CL4

BUISNESS INTELLIGENCE

ASSIGNMENT 1. Data Visualization from Extraction Transformation and Loading (ETL) Process

📊 1. What is the ETL Process?

ETL stands for:

Step	Description
Extract	Collect raw data from various sources (e.g., databases, APIs, CSVs)
Transform	Clean, filter, and format the data to make it usable
Load	Store the transformed data into a **data warehouse** or **dashboard**

This process is foundational in **data pipelines** — used in businesses, analytics, and machine learning.

📥 A. Extraction

Extraction means collecting data from **multiple and possibly messy sources**:

- Examples of sources:
    - Excel or CSV files
    - MySQL, PostgreSQL, Oracle databases
    - APIs (like weather, stock, or Twitter)
    - Cloud storage (AWS, Azure, Google Cloud)

Key Tasks:

- Connect to source
- Read structured or unstructured data
- Ensure minimal loss/corruption

🧹 B. Transformation

The most important step!

Here, raw data is **cleaned and shaped** into usable form:

🔧 Common Transformations:

- Handling missing/null values

- Converting formats (e.g., date strings → datetime)
- Filtering rows/columns
- Creating new columns (e.g., Total = Price × Quantity)
- Joining datasets
- Normalizing or scaling data
- Removing duplicates

This step improves **data quality** and makes it ready for analysis.

💾 C. Loading

This is where transformed data is **moved into a target system**:

- Common targets:
    - **Data warehouses** (e.g., Google BigQuery, Amazon Redshift)
    - **SQL databases**
    - **Excel or CSV for manual use**
    - **BI tools like Power BI, Tableau, or Excel Dashboards**
    - **Web dashboards (e.g., built with Streamlit or Flask)**

After loading, data is ready for **visualization**.

📈 3. Data Visualization after ETL

After data is extracted, cleaned, and loaded — we use **data visualization** tools to:

- Understand trends and patterns
- Communicate insights to stakeholders
- Monitor business KPIs

🧰 Tools for Visualization

- **Power BI**
- **Tableau**
- **Microsoft Excel (PivotCharts, Graphs)**
- **Python Libraries**:
    - matplotlib, seaborn, plotly, dash
- **Web dashboards**:

- Streamlit, Flask, ReactJS + D3

📊 Types of Visualizations

Type	When to Use
Bar Chart	Comparing values (e.g., Sales by Region)
Line Chart	Showing trends over time (e.g., Stock prices)
Pie Chart	Showing proportions (e.g., Market share)
Heatmap	Showing correlations or intensity
Scatter Plot	Relationship between 2 numeric variables
KPIs & Gauges	Displaying business indicators

✅ 5. Benefits of Doing Visualization from ETL Process

Benefit	Description
💡 Insight Generation	Understand what's happening in your business/data
📉 Error Detection	Easily spot outliers, missing trends
📊 Data-Driven Decisions	Guide actions based on actual evidence
🔁 Automation-Friendly	ETL can be scheduled and visualizations auto-updated

ASSIGNMENT 2: Perform the Extraction Transformation and Loading (ETL) process to construct the database in the Sql server / Power BI.

Same as ass1

ASSIGNMENT 3: Data Analysis and Visualization using Advanced Excel.

🔍 1. What is Data Analysis in Excel?

Data Analysis is the process of inspecting, cleaning, transforming, and modeling data to extract useful insights, answer questions, and support decision-making.

In Excel, this involves:

- Organizing data in a structured format (tables)
- Applying functions and formulas

- Using built-in analysis tools like PivotTables and Solver

📊 2. What is Data Visualization in Excel?

Data Visualization is the graphical representation of data using charts, graphs, and dashboards. It helps in:

- Understanding trends and patterns
- Identifying outliers
- Communicating insights visually

Excel provides many built-in tools and chart types for this.

🧠 3. Key Theoretical Concepts in Advanced Excel

✅ A. Excel Data Structures

- **Tables**: Structured range with headers and built-in sorting/filtering
- **Named Ranges**: Assign names to cell ranges for better readability
- **Dynamic Named Ranges**: Expand as new data is added

✅ B. Data Cleaning and Preparation

Before analysis, raw data must be cleaned. This includes:

- Removing duplicates
- Handling blanks or errors
- Standardizing data formats (e.g., dates, text cases)
- Using functions like:
    - TRIM() – removes extra spaces
    - CLEAN() – removes non-printable characters
    - ISERROR() – checks for errors

✅ C. Formulas and Functions for Analysis

1. Logical Functions

- IF, IFERROR, IFS, AND, OR, NOT
- Example: =IF(Sales>1000, "High", "Low")

2. Lookup Functions

- VLOOKUP, HLOOKUP, INDEX, MATCH, XLOOKUP

- Useful for searching and retrieving values

3. Statistical Functions

- AVERAGE, MEDIAN, MODE, STDEV, VAR
- COUNTIF, SUMIF, AVERAGEIF — for conditional aggregation

4. Text Functions

- LEFT(), RIGHT(), MID(), LEN(), CONCAT(), TEXTJOIN()
- Help clean and restructure text data

✅ D. PivotTables and PivotCharts

PivotTables are one of the most powerful Excel tools.

Theory:

- They allow dynamic summary of large data sets
- Support **grouping, filtering, slicing, aggregation**

Features:

- Drag-and-drop interface
- Can perform sums, averages, counts by category
- Used for trend analysis and KPI tracking

PivotCharts are visual representations of PivotTables.

✅ E. What-If Analysis

Used for scenario-based forecasting.

Tools:

1. **Goal Seek**: Finds input needed to reach a specific output
    - Example: What sales are needed to earn $10,000?
2. **Data Tables**: Show how changing one or two inputs affects output
3. **Scenario Manager**: Compare different sets of values

✅ F. Conditional Formatting

This allows highlighting data based on conditions.

Common formats:

- Color scales (green to red based on value)

- Data bars (mini bar chart in a cell)
- Icon sets (arrows, flags)

Example: Highlight sales below target in red.

✅ G. Advanced Charting Techniques

Common Charts:

- **Bar/Column Charts** – Compare categories
- **Line Charts** – Track trends over time
- **Pie Charts** – Show proportions
- **Scatter Plots** – Explore relationships between variables

Advanced Options:

- Combo Charts (e.g., line + column)
- Dynamic charts using named ranges and slicers
- Secondary axes for comparing different scales

✅ H. Data Validation

Used to control user input:

- Dropdown lists (e.g., for region selection)
- Restrict entries (e.g., only numbers between 1 and 100)

Helps maintain **data integrity**.

✅ I. Dashboards in Excel

A **dashboard** is a visual report composed of multiple charts, KPIs, and summaries on a single page.

Theory Behind Dashboards:

- Combine various outputs from data models
- Use slicers for interactivity
- Clean layout improves interpretation

ASSIGNMENT 5: Perform the data classification algorithm using any Classification algorithm

- What is Classification in Machine Learning?

Classification is a type of supervised learning where the output variable (target) is categorical — it belongs to a class or category

- **Types of Classification:**

Type	**Description**	**Example**
Binary Classification	Two possible outcomes	Spam or Not Spam
Multiclass Classification	More than two classes	Types of flowers: Setosa, Versicolor, Virginica
Multilabel Classification	Each sample can belong to multiple classes	Movie genres: Action + Comedy

ASSIGNMENT 6: Perform the data clustering algorithm using any Clustering algorithm

- What is Clustering?

Clustering is a machine learning method where we group similar data points into **clusters** based on some similarity. It's called **unsupervised** because it does not require labeled outputs.

- Goals of Clustering:

  High intra-cluster similarity: Items in the same cluster should be similar.

  Low inter-cluster similarity: Items in different clusters should be dissimilar.

- Types of Clustering Algorithms

Algorithm	Description	Use Case
K-Means	Divides data into k clusters based on mean distance	Customer Segmentation
Hierarchical	Creates a tree of clusters	Gene analysis

Algorithm	Description	Use Case
DBSCAN	Forms clusters based on density	Spatial data
GMM (Gaussian Mixture Model)	Probabilistic clustering	Soft clustering needs

- Theory of K-Means Clustering

Steps:

1. Select number of clusters (k)
2. Initialize k centroids randomly
3. Assign each point to the nearest centroid
4. Recalculate centroids of clusters
5. Repeat steps 3–4 until convergence (centroids stop changing)

🔢 Objective Function

K-Means tries to **minimize the Within-Cluster Sum of Squares (WCSS)**:

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- Ci: ith cluster
- μi: centroid of ith cluster
- x: a data point

📊 Choosing Optimal k – Elbow Method

Plot WCSS against different values of k. The point where WCSS starts to decrease slowly is called the "elbow" and gives an optimal k.

Code:

🔍 1. Imports and Dataset Loading

```
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
```

These libraries are essential for:

- **Pandas/Scikit-learn**: handling data and machine learning
- **Matplotlib/Seaborn**: plotting results

```
iris = datasets.load_iris()
X = iris.data
```

- Loads the **Iris dataset**, a classic dataset with 3 flower species and 4 features (sepal length/width, petal length/width).
- X stores the features for clustering.

⚙️ 2. Feature Scaling

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

- **Standardizes** the data (mean = 0, std = 1), which is **very important** for clustering since K-Means is distance-based and sensitive to scale.

🤖 3. K-Means Clustering

```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
```

- Initializes and runs **K-Means** to find **3 clusters** (we know there are 3 species).
- fit() assigns points to clusters by minimizing the **distance to centroids**.

```
labels = kmeans.labels_
centers = kmeans.cluster_centers_
```

- labels: cluster each point belongs to
- centers: coordinates of the cluster centroids

📊 4. Visualization

```
plt.figure(figsize=(8, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis', marker='o', edgecolor='k', s=100)
plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='X', s=200, label='Centroids')
plt.title('K-Means Clustering (Iris Dataset)')
plt.xlabel('Sepal Length (scaled)')
plt.ylabel('Sepal Width (scaled)')
plt.legend()
plt.show()
```

- Plots data points using **first two features** after scaling
- Colors show cluster membership
- Red **X** marks the **cluster centroids**
- Helps **visually evaluate** cluster formation

🧪 5. Evaluation: Adjusted Rand Index (ARI)

```
from sklearn.metrics import adjusted_rand_score
true_labels = iris.target
ari_score = adjusted_rand_score(true_labels, labels)
print(f"Adjusted Rand Index (ARI) score: {ari_score:.4f}")
```

- Compares predicted clusters to **true species labels**
- **ARI = 1** means perfect clustering, **0 = random**
- Shows how well unsupervised clustering matched real class labels

✅ **Importance Summary:**

Step	Purpose	Importance
StandardScaler	Normalizes features	Prevents bias in clustering
KMeans	Clusters data	Unsupervised grouping
Visualization	See clusters	Interpret results easily
ARI score	Measures clustering accuracy	Evaluate against ground truth

DEEP LEARNING

ASSIGNMENT 1. Problem Statement – Real estate agents want help to predict the house price for regions in the USA. He gave you the dataset to work on and you decided to use the Linear Regression Model. Create a model that will help him to estimate what the house would sell for. URL for a dataset: https://github.com/huzaifsayed/Linear-Regression-Model-for-House-Price Prediction/blob/master/USA_Housing.csv

Linear Regression is a **supervised learning** algorithm used for **predicting a continuous target variable** based on one or more input features.

Simple Linear Regression: One input variable.

Multiple Linear Regression: Multiple input variables.

The basic idea is to **fit a straight line (or hyperplane in higher dimensions)** that best represents the relationship between the input variables and the output.

The **linear equation** is:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Where:

- y^ = predicted value
- β0 = intercept
- βi = coefficients for each feature xi

The goal is to **minimize the error** between the actual price and the predicted price using **loss functions** like:

- **MSE (Mean Squared Error)**
- **RMSE (Root MSE)**
- **R^2 Score (Goodness of fit)**

1. MSE (Mean Squared Error)

📌 **What is it?**

MSE measures the **average squared difference** between the actual values and the predicted values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- yi = actual value
- y^i = predicted value
- n = number of observations

✅ **Why use it?**

- It **penalizes large errors** more than small ones (because the errors are squared).
- A **lower MSE** means better performance.

❌ **Limitation:**

- The result is in **squared units** (e.g., dollars²), which makes it harder to interpret.

2. RMSE (Root Mean Squared Error)

📌 **What is it?**

RMSE is just the **square root of MSE**.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

✅ **Why use it?**

- It brings the error **back to the same unit** as the target variable (e.g., dollars).
- More **interpretable** than MSE.

3. R^2 Score (R-squared or Coefficient of Determination)

📌 **What is it?**

R^2 measures how well the model explains the **variance** in the target variable.

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}}$$

Where:

- SSres = sum of squared residuals (errors)
- SStot = total sum of squares (variance in actual values)

✅ **Why use it?**

- **R^2 ranges from 0 to 1**:
  - **1** = perfect prediction (model explains all the variance)
  - **0** = model explains nothing
  - Can even be **negative** if the model performs worse than just predicting the mean!

Summary Comparison

Metric	Description	Range	Interpretation
MSE	Average of squared errors	≥0	Lower is better; hard to interpret directly
RMSE	Square root of MSE	≥0	Lower is better; same unit as target
R^2 Score	Proportion of variance explained	-∞ to 1	Closer to 1 is better

ASSIGNMENT 2. Build a Multiclass classifier using the CNN model. Use MNIST or any other suitable dataset.

a. Perform Data Pre-processing

b. Define Model and perform training

c. Evaluate Results using confusion matrix.

Goal:

Build a **multiclass classifier** using a **CNN** that can recognize handwritten digits (0 to 9) using the **MNIST dataset** (or any similar dataset).

📌 **What is MNIST?**

MNIST (Modified National Institute of Standards and Technology) is a benchmark dataset in machine learning consisting of:

- **60,000** training images
- **10,000** test images
- Each image is a **28x28 grayscale image** of a **digit between 0 and 9**

🧠 Theoretical Concepts

1. Multiclass Classification

Multiclass classification is a task where the model must classify an input into one of **more than two** possible categories. In MNIST:

- Class labels: 0, 1, 2, ..., 9 → total **10 classes**

---

2. CNN (Convolutional Neural Network)

CNNs are deep learning models particularly well-suited for image data because they can:

- Capture **spatial relationships** (nearby pixels matter)
- Use **filters/kernels** to extract **features** (edges, shapes, textures)

Key CNN Layers:

- Conv2D: Applies filters to the image to extract features
- MaxPooling2D: Downsamples the image to reduce dimensions and computation
- Flatten: Converts image matrix to a vector for input to fully connected layers
- Dense: Fully connected layers used for classification
- Softmax: Converts output into probabilities across all 10 classes

---

3. Confusion Matrix

A table used to evaluate classification performance. It shows:

- Correct predictions on the **diagonal**
- Misclassifications as **off-diagonal** entries

ASSIGNMENT 3:Design RNN or its variant including LSTM or GRU

a) Select a suitable time series dataset.

Example – predict sentiments based on product reviews

b) Apply for prediction

Theory Overview:

1. RNN (Recurrent Neural Network):

- Designed for **sequence data**.
- Maintains a **memory** (hidden state) of previous inputs.
- Good for short sequences, but suffers from **vanishing gradients** over long texts.

2. LSTM (Long Short-Term Memory):

- An advanced RNN that solves the vanishing gradient problem.
- Keeps long-term dependencies using **cell state** and **gates** (input, forget, output).
- Ideal for **sentiment analysis**, text classification, or language modeling.

3. GRU (Gated Recurrent Unit):

- A simpler version of LSTM with fewer gates (update and reset).
- Faster to train and often performs similarly to LSTM.

ASSIGNMENT 4:Design and implement a CNN for Image Classification a) Select a suitable image classification dataset (medical imaging, agricultural, etc.). b) Optimized with different hyper-parameters including learning rate, filter size, no. of layers, optimizers, dropouts, etc.

📌 **1. What is Image Classification?**

Image classification is the process of assigning a label (or class) to an image based on its visual content. For example, classifying X-ray images as *"normal"* or *"pneumonia"*, or classifying plant leaf images as *"healthy"* or *"diseased"*.

To do this automatically, we use deep learning models — particularly **Convolutional Neural Networks (CNNs)**.

🧠 **2. What is a CNN (Convolutional Neural Network)?**

A **CNN** is a type of deep learning model designed specifically for processing grid-like data (such as images). CNNs are great at extracting spatial features like edges, textures, and object parts from images.

🔁 **Key Layers in a CNN:**

- **Convolution Layer**: Applies filters to extract features like edges and shapes.
- **ReLU (Activation)**: Adds non-linearity so the model can learn complex patterns.
- **Pooling Layer (MaxPooling)**: Reduces dimensionality and helps retain important features.
- **Dropout Layer**: Randomly disables neurons during training to prevent overfitting.
- **Fully Connected (Dense) Layer**: Makes final predictions based on features.
- **Softmax/Output Layer**: Converts final scores into probabilities for classification.

3. Hyperparameter Optimization (Step b)

You can improve model accuracy and generalization by tuning **hyperparameters**:

Hyperparameter	Description	Typical Values
Learning Rate	Controls step size during optimization	0.1, 0.01, 0.001, 0.0001
Filter Size	Size of convolutional kernel	(3x3), (5x5)
Number of Layers	Depth of the model	2 to 5 convolution blocks
Optimizers	Optimization algorithms	SGD, Adam, RMSprop
Dropout Rate	% of neurons deactivated during training	0.2 to 0.5
Batch Size	Number of samples per training update	16, 32, 64
Epochs	Number of passes through the dataset	10 to 100+

📌 **Common Methods for Optimization:**

- **Manual tuning**: Try different values and observe performance
- **Grid Search / Random Search**: Automated systematic testing
- **Keras Tuner**: A library for tuning Keras models

4. Model Evaluation Metrics

To assess performance, you use:

- **Accuracy**: % of correct predictions
- **Loss**: Difference between prediction and actual label
- **Confusion Matrix**: Shows true vs predicted classes
- **Precision, Recall, F1-score**: Useful in imbalanced datasets

5.Visualization

- **Training Curves**: Plot loss and accuracy over epochs
- **Confusion Matrix**: Heatmap of prediction performance
- **Grad-CAM**: Visualize which part of image influenced the decision

ASSIGNMENT 5. Perform Sentiment Analysis in the network graph using RNN.

📌 **1. What is Sentiment Analysis?**

Sentiment Analysis (also called Opinion Mining) is the process of determining whether a piece of text (like a tweet, product review, or comment) expresses a **positive**, **negative**, or **neutral** sentiment.

It's widely used in:

- Social media monitoring
- Customer feedback analysis
- Political opinion mining
- Market research

🔁 **2. Why Use RNN for Sentiment Analysis?**

RNNs (Recurrent Neural Networks) are a class of neural networks designed for **sequential data** like text, speech, or time series.

In sentiment analysis, text data has a **natural order**, and meaning depends on word sequence (e.g., *"not good" ≠ "good not"*). RNNs can:

- **Remember context** from previous words
- **Predict next steps** in a sequence
- **Capture sentence structure**

🧠 3. What is a Network Graph in this Context?

In the context of this task, **Network Graph** typically refers to:

- A **graph of relationships** between words (word co-occurrence networks)
- A **social network** where nodes represent users and edges represent relationships
- Or a **neural network graph** (architecture of RNNs)

Here, we assume the **goal is to perform sentiment analysis on text data represented as nodes and edges** — such as:

- Tweets linked by hashtags or retweets
- Comments linked by common topics
- Nodes with text content; edges indicate interaction

So we analyze **text at each node** using RNN to detect sentiment, and can even propagate or visualize **sentiment across the network graph**.

⚙️ 4. Core Components of Sentiment Analysis using RNN

📄 Step 1: Preprocessing Text

- **Tokenization**: Split text into words/tokens
- **Lowercasing**: Convert all to lowercase
- **Stopword removal**: Remove common unimportant words (e.g., “the”, “is”)
- **Stemming/Lemmatization**: Reduce words to root form (e.g., “running” → “run”)
- **Padding**: Make all sequences the same length

🧠 Step 2: Word Embedding

Words are converted to dense vectors using:

- **Word2Vec**
- **GloVe**
- **TF-IDF**
- **Embedding layer (trainable)**

These embeddings capture semantic relationships (e.g., *happy* is close to *joyful*)

🧪 Step 4: Train the Model

- Input: Preprocessed sequences of text

- Output: Labels (0 = negative, 1 = positive)
- Loss Function: binary_crossentropy (for 2 classes)
- Optimizer: adam (commonly used)

📊 **Step 5: Evaluate**

- Accuracy
- Confusion Matrix
- Precision, Recall, F1 Score

🔁 **5. Advanced: Sentiment in Network Graphs**

If text nodes are part of a **social or semantic network**, you can:

- Use **Graph Neural Networks (GNNs)** to incorporate network structure
- Or apply **RNN individually to each node's text**, then use network metrics (like centrality) to analyze global sentiment flow

Example:

- Tweets are nodes
- Retweet/mention links are edges
- Sentiment is calculated per tweet → can visualize **sentiment clusters**

📌 **6. Why RNN over Other Models?**

Model	Strength
RNN	Handles sequence data well, good for short texts
LSTM/GRU	Better at long-term dependencies (e.g., long sentences)
CNN	Fast and good at local patterns, but weaker on long dependencies
Transformer	More powerful than RNN for big data, but heavier and complex