Phase-3 Submission Document

Student Name: [B.Gayathri]

Register Number: [620123104022]

Institution: [AVS Engineering college ]

Department: [B.E. Computer science and engineering ]

Date of Submission: [Insert Date]

# 1. Problem Statement

* Describe the real-world scenario that demands a solution.
* Highlight its societal or business importance.
* Define whether the problem is Classification, Regression, Clustering, etc.
* Explain how solving this will provide measurable value.

# 2. Abstract

* Briefly describe the problem domain.
* State your project's core objective.
* Outline your methodology and tools used.
* Summarize the expected or obtained outcome.
* Mention how this benefits the stakeholders.

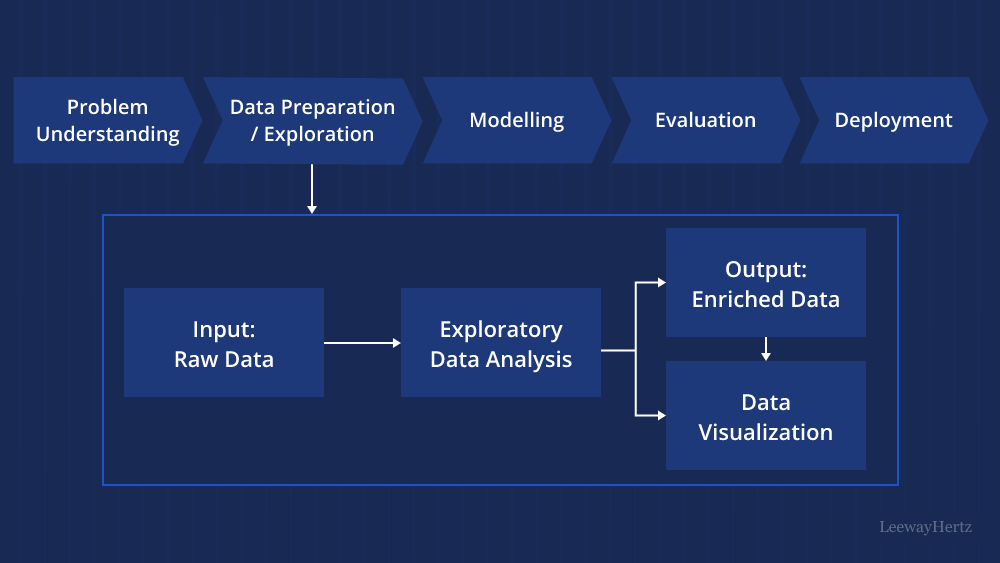
# 3. System Requirements

* Hardware Requirements:
* - Minimum 8GB RAM
* - i5 or higher processor
* Software Requirements:
* - Python 3.8+
* - Jupyter Notebook or Google Colab
* - Libraries: pandas, numpy, matplotlib, seaborn, sklearn, etc.

# 4. Objectives

* Clearly list measurable goals of the project.
* Specify what predictions, insights, or decisions the system will support.
* Align these objectives with the business or academic value.

# 5. Flowchart of Project Workflow

* Stages involved:
* - Data Collection
* - Data Preprocessing
* - Exploratory Data Analysis (EDA)
* - Feature Engineering
* - Model Building
* - Model Evaluation
* - Deployment
* 

# 6. Dataset Description

* Mention data source (e.g., Kaggle, UCI).
* Specify type (Public/Private/Synthetic).
* Indicate size (e.g., 10,000 rows, 20 columns).
* Describe columns and data types.
* Include a sample screenshot using df.head().

# 7. Data Preprocessing

* Steps:
* - Handling missing values using mean/median or dropping.
* - Removing duplicate rows.
* - Outlier detection with IQR or Z-score.
* - Encoding categorical variables.
* - Feature scaling using StandardScaler/MinMaxScaler.
* Include before/after screenshots of dataset.

# 8. Exploratory Data Analysis (EDA)

* Use visualization tools:
* - Histograms, Boxplots, Pairplots
* - Correlation Heatmaps
* Analyze distributions, outliers, and trends.
* Summarize key insights found.

# 9. Feature Engineering

* Create new relevant features (e.g., ratios, time-based).
* Use domain knowledge for feature transformation.
* Perform feature selection via importance or correlation.
* Explain rationale behind selected features.

# 10. Model Building

* Train multiple models:
* - Logistic Regression, Decision Trees, Random Forest, etc.
* Use GridSearchCV or RandomizedSearchCV for tuning.
* Justify choice based on problem type and performance.

# 11. Model Evaluation

* Use metrics:
* - Classification: Accuracy, F1-Score, Confusion Matrix, ROC-AUC
* - Regression: RMSE, MAE, R² Score
* Include visualizations like ROC curve and confusion matrix.
* Compare models and choose the best.

# 12. Deployment

* Choose platform: Streamlit, Gradio, Flask API.
* Create user interface for input/output.
* Host on Render, HuggingFace, or Deta.
* Share live link and sample UI screenshot.

# 13. Source Code

import numpy as np

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from keras.utils import to\_categorical

# 1. Load the MNIST dataset

(X\_train, y\_train), (X\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

# 2. Preprocess the data

X\_train = X\_train.astype('float32') / 255.0

X\_test = X\_test.astype('float32') / 255.0

y\_train = to\_categorical(y\_train, num\_classes=10)

y\_test = to\_categorical(y\_test, num\_classes=10)

X\_train = X\_train.reshape(-1, 28, 28, 1) # Add a channel dimension

X\_test = X\_test.reshape(-1, 28, 28, 1) # Add a channel dimension

# 3. Build a CNN model

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.2)) # Add dropout for regularization

model.add(Dense(10, activation='softmax'))

# 4. Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# 5. Train the model

model.fit(X\_train, y\_train, epochs=5, batch\_size=32, validation\_data=(X\_test, y\_test))

# 6. Evaluate the model

loss, accuracy = model.evaluate(X\_test, y\_test)

print('Test loss:', loss)

print('Test accuracy:', accuracy)

# 7. Make predictions

predictions = model.predict(X\_test)

predicted\_labels = np.argmax(predictions, axis=1)

# Display predictions (optional)

import matplotlib.pyplot as plt

plt.imshow(X\_test[0].reshape(28, 28), cmap='gray')

plt.title(f"Predicted: {predicted\_labels[0]}")

plt.axis('off')

plt.show()

# 14. Future Scope

* Enhance model with more data or external APIs.
* Enable real-time prediction or dashboard analytics.
* Improve accuracy by ensemble methods or deep learning.
* Add advanced deployment with Docker or CI/CD.

# 15. Team Members and Roles

* GiwriPriya :Data Preprocessing
* Keerthana EDA
* Ashitha : Model Building, Evaluation
* Gayathri:Deployment, Documentation