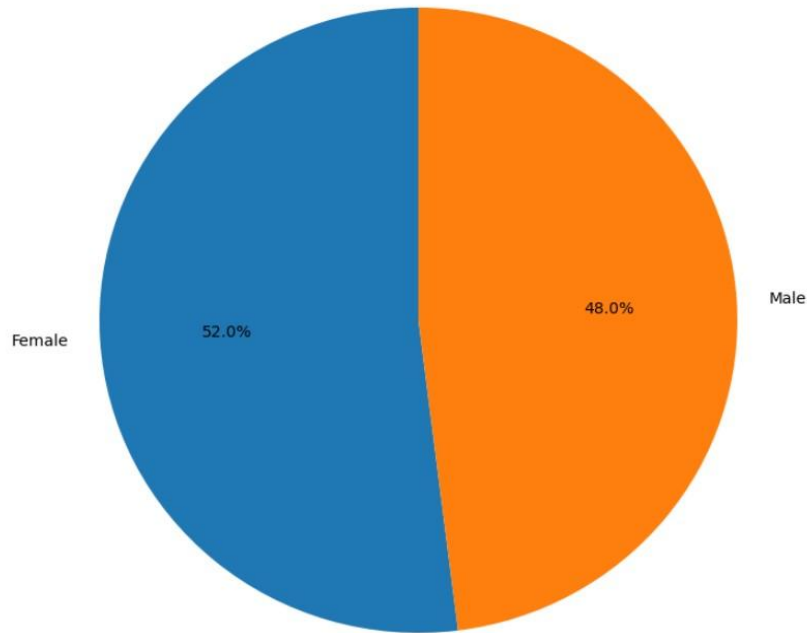
print(grouped_by_gender_df)

labels = grouped_by_gender_df.index
values = grouped_by_gender_df['Total']

display_pie_chart(values, labels, 'Total Sales by Gender')
```

```
Total
Gender
Female 167882.925
Male 155083.824
```

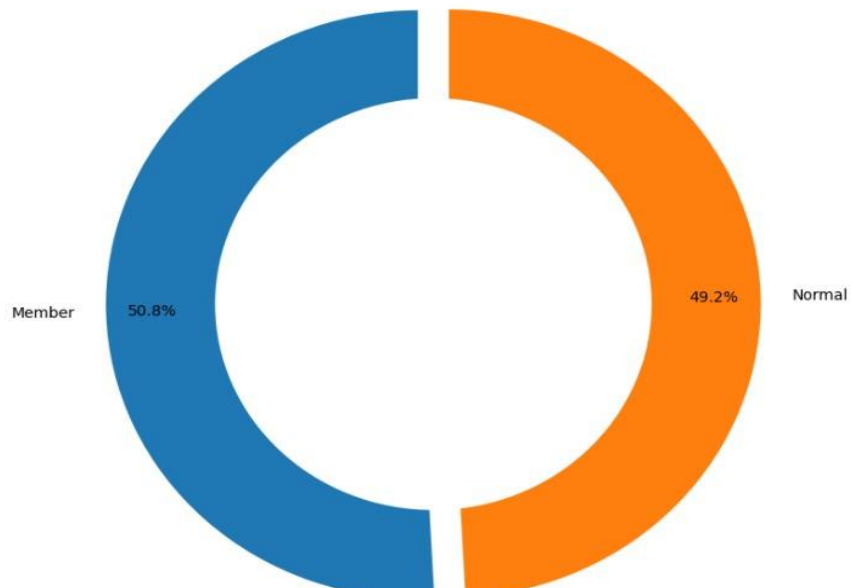
Total Sales by Gender



```
[14]: group_by_type = data[['Customer type', 'Total']]
      group_by_type_df = group_by_column(group_by_type, 'Customer type', 'sum')
      print(group_by_type_df)
      labels = group_by_type_df.index
      values = group_by_type_df['Total']
      display_pie_chart(values, labels, 'Total Sales by Customer type', chart_type = 'doughnat')
```

```
Total
Customer type
Member 164223.444
Normal 158743.305
```

Total Sales by Customer type




```
[15]: group = data.groupby(['Product line', 'Gender']).size().reset_index()
group.columns = ['Product line', 'Gender', 'Count']

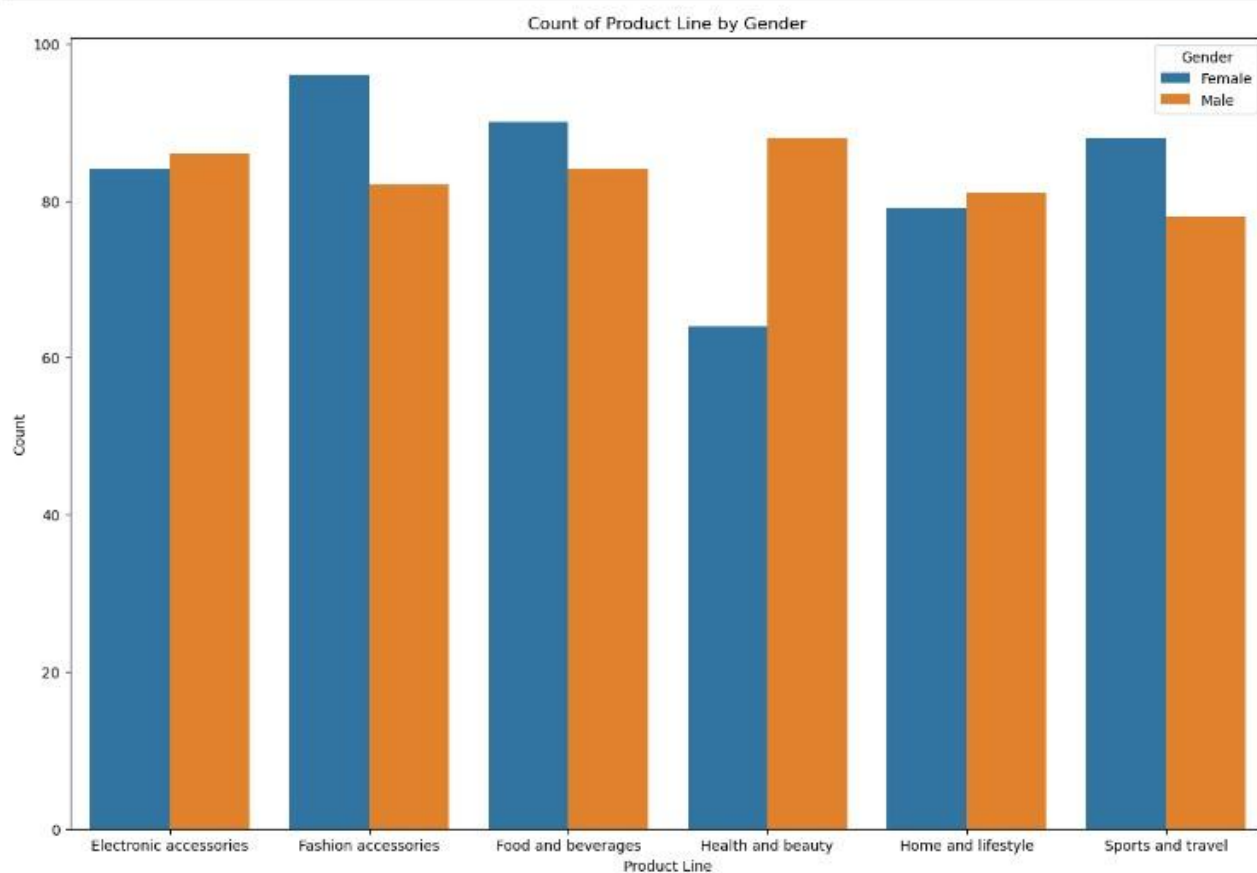
group.head(5)
```

```
[15]:
```

	Product line	Gender	Count
0	Electronic accessories	Female	84
1	Electronic accessories	Male	86
2	Fashion accessories	Female	96
3	Fashion accessories	Male	82
4	Food and beverages	Female	90

```
[16]: plt.figure(figsize=(15, 10))

sns.barplot(data=group, x="Product line", y="Count", hue="Gender")
plt.title("Count of Product Line by Gender")
plt.xlabel("Product Line")
plt.ylabel("Count")
plt.savefig('count_by_gender')
plt.show()
```



```
[17]: Yangon ---> A
      Naypyitaw ---> C
      Mandalay ---> B

      Cell In[17], line 1
      Yangon ---> A
              ^
      SyntaxError: invalid syntax

[18]: data.City.unique()

[18]: array(['Yangon', 'Naypyitaw', 'Mandalay'], dtype=object)

[19]: branches_sales = data.groupby('Branch')['Total'].sum()

      for branch, sales in branches_sales.items():
          print(f'Branch {branch} : {sales}')

      print('-' * 50)

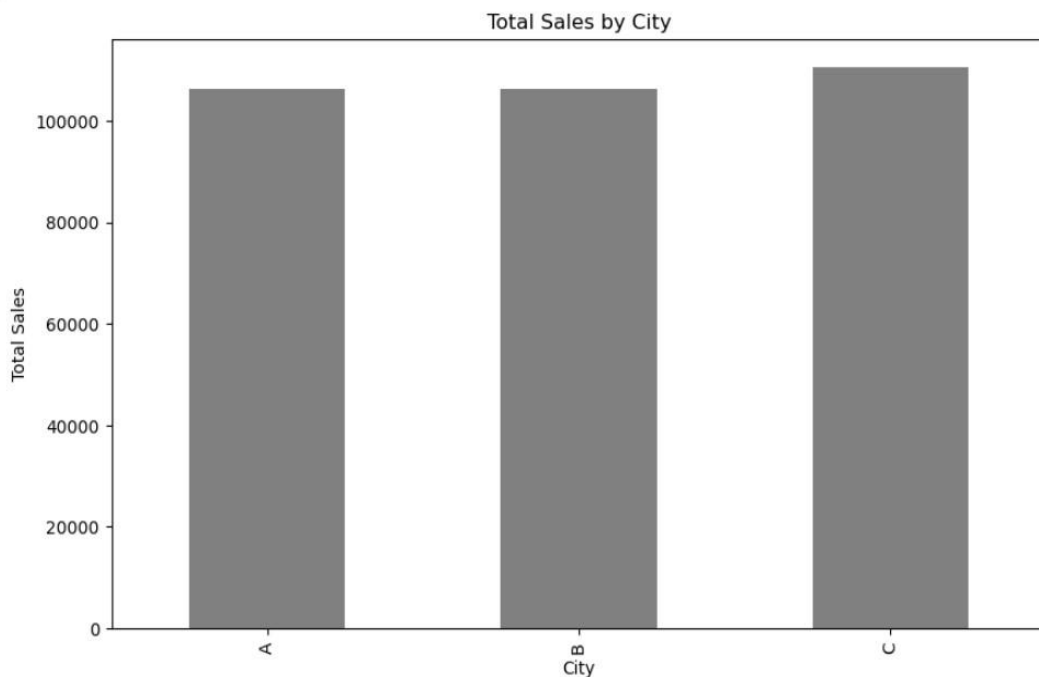
      most_selling_branch = branches_sales.idxmax()

      print(f'The most selling branch is {most_selling_branch}')

      Branch A : 106200.3705
      Branch B : 106197.672
      Branch C : 110568.7065
      -----
      The most selling branch is C
```

```
[20]: plt.figure(figsize=(10, 6))

      branches_sales.plot(kind='bar', color='grey')
      plt.title('Total Sales by City')
      plt.xlabel('City')
      plt.ylabel('Total Sales')
      plt.show()
```



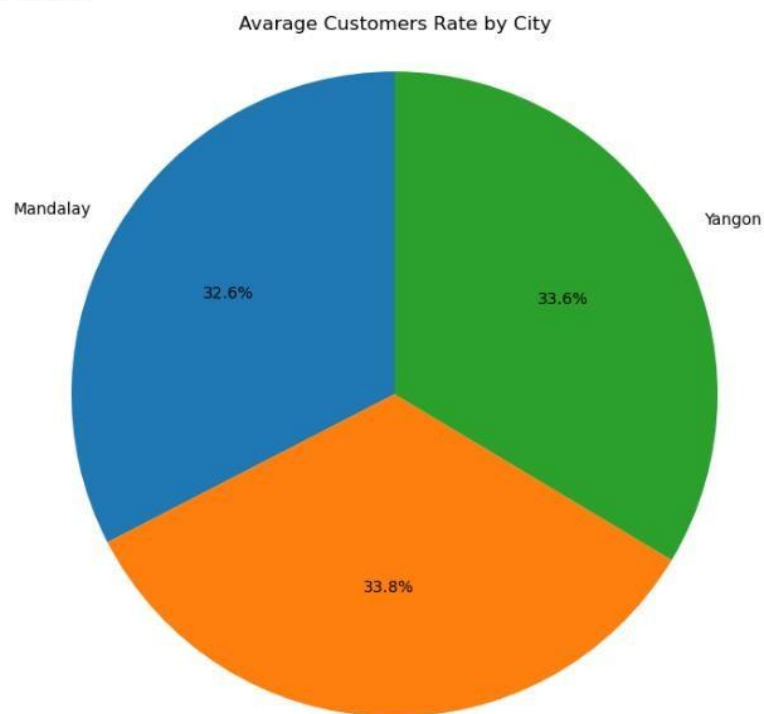
```
[21]: customer_rate = data.groupby('City')['Rating'].mean()

cities = customer_rate.index
rates = customer_rate

print(customer_rate)

display_pie_chart(rates, cities, 'Average Customers Rate by City')

City
Mandalay    6.818072
Naypyitaw   7.072866
Yangon      7.027059
Name: Rating, dtype: float64
```




```
[22]: product_sales = data.groupby('Product line')['Quantity'].sum()

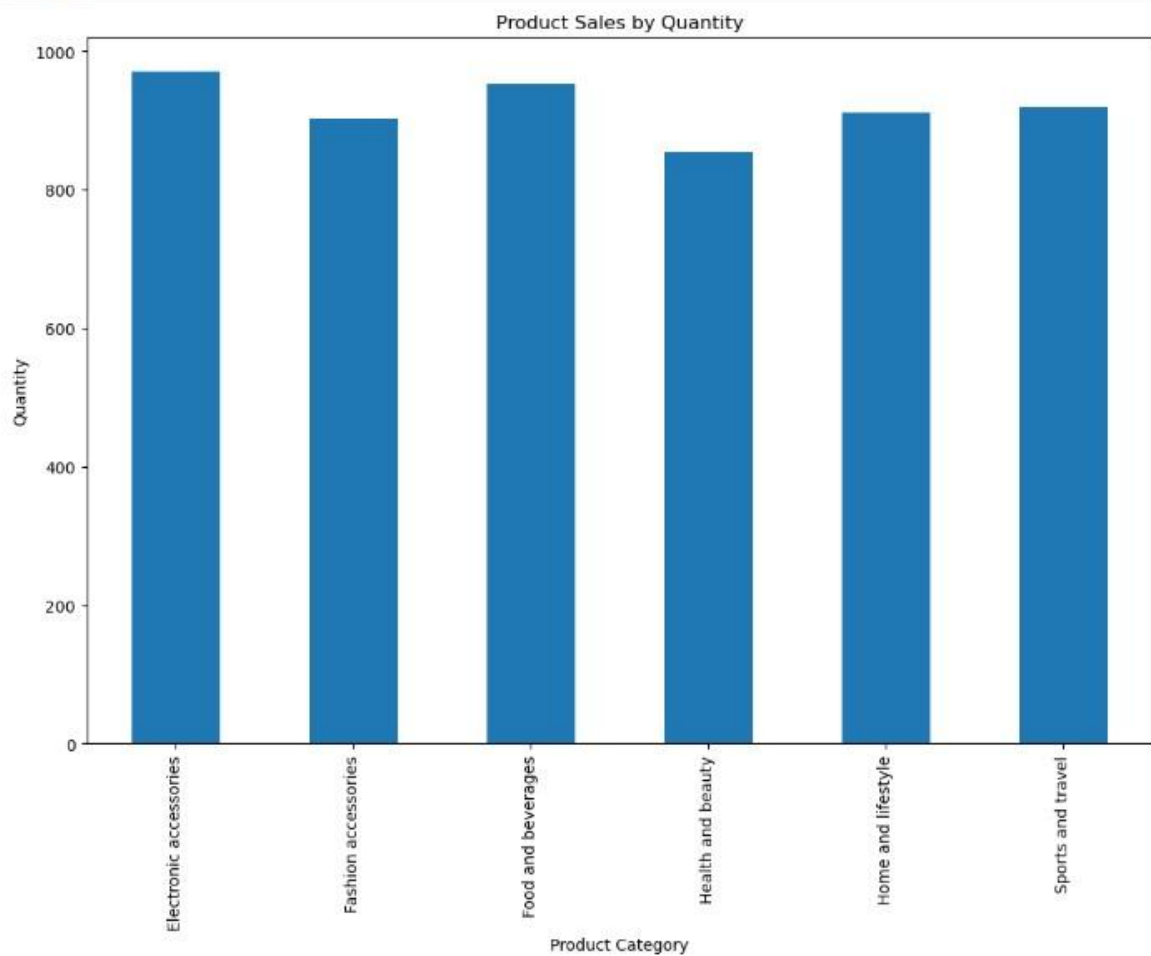
highest_selling_product = product_sales.idxmax()

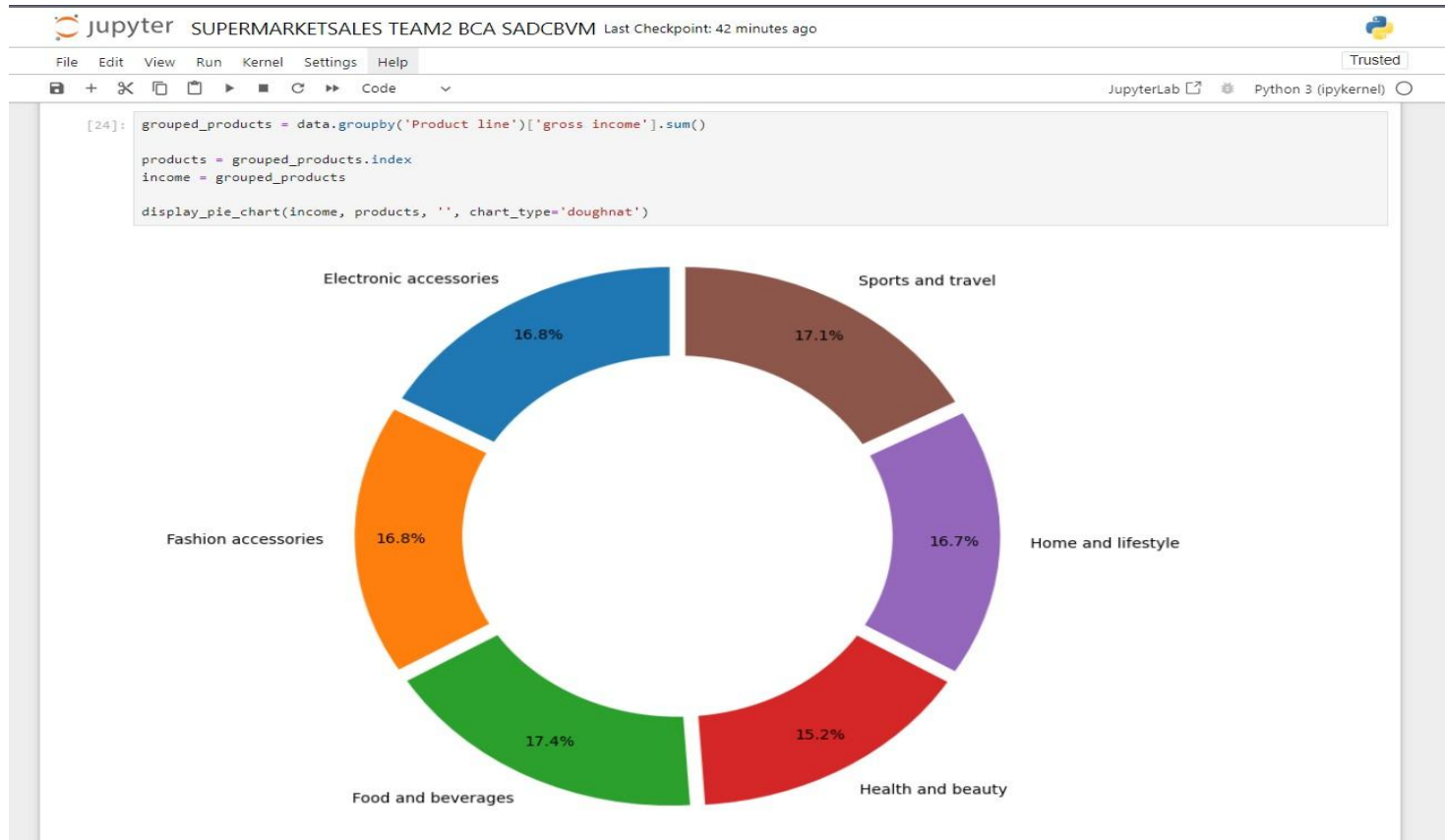
print("Highest selling product:", highest_selling_product)

Highest selling product: Electronic accessories
```

```
[23]: plt.figure(figsize=(12, 8))

product_sales.plot(kind='bar')
plt.title('Product Sales by Quantity')
plt.xlabel('Product Category', rotation=0)
plt.ylabel('Quantity')
plt.show()
```





Jupyter SUPERMARKETSALES TEAM2 BCA SADCBVM Last Checkpoint: 43 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
[25]: dates = pd.DataFrame()

# Convert Date values in dataset to pd.datetime object
dates['Date'] = pd.to_datetime(data['Date'])

# separate year value form datetime object
dates['Year'] = dates['Date'].dt.year

# separate month value form datetime object
dates['Month'] = dates['Date'].dt.month

# separate day value form datetime object
dates['Day'] = dates['Date'].dt.day

# Get Total column values from the original dataframe
dates['Total'] = data['Total']

dates
```

```
[25]:
```

	Date	Year	Month	Day	Total
0	2019-01-05	2019	1	5	548.9715
1	2019-03-08	2019	3	8	80.2200
2	2019-03-03	2019	3	3	340.5255
3	2019-01-27	2019	1	27	489.0480
4	2019-02-08	2019	2	8	634.3785
...
995	2019-01-29	2019	1	29	42.3675
996	2019-03-02	2019	3	2	1022.4900
997	2019-02-09	2019	2	9	33.4320
998	2019-02-22	2019	2	22	69.1110
999	2019-02-18	2019	2	18	649.2990

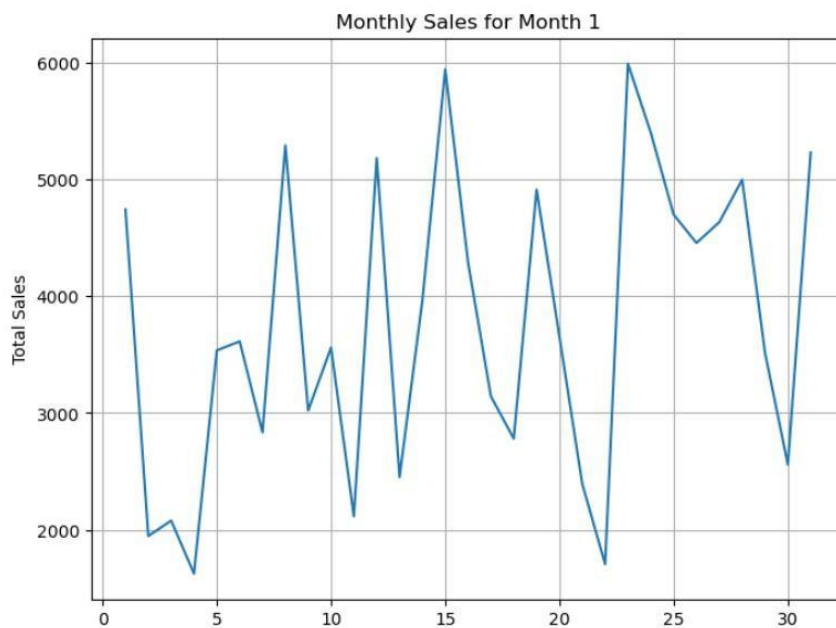
1000 rows x 5 columns

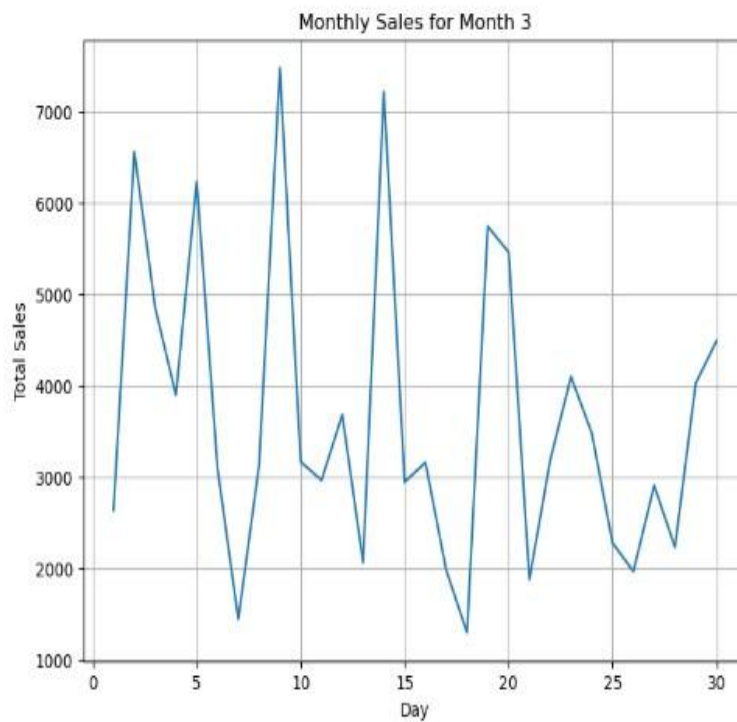
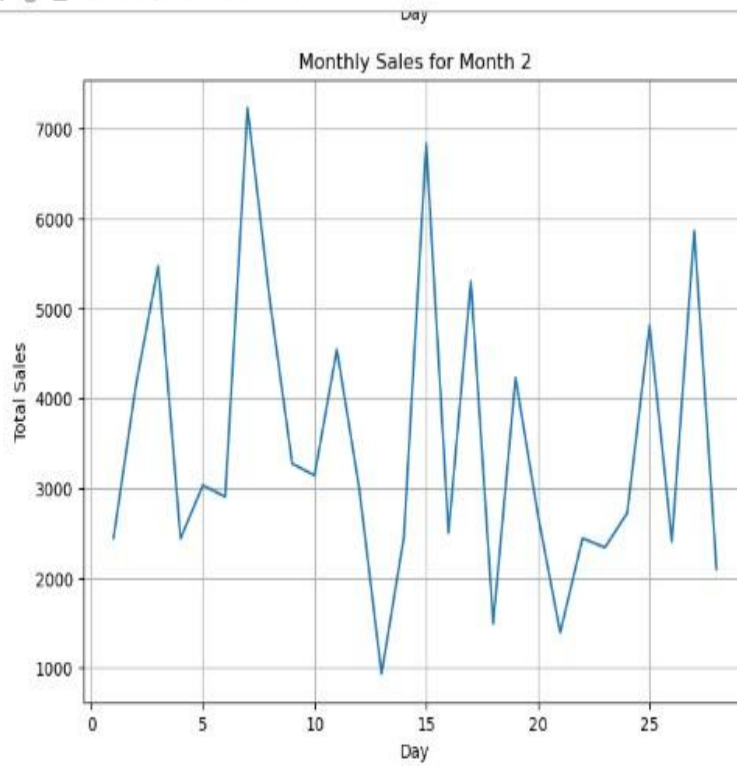

```
[26]: monthly_sales = dates.groupby(['Month', 'Day'])['Total'].sum()
```

monthly_sales

```
[26]: Month Day
1      1      4745.1810
      2      1945.5030
      3      2078.1285
      4      1623.6885
      5      3536.6835
      ...
3      26      1962.5130
      27      2902.8195
      28      2229.4020
      29      4023.2430
      30      4487.0595
Name: Total, Length: 89, dtype: float64
```

```
[27]: for month in range(1, 4):
      plt.figure(figsize=(8, 6))
      month_data = monthly_sales.loc[month]
      plt.plot(month_data.index.get_level_values('Day'), month_data.values)
      plt.title(f'Monthly Sales for Month {month}')
      plt.xlabel('Day')
      plt.ylabel('Total Sales')
      plt.grid(True)
      plt.savefig(f'{month}.png')
      plt.show()
```





[]:

CONCLUSION

In conclusion, Supermarkets typically are chain stores, supplied by the distribution centers of their parent companies, thus increasing opportunities for economies of scale. Supermarkets usually offer products at relatively low prices by using their buying power to buy goods from manufacturers at lower prices than smaller stores can. They also minimize financing costs by paying for goods at least 30 days after receipt and some extract credit terms of 90 days or more from vendors. Certain products (typically staple foods such as bread, milk and sugar) are very occasionally sold as loss leaders so as to attract shoppers to their store. Supermarkets make up for their low margins by a high volume of sales, and with of higher-margin items bought by the customers. Self-service with shopping carts (trolleys) or baskets reduces labor costs, and many supermarket chains are attempting further reduction by shifting to self-service check-outs