SPEECH RECOGNITION

NAME -
SUDHANSHU KAKKAR

STUDENT ID -
20036779

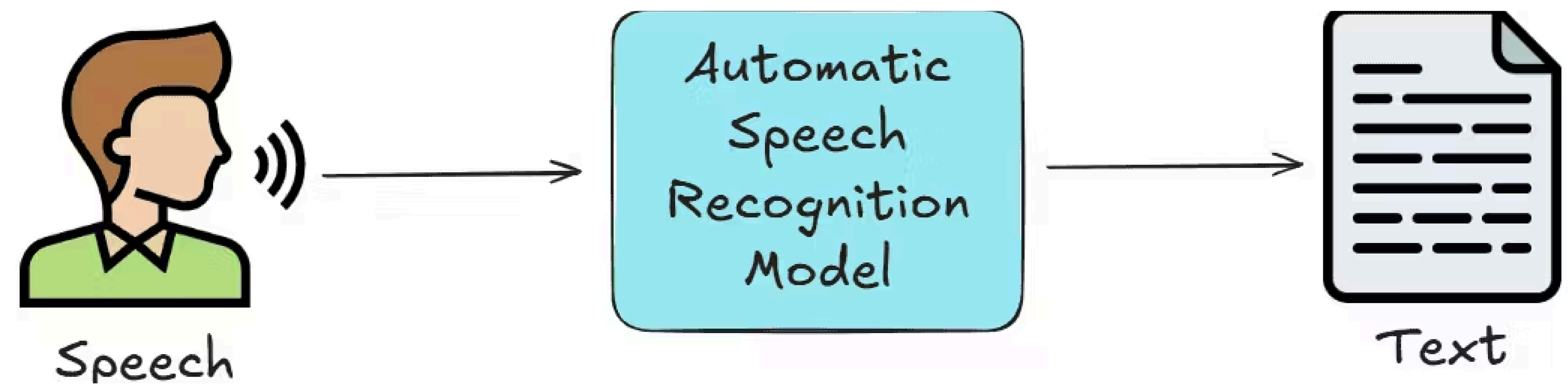
ENABLING MACHINES TO UNDERSTAND AND PROCESS HUMAN SPEECH

INTRODUCTION

Speech recognition is the process of converting spoken language into text using AI-powered systems.

APPLICATIONS

- Virtual assistants (Siri, Alexa, Google Assistant)
- Healthcare (medical dictation, voice-driven diagnostics)
- Customer service (call center automation, chatbots)
- Accessibility (voice control for differently-abled users)
- Transcription (meetings, lectures, media production)



MACHINE LEARNING (ML):

Role: Learns from speech datasets to improve recognition of accents, dialects, and context.

Example: Logistic regression or SVM models used for phoneme classification

DEEP LEARNING (DL)

Role: Neural networks (e.g., RNNs, Transformers) model sequences of speech for high accuracy.

Example: End-to-end systems like DeepSpeech achieving near-human transcription performance.

LARGE LANGUAGE MODELS (LLMS)

Role: Add contextual understanding to speech recognition, correcting errors and predicting intent.

Example: GPT-4 used to enhance accuracy in domain-specific voice transcription.

ARTIFICIAL INTELLIGENCE (AI):

Role: Provides the foundation for speech-to-text pipelines by combining language understanding and signal processing.

Example: AI systems that filter background noise to improve recognition accuracy.

CHATGPT

Role: Processes recognized text into meaningful dialogue, generating natural responses.

Example: Integrated with voice assistants for conversational AI.

NAME - **SUDHANSHU KAKKAR** ID - **20036779**

MATLAB = MATH + CODE + VISUALIZATION

MATLAB

Powerful, tool for engineers & scientists to analyze, simulate, and visualize real-world problems.

AVAILABLE: WINDOWS, MACOS, LINUX
LANGUAGE: C++, C, JAVA, MATLAB

KEY FEATURES

- Matrix-based computing
- 2D & 3D visualization
- Toolboxes for AI, Image Processing, Control Systems, Finance
- Easy integration with Python, C/C++, hardware

APPLICATIONS

- Data Analysis & Visualization 📊
- Signal & Image Processing 🎵🖼️
- Machine Learning & AI 🤖
- Robotics & Control ⚙️
- Financial Modeling 💰

HW-Topic-3

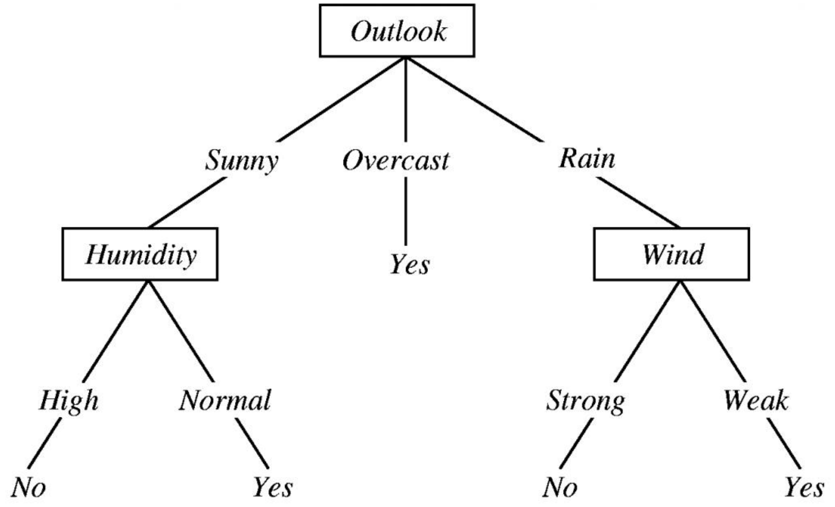
Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

DLT – Decision Tree Learning

WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Represents decisions as a **tree structure.**



KEY CONCEPTS

- **Splitting:** Dividing dataset based on feature values.
- **Entropy:** Measure of randomness or impurity.
- **Information Gain:** Reduction in entropy after a split.
- **Pruning:** Removing branches to reduce overfitting.

TYPES

- **Classification Tree:** Predicts discrete labels (e.g., Yes/No)
- **Regression Tree:** Predicts continuous values

Pros	Cons
• Easy to understand & visualize • Handles categorical & numerical data	• Prone to overfitting • Sensitive to small changes in data

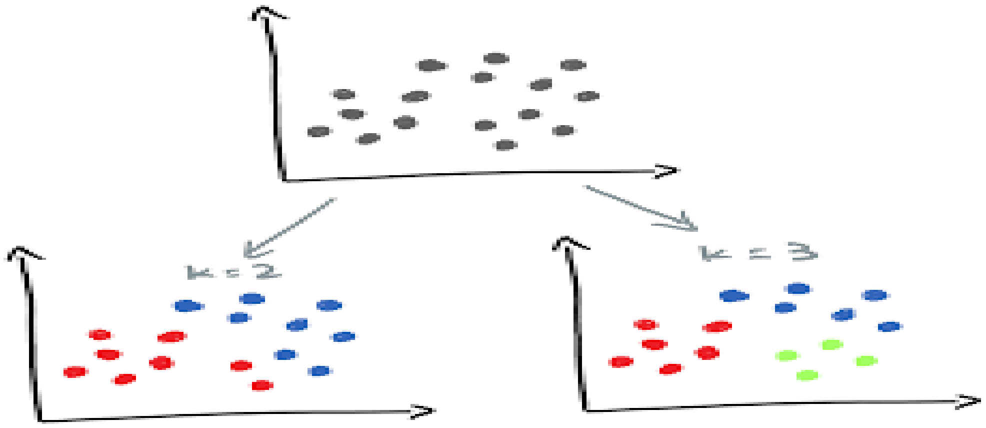
KNN – K-Nearest Neighbours

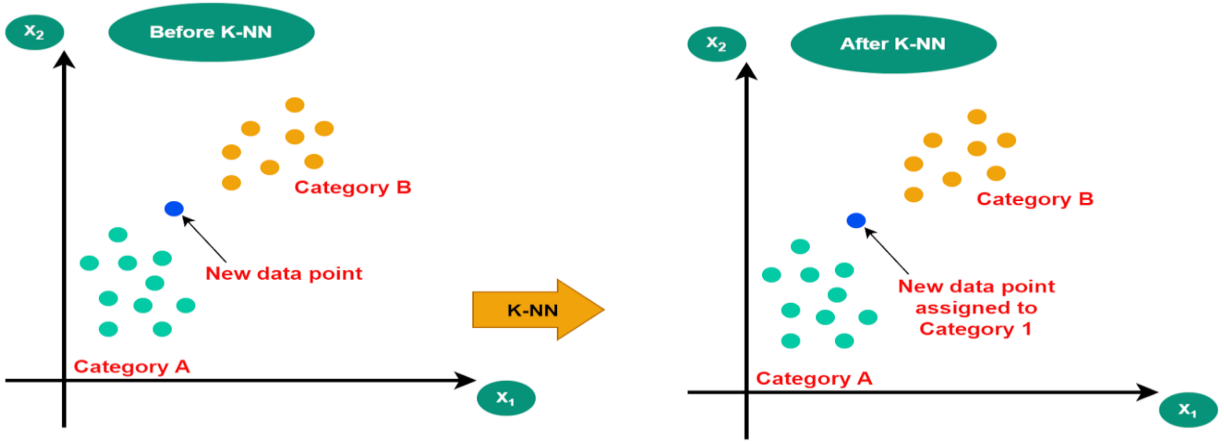
WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Classifies a data point based on the **majority class of its K nearest neighbors**

KEY CONCEPTS

- **K Value:** Number of neighbors considered
- **Distance Metrics:** How similarity is measured
    - **Euclidean Distance**
    - **Manhattan Distance**





Pros	Cons
• Simple and easy to understand • Works well with small datasets	• Computationally expensive for large datasets • Poor performance on high-dimensional data

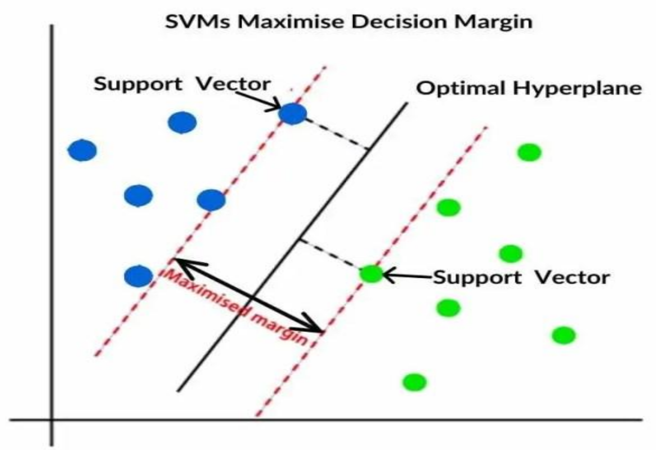
SVM – Support Vector Machine

WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Finds the **best hyperplane** that separates classes with the **maximum margin**.
- Works well for **linear and non-linear** data using kernel functions.

KEY CONCEPTS

- **Hyperplane:** Decision boundary separating classes
- **Support Vectors:** Data points closest to the hyperplane
- **Margin:** Distance between support vectors of different classes (maximizing this improves generalization)



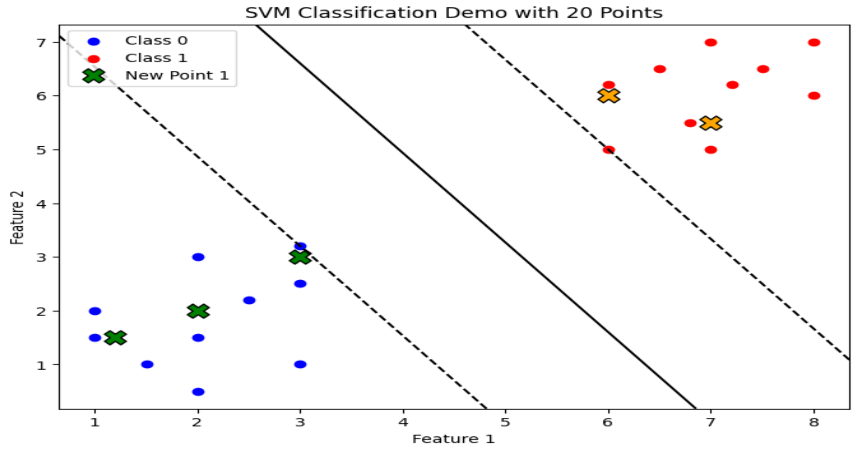
TYPES

- **Linear SVM:** Data is linearly separable
- **Non-linear SVM:** Uses kernels for complex datasets

Pros	Cons
• Effective in high-dimensional spaces • Works well for linearly and non-linearly separable data	• Computationally expensive for very large datasets • Choosing the right kernel can be tricky

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# declaring points for training
X = np.array([
    [1,2], [2,3], [3,1], [2,1.5], [3,2.5],
    [1.5,1], [2.5,2.2], [3,3.2], [2,0.5], [1,1.5],
    [6,5], [7,7], [8,6], [7,5], [6.5,6.5],
    [8,7], [7.2,6.2], [6.8,5.5], [7.5,6.5], [6,6.2]
])
# declaring labels for points
y = np.array([0]*10 + [1]*10)  # 0 = Class A, 1 = Class B

# Train SVM
model = svm.SVC(kernel='linear')
model.fit(X, y)

# New points to classify
new_points = np.array([
    [2,2], [3,3], [6,6], [7,5.5], [1.2,1.5]
])
predictions = model.predict(new_points)

# Print classifications
for point, label in zip(new_points, predictions):
    print(f"Point {point} is classified as Class {label}")

# Plot
plt.figure(figsize=(8,6))

# Plot original points
plt.scatter(X[:10,0], X[:10,1], color='blue', label='Class 0')
plt.scatter(X[10:,0], X[10:,1], color='red', label='Class 1')
```

SVM Code Demo (python)



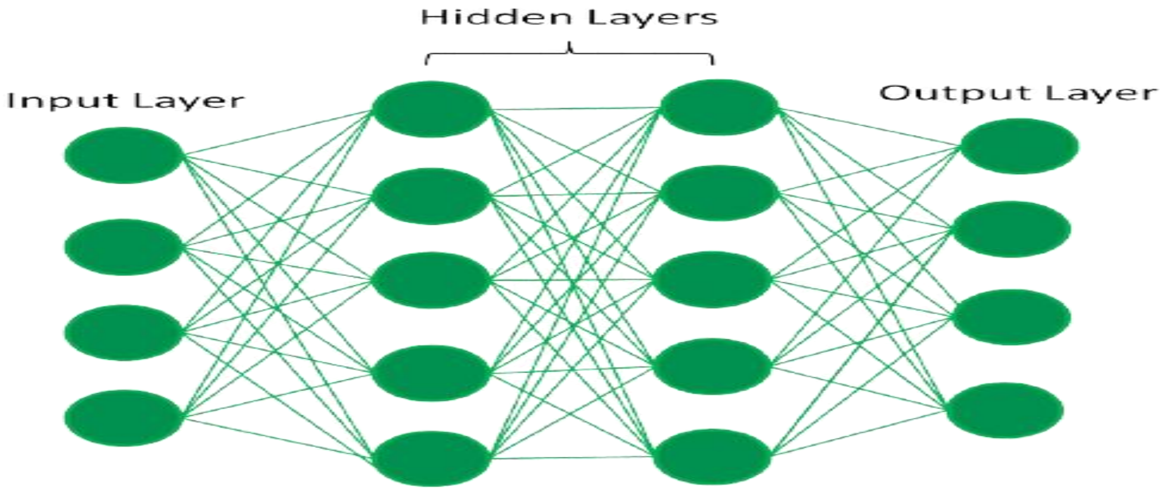

```python
# Plot new points with classification
for i, point in enumerate(new_points):
    plt.scatter(point[0], point[1], color='green' if predictions[i]==0 else 'orange',
                edgecolors='k', s=150, marker='X', label=f'New Point {i+1}' if i==0 else
"")


# Plot decision boundary
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
xx = np.linspace(xlim[0], xlim[1], 50)
yy = np.linspace(ylim[0], ylim[1], 50)
YY, XX = np.meshgrid(yy, xx)
Z = model.decision_function(np.c_[XX.ravel(), YY.ravel()]).reshape(XX.shape)
ax.contour(XX, YY, Z, levels=[0], colors='k')      # hyperplane
ax.contour(XX, YY, Z, levels=[-1,1], colors='k', linestyles='--')  # margins


plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("SVM Classification Demo with 20 Points")
plt.legend()
plt.show()
```

HW-Topic-4

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

Deep Learning



WHAT IS IT?

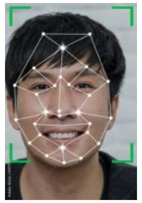
Deep learning is a subfield of **machine learning** that uses **artificial neural networks** with multiple layers (hence "deep") to analyze data, learn patterns, and make decisions or predictions.

KEY CONCEPTS

- **Neural Networks:** Artificial neural networks with multiple hidden layers (deep) form the computational core, inspired by the human brain.
- **Feature Extraction:** Deep learning automatically learns the most important features directly from raw data, eliminating manual feature engineering.

APPLICATIONS

- Self Driving Cars
- Natural Language Processing
- Movie Recommendation Systems
- Natural Language Processing
- Computer Vision
- Financial Fraud Detection
- Healthcare



THE PROCESS

1. **Forward Propagation:** The input data is processed through the network's layers, where initial weights are applied to make the model's first prediction (the "guess").

2. **Measure Loss:** A mathematical function calculates the difference (the "loss") between the model's prediction and the actual correct answer, measuring how wrong the guess was.

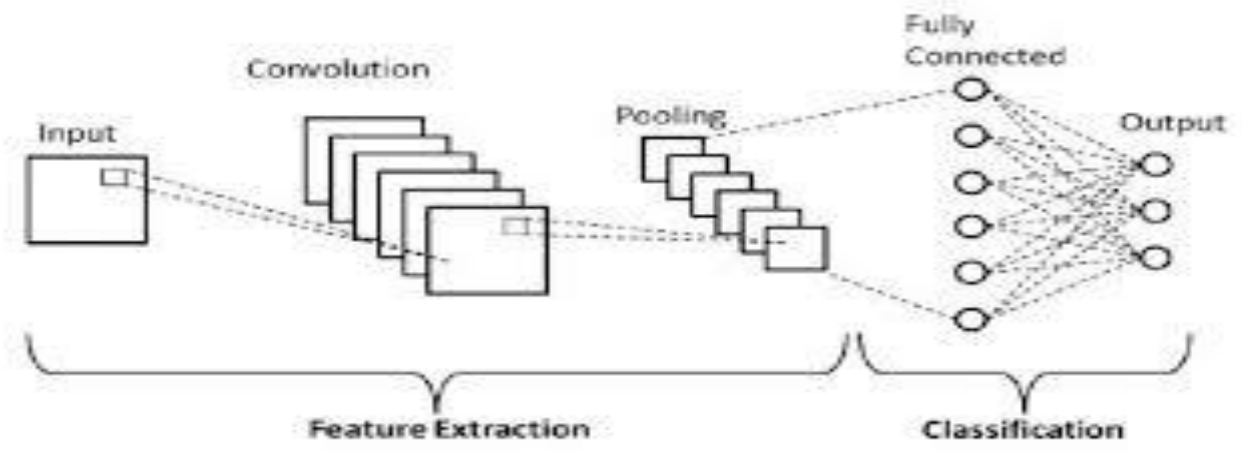
3. **Backpropagation:** The error signal is sent backward through the network layers, revealing how much each internal connection (weight) contributed to the total error.

4. **Weight Adjustment:** Using the error feedback, an optimizer (Gradient Descent) makes small, iterative adjustments to the weights and biases to ensure the next guess is more accurate.

Convolution Neural Network

WHAT IS IT?

A **Convolutional Neural Network (CNN)** is a type of deep learning network specifically designed to process data with a known grid-like structure, such as **images**. Its process is specialized in automatically extracting visual features.

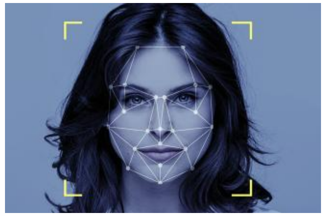


KEY CONCEPTS

- **Convolution:** Filter slides over input to extract features (edges, textures).
- **Filter / Kernel**: A small learnable weight matrix that detects one specific feature
- **Parameter Sharing**: A single filter's weights are reused across the entire image.
- **Pooling (Max):** Down samples feature map size, adding robustness to shifts.
- **Flatten**: Converts 3D features into a 1D vector for final classification

APPLICATIONS

- Facial Recognition
- Autonomous Vehicles
- Image Classification
- OCR / Handwriting recognition
- Product recommendation engine



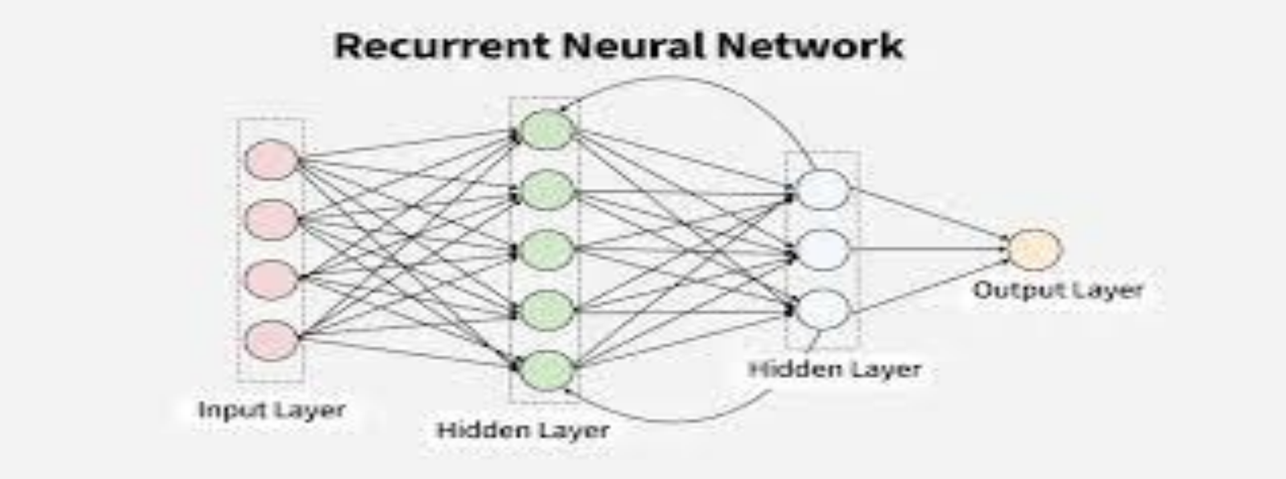
THE PROCESS

1. **Convolution:** Extracts features (edges, textures, patterns) by applying a filter (kernel) across the input.
2. **Activation (ReLU):** Introduces non-linearity to the extracted features, enabling the model to learn complex data maps.
3. **Pooling:** Reduces the spatial dimensions (width and height) of the feature data.
4. **Flattening:** Transforms the 2D/3D feature maps into a single 1D vector.
5. **Fully Connected:** Learns high-level global patterns from the flattened feature vector.
6. **Output (Softmax):** Calculates the final probability for each potential class label.

Recurrent Neural Network

WHAT IS IT?

A **Recurrent Neural Network (RNN)** is a type of deep learning model specifically designed to process **sequential data** or data where the order matters. Unlike a traditional neural network, an RNN has an internal memory that allows it to remember information from previous steps in the sequence.

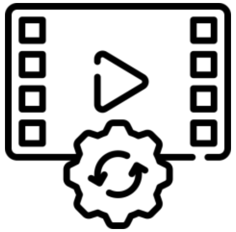


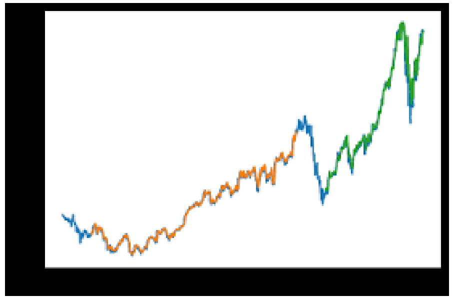
KEY CONCEPTS

- **Contextual Understanding:** Uses internal memory to process current data based on its entire history.
- **Variable Length Input:** Naturally handles sequences of any length (e.g., short or long sentences).
- **Sequence Mapping:** Great at mapping one sequence to another (like translation) or to a single label.
- **Long-Term Memory (LSTM):** Advanced versions use gates to remember critical info over extended periods.

APPLICATIONS

- Sentiment Analysis
- Video Processing
- Stock Prices Prediction
- Text Prediction





THE PROCESS

1. **Input Embedding:** Converts the sequential element (like a word) into a **dense numerical vector** for the network to understand.
2. **Initial Hidden State (Old Memory):** Sets the starting **internal memory** (context) to an empty state.
3. **Recurrence (Calculation):** Takes the **old memory** and the **current input** to compute the **new, updated memory**.
4. **Weight Sharing: Reuses the same set of weights** (rules) for every single time step in the sequence.
5. **Final Hidden State (Full Context):** The memory vector that holds the **complete understanding** of the entire sequence.
6. **Output (Prediction):** Uses the memory (from a specific step or the final one) to make a final **prediction** (e.g., translation, sentiment).

CNN Code Demo (python)

Training and Model Building Step

```python
# initializing the image generators each for training and validation set

#using 20% of train images for validation and rest 80% for training

train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary',
    subset='training',
    shuffle=True
)

validation_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary',
    subset='validation',
    shuffle=False
)
```

In the above code snippet, we are dividing the data set into 20% validation and rest 80% for training.

```python
# building the model here
#as per the approach 3 I am using 3 convolution layers and respective maxpooling layers

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)), #1st con layer
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'), #2nd conv layer
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'), #3rd conv layer
    MaxPooling2D(2,2),
    Flatten(),
    Dense(512, activation='relu'),
    Dense(1)
])

model.compile(
    optimizer=Adam(learning_rate=1e-5, clipnorm=1.0),
    loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=['accuracy']
)
```

In the above code snippet, we are building the model with 3 convolution layers.

Testing Step

```python
from tensorflow.keras.models import load_model
import numpy as np
from tensorflow.keras.preprocessing import image

#loading the model for image label testing
model = load_model('./cat_dog_model.h5')

def predict_image(model, img_path):
    # processing the image to make it ready to feed into the model for prediction
    img = image.load_img(img_path, target_size=(150, 150))
    img_array = image.img_to_array(img)
    img_array = img_array / 255.0  # normalize
    img_array = np.expand_dims(img_array, axis=0)  # add batch dimension

    prediction = model.predict(img_array)[0][0]  # get prediction value

    if prediction > 0.5:
        print(f"Prediction: dog ({prediction:.2f})")
    else:
        print(f"Prediction: Cat ({1 - prediction:.2f})")

imgPath = "./cat.14.jpg"  # input image 1
predict_image(model, imgPath);

imgPath = "./dog.5.jpg" # image image 2
predict_image(model, imgPath);
```

In the above code snippet, we are testing the trained model with images of cat and dog.

Input – Dog image

Output –

```
Prediction: dog (0.9101)
```

HW-Topic-5

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**

CWID – **20036779**

Large Language Models (LLMs)

Sudhanshu Kakkar | ID - 20036779

WHAT IS IT?

An LLM is a type of **deep learning model** designed to understand, generate, and interact with human language.

KEY CONCEPTS

- **Transformer Architecture:** The neural network design enabling parallel data processing.
- **Attention Mechanism:** A process that weighs the importance of input tokens to understand context.
- **Tokenization:** Breaking down raw text into smaller, readable units (tokens).
- **Context Window:** The maximum number of tokens the model can process or "remember" in one interaction.

Advantages

1. Increased Productivity in daily tasks
2. Natural Language Understanding
3. Code Generation
4. Highly Scalable

APPLICATIONS

- Code Generation
- Human Like Chatbots
- Content Creation
- Information Summarization

Limitations

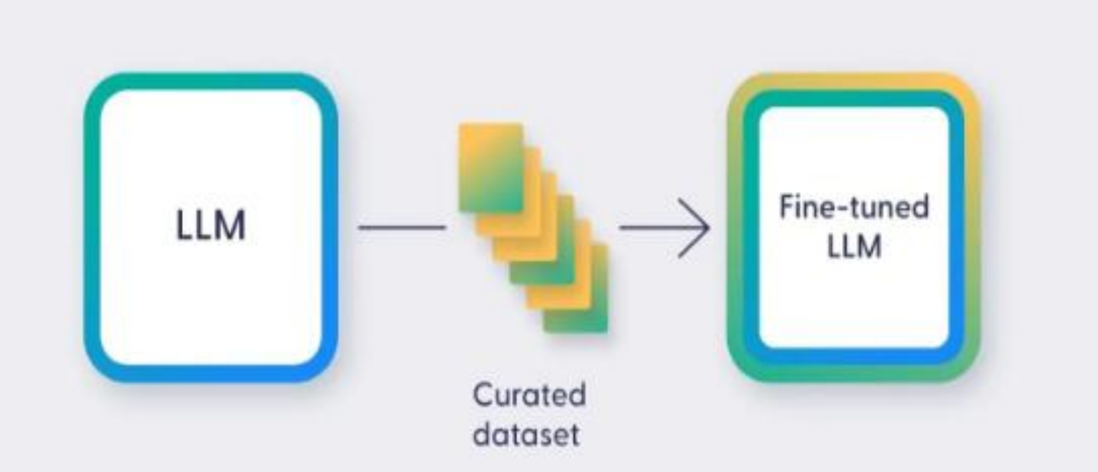
1. Hallucinations and inaccuracies
2. Limited or outdated knowledge
3. Expensive training
4. Lack of transparency

Fine-Tuning LLM

Sudhanshu Kakkar | ID - 20036779

WHAT IS IT?

Fine tuning a Large Language Model (LLM) is the process of taking a pre-trained general-purpose model and adapting it for a specific task or domain by continuing to train it on a smaller, targeted dataset



KEY CONCEPTS

- **Starting Point:** You begin with a pre-trained LLM, which has already learned extensive language and grammar, and a wide from huge amounts of public data.
- **Specialization:** You then introduce a **smaller, task specific dataset** (often a collection of labeled input/output examples).
- **Process:** The model continues its training process, adjusting its internal parameters (weights) based on the specialization dataset.

APPLICATIONS

- **Healthcare**: Summarize clinical notes, generate patient reports.
- **Finance**: Perform sentiment analysis on financial reports and news for market trend prediction
- **Legal**: Analyze, summarize, and extract key clauses from legal documents

Advantages

1. **Industry Specific Tasks**
2. **Productivity in daily tasks**
3. **Reduced Hallucinations**

Limitations

1. **Hard to Maintain with latest data**
2. **Finding optimum data for training becomes difficult some times.**

HW-Topic-6

Data Acquisition, Modeling and Analysis: Big Data Analytics

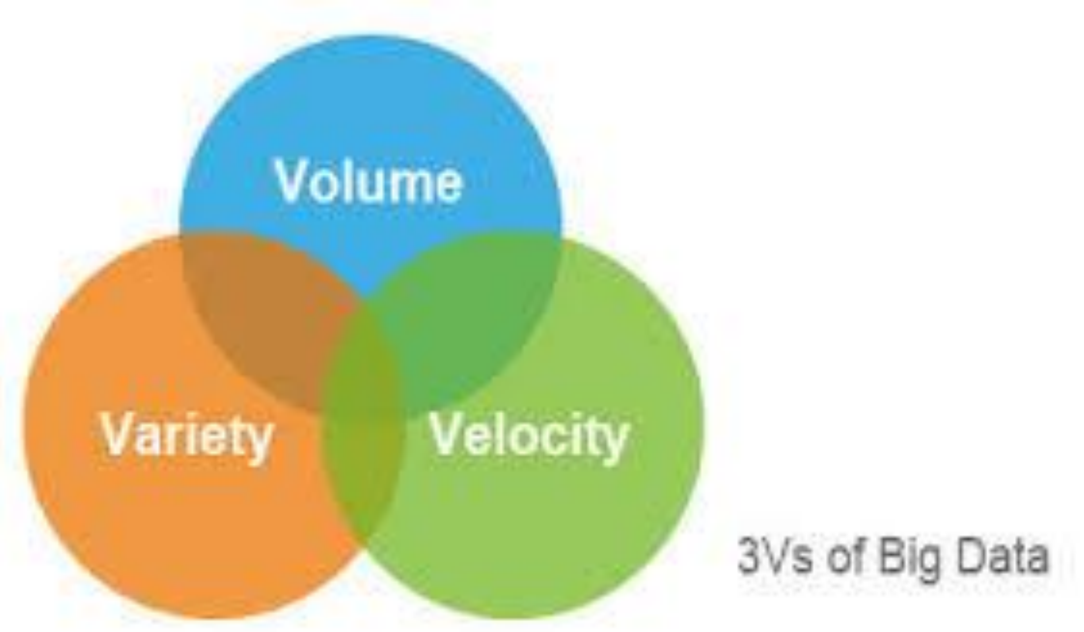
Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

Big Data Technologies

Sudhanshu Kakkar | ID - 20036779

WHAT IS IT?

Big Data Technologies are the special computer tools and systems that let us **collect**, **store**, and **understand** huge amounts of fast moving, diverse information.



THE THREE V'S - THE PROBLEM

- **Volume:** The enormous **scale** of data that is too massive for traditional databases to store and process.
- **Velocity:** The extreme **speed** at which data is generated, collected, and must be analyzed to provide real time value.
- **Variety:** The vast **diversity** of data types, including structured, semi structured, and complex unstructured formats like video and audio.

APPLICATIONS

- Personalized Recommendations
- Fraud Detection
- E-commerce Dynamic Pricing

Processing Frameworks

1. Apache Hadoop
2. Apache Spark
3. PySpark

Challenges

1. High Cost
2. Shortage of skilled professionals
3. Collecting huge data does not guarantee value

HW-Topic-7

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

Correlation

Sudhanshu Kakkar | ID - 20036779

WHAT IS IT?

- A statistical measure that describes the relationship between two or more variables.
- It lets us understand the relatedness between two variables, allowing for testing against models or simplification of analysis

TYPES

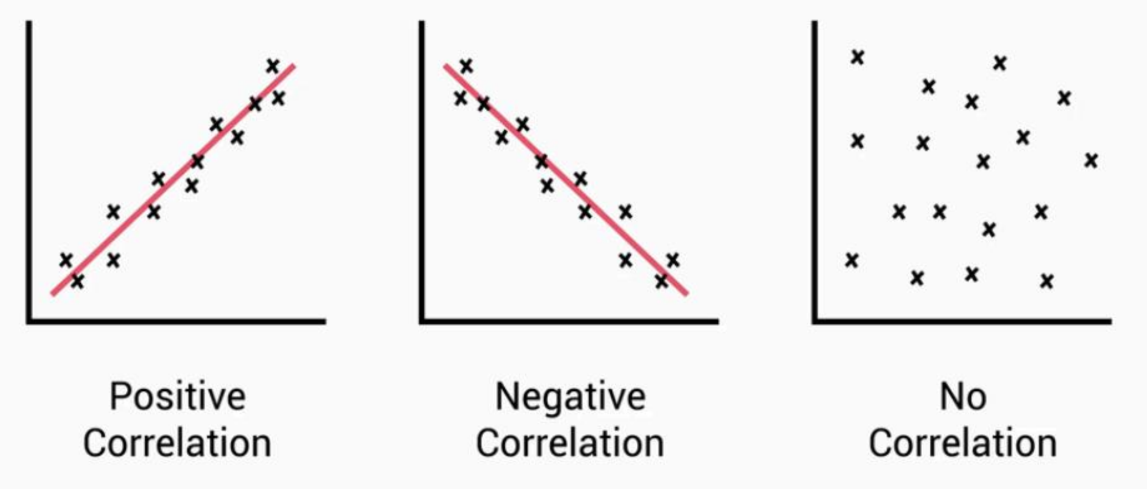
- **Positive correlation**:
  When one variable increases, so does the other.
- **Negative correlation**:
  When one variable increases, the other variable decreases.
- **No correlation**:
  When there is no linear relationship between variables.

APPLICATIONS

- Feature Selection in Data Science
- Dimensionality Reduction in case of correlated features

The Correlation Scale

- Range of -1 to 1
- 1 = strong positive correlation
- 0 to 1 = some positive correlation
- 0 = no correlation
- -1 to 0 = some negative correlation
- -1 = strong negative correlation



Sudhanshu Kakkar | ID - 20036779

Manual calculation of Correlation Coefficient

Dataset 1

$X = 10, 20, 30, 40, 50$

$Y = 5, 15, 25, 35, 45$

$$\mu_x = \frac{10+20+30+40+50}{5} = 30$$

$$\mu_y = \frac{5+15+25+35+45}{5} = 25$$

$$Var(X) = \frac{\Sigma(x_i - \mu_x)^2}{n-1} = \frac{(-20)^2+(-10)^2+(0)^2+10^2+20^2}{4} = 250$$

$$Var(Y) = \frac{\Sigma(y_i - \mu_y)^2}{n-1} = \frac{(-20)^2+(-10)^2+0^2+10^2+20^2}{4} = 250$$

$$Cov(X,Y) = \frac{\Sigma(x_i - \mu_x)(y_i - \mu_y)}{4} = \frac{(-20)^2+(-10)^2+0^2+10^2+20^2}{4} = 250$$

$$r = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{250}{\sqrt{250 \times 250}} = 1.0$$

Solution for Dataset 1

Dataset 2

$X = 20, 40, 60, 80, 100$

$Y = 10, 30, 50, 70, 90$

$$\mu_x = \frac{20+40+60+80+100}{5} = 60$$

$$\mu_y = \frac{10+30+50+70+90}{5} = 50$$

$$Var(X) = \frac{\Sigma(x_i - \mu_x)}{n-1} = \frac{(-40)^2+(-20)^2+0^2+20^2+40^2}{4} = \frac{4000}{4} = 1000$$

$$Var(Y) = \frac{\Sigma(y_i - \mu_y)}{n-1} = \frac{(-40)^2+(-20)^2+0^2+20^2+40^2}{4} = \frac{4000}{4} = 1000$$

$$Cov(X,Y) = \frac{\Sigma(x_i - \mu_x)(y_i - \mu_y)}{4} = \frac{(-40)^2+(-20)^2+0^2+20^2+40^2}{4} = \frac{4000}{4} = 1000$$

$$r = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{1000}{\sqrt{1000 \times 1000}} = 1.0$$

Solution for Dataset 2

Sudhanshu Kakkar | ID - 20036779

```python
import numpy as np

def CorrCoef(X,Y) :
    #  calculating mean of X and Y
    mux = np.mean (X)
    muy = np. mean (Y)

    # calculating variance for X and Y
    VarX = np. sum ( (X-mux)**2)/(len(X)-1)
    VarY = np. sum ( (Y-muy)**2)/(len(Y)-1)

    # calculating covariance of X and Y
    CoV = np. sum ( (X-mux)*(Y-muy))/(len(X)-1)

    # calculating correlating coeff here
    CoCo = CoV/np.sqrt(VarX*VarY)

    return CoCo

# Dataset 1
X = [10,20,30,40,50]
Y = [5,15,25,35,45]

C = CorrCoef(X,Y)
print(f"The Correlation Coefficient for dataset 1 is {C:.2f}")

# Dataset 2
x = [20,40,60,80,100]
Y = [10,30,50,70,90]
C = CorrCoef(X,Y)

print(f"The Correlation Coefficient for dataset 2 is {C:.2f}")
```

```
The Correlation Coefficient for dataset 1 is 1.00
The Correlation Coefficient for dataset 2 is 1.00
```

Program to calculate: Correlation Coefficient

HW-Topic-8

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**

CWID – **20036779**

Autoregressive (AR) Models

Sudhanshu Kakkar | ID - 20036779

WHAT IS IT?

- It's a way to **predict the future of a time series**
- The main idea is that **value depends only on its own past values.**
- The model learns the **relationship** a variable has with its history.

APPLICATIONS

- **Finance**: Stock or FX forecasting
- **Industry**: Predictive maintenance
- **Healthcare**: ECG/EEG analysis
- **Climate**: Weather prediction
- **Economics**: Trend modeling

ADVANTAGES

- Simplicity and Speed
- Effective for **Short-Term** Data
- Foundation for Forecasting

THE AR EQUATION

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \varepsilon_t$$

Term	Description
X_t	The value of the time series at the current time, t (the predicted value).
c	A constant term (intercept).
$\phi_1, \ldots, \phi_p$	The **autoregressive coefficients** (model parameters) that measure the influence of each past value.
$X_{t-1}, \ldots, X_{t-p}$	The past observations, known as **lagged values**.
ε_t	The **white noise** or random error term at time t. (Assumed to be independent and identically distributed, with a mean of zero and constant variance).

DISADVANTAGES

- Sensitive to model order misspecification.
- Requires stationarity
- Assumes linear relationships only.

HW-Topic-9

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

ARMA and ARIMA Models

Sudhanshu Kakkar | ID - 20036779

CORE IDEA

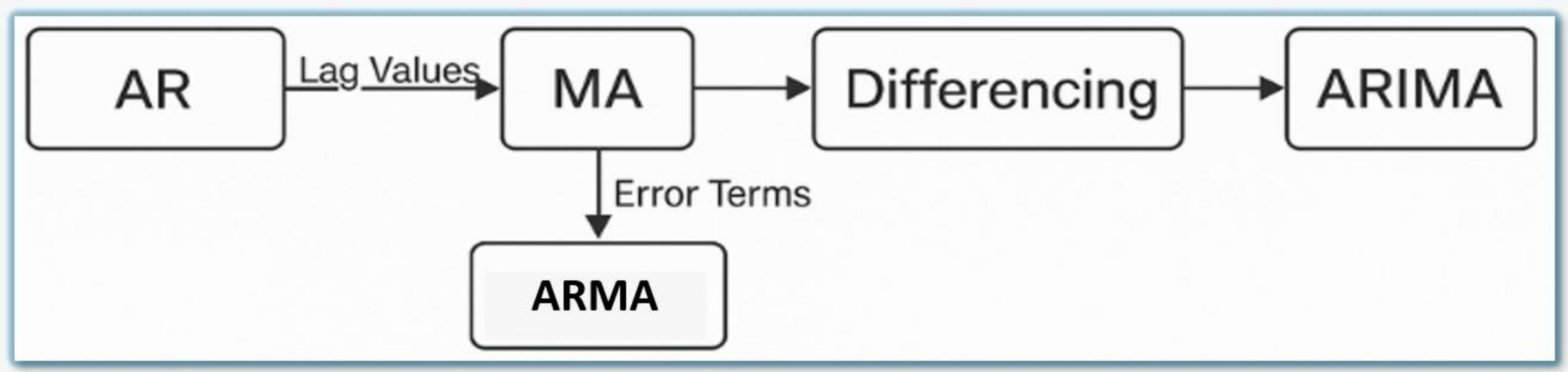
- ARMA and ARIMA aim to **model time-dependent data**
- **They use past values and past errors** to forecast future values.
- **ARMA:** Works for *stationary* time series.
- **ARIMA:** Adds *differencing* (I) to handle *non-stationary* data.

MATHEMATICAL FORM:

- **ARMA:** $X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$
- **ARIMA:** $(1-B)^d X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t$

APPLICATIONS

- Exchange rate prediction
- Stock price forecasting
- Air quality index prediction



Advantages

- Effective for **short-term forecasting**.
- Works well for **univariate time-series** data.

Limitations

- Assumes linear relationships between variables.
- Parameter tuning can be complex.

HW-Topic-10

Data Acquisition, Modeling and Analysis: Big Data Analytics

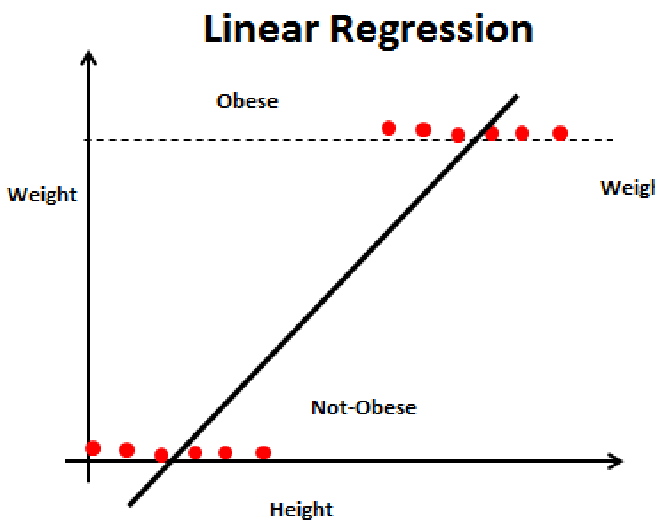
Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

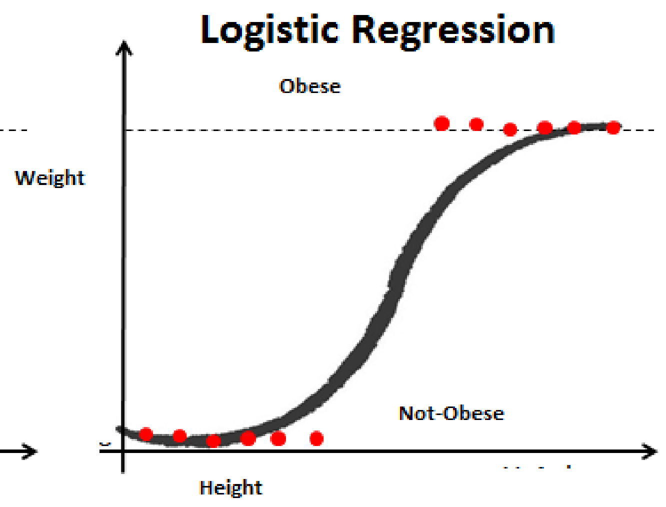
Linear Regression & Logistic Regression

Sudhanshu Kakkar | ID - 20036779

CORE IDEA

- **Linear:** predicts a continuous numeric value using a straight-line relationship between input features and output.
- **Logistic:** predicts a categorical outcome (like yes/no) by estimating probabilities using the logistic (sigmoid) function.





MATHEMATICAL FORM:

- **Linear:** $\hat{y} = \beta_0 + \beta_1 x$
- **Logistic:** $P(y=1|x) = \dfrac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$

APPLICATIONS

- **Linear**: House price, temperature, sales forecasting.
- **Logistic**: Disease detection, churn prediction, fraud detection.

Advantages

- **Linear:**
  - Simple & fast
  - Easy to interpret
  - Good for linear data
- **Logistic:**
  - Gives probabilities
  - Good for classification
  - Works well on small, separable data

HW-Topic-11

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

Sudhanshu Kakkar | ID - 20036779

Feature Selection

CORE IDEA

- Choosing the most important input features for a model.
- Helps the model focus on what matters.
- Removes noise and makes learning easier.

GOAL

- Reduce dimensionality.
- Improve model accuracy.

TYPES

Type	Speed	Accuracy	Model Used	Examples
Filter	Fast	Low-Medium	No	Correlation, Chi Square
Wrapper	Slow	High	Yes (many times)	Forward Selection
Embedded	Medium	Medium-High	Yes (during training)	Lasso

Advantages

- Faster training and prediction.
- Less overfitting.
- Cleaner and simpler models.
- Better accuracy when data has noise.

Disadvantages

- May remove useful information
- Hard to decide the right number of features
- Wrapper methods are slow
- Can be computationally heavy

HW-Topic-12

Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

Sudhanshu Kakkar | ID - 20036779

Embodied AI

CORE IDEA

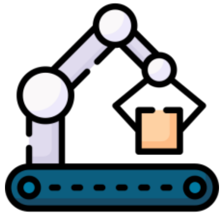
- AI models that **interact with the physical world** through sensors & actuators.
- Combines **perception, learning, planning, and control**.
- Learns from **real-world feedback**, not just data.
- Helps robots behave **more intelligently + context-aware**.

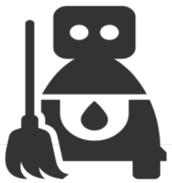
GOAL

- Enable robots to **navigate and act** like humans.
- Build systems that adapt to **unpredictable environments**.
- Improve **autonomy**, decision-making, and safety.

APPLICATIONS

- Self-driving cars
- Home & service robots
- Factory automation
- Drones & mapping





Advantages

- Robots learn better by interacting with the real world.
- Works well in messy, changing environments.
- More natural, human-like actions.
- Can improve safety and accuracy over time.

Disadvantages

- Needs a lot of training and good hardware.
- Hard to simulate real-world situations perfectly.
- Can make mistakes in unpredictable environments.
- Expensive to build and maintain.