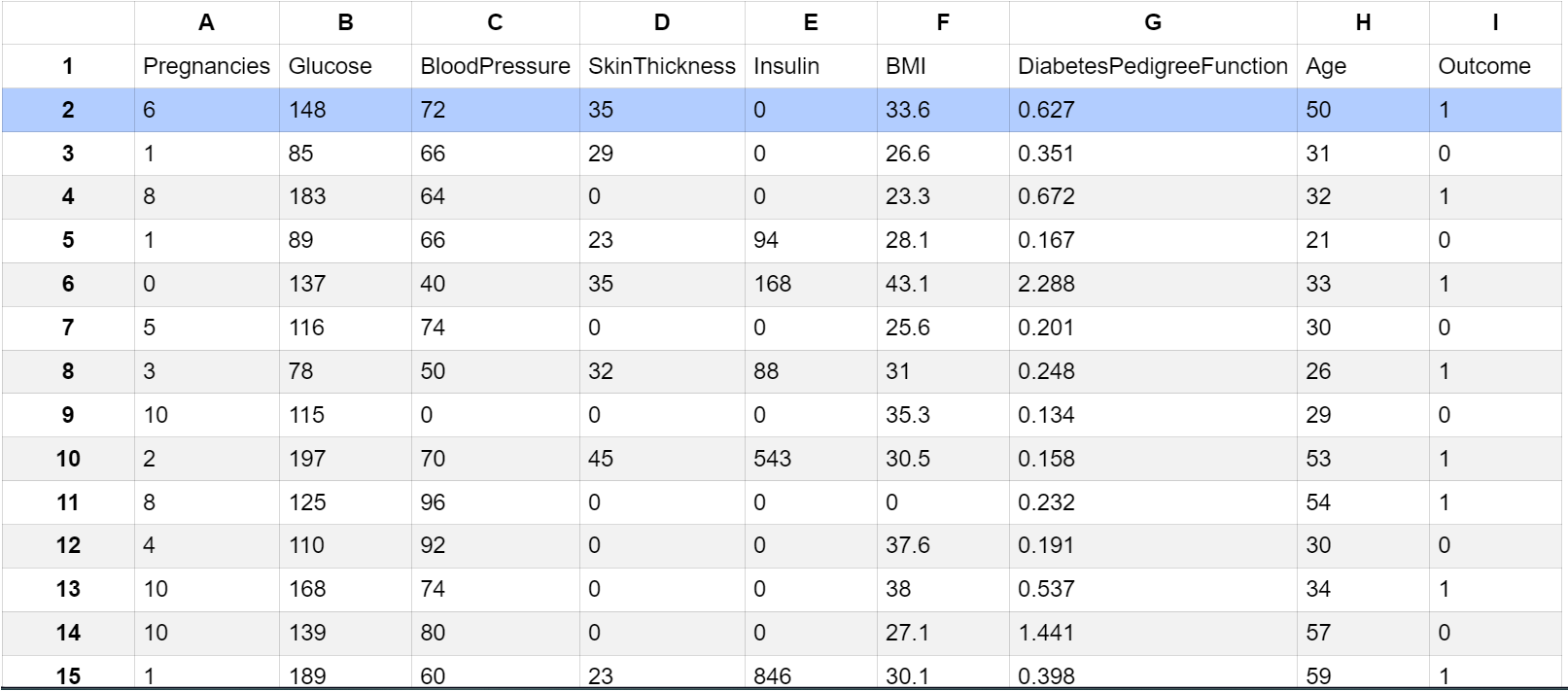
DIABETES PREDICTION SYSTEM USING AI & ML

PHASE 3 SUBMISSION

Diabetes Prediction

Given dataset:



1.Loading the dataset:

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.

a.Identify the dataset:

The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

b.Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c.Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

Loading dataset

dataset = pd.read\_csv('diabetes.csv')

Some common data preprocessing tasks include:

* Data cleaning: This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.
* Data transformation: This involves converting the data into a format that is suitable for the analysis task. For example, this may involve converting categorical data to numerical data, or scaling the data to a suitable range.
* Feature engineering: This involves creating new features from the existing data. For example, this may involve creating features.
* represent interactions between variables, or features that represent summary statistics of the data.
* Program:

### 1. Descriptive Statistics:

import pandas as pd

2.Load the Dataset:

Load your dataset into a Pandas DataFrame. You can typically find house price datasets in CSV format, but you can adapt this code to other formats as needed.

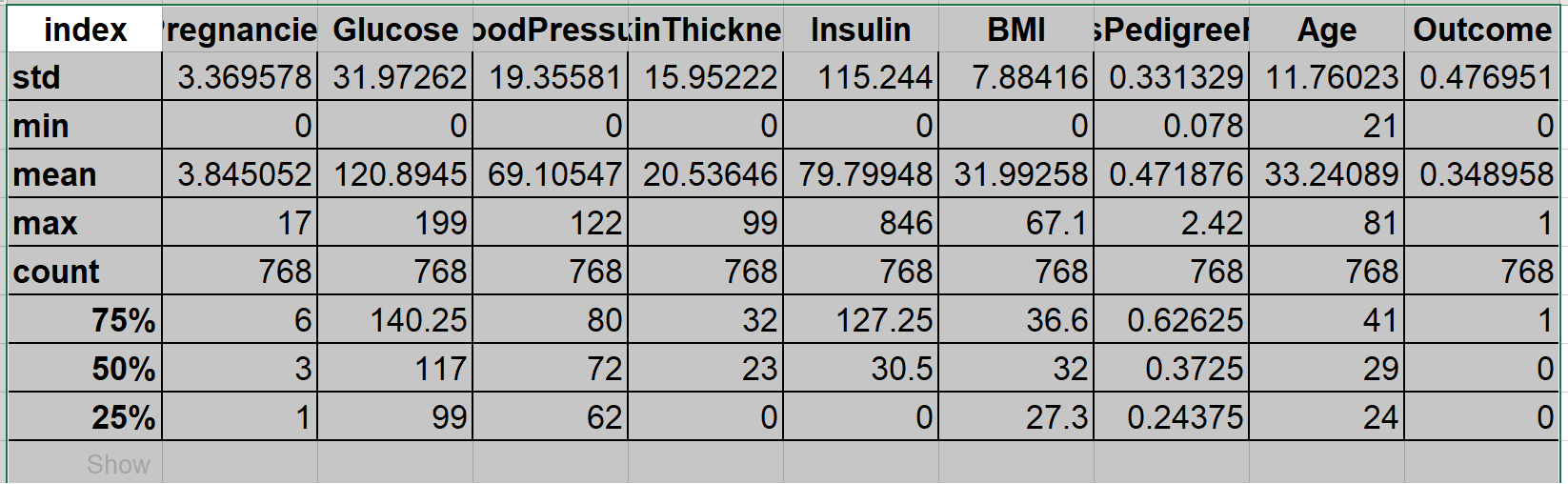
Program:

# Load the dataset (assuming it's in a CSV file)

data = pd.read\_csv(diabetes.csv')

# Display basic statistics

data.describe()



### 2. Data Visualization:

import matplotlib.pyplot as plt

import seaborn as sns

# Histograms for all features

data.hist(figsize=(12, 10))

plt.tight\_layout()

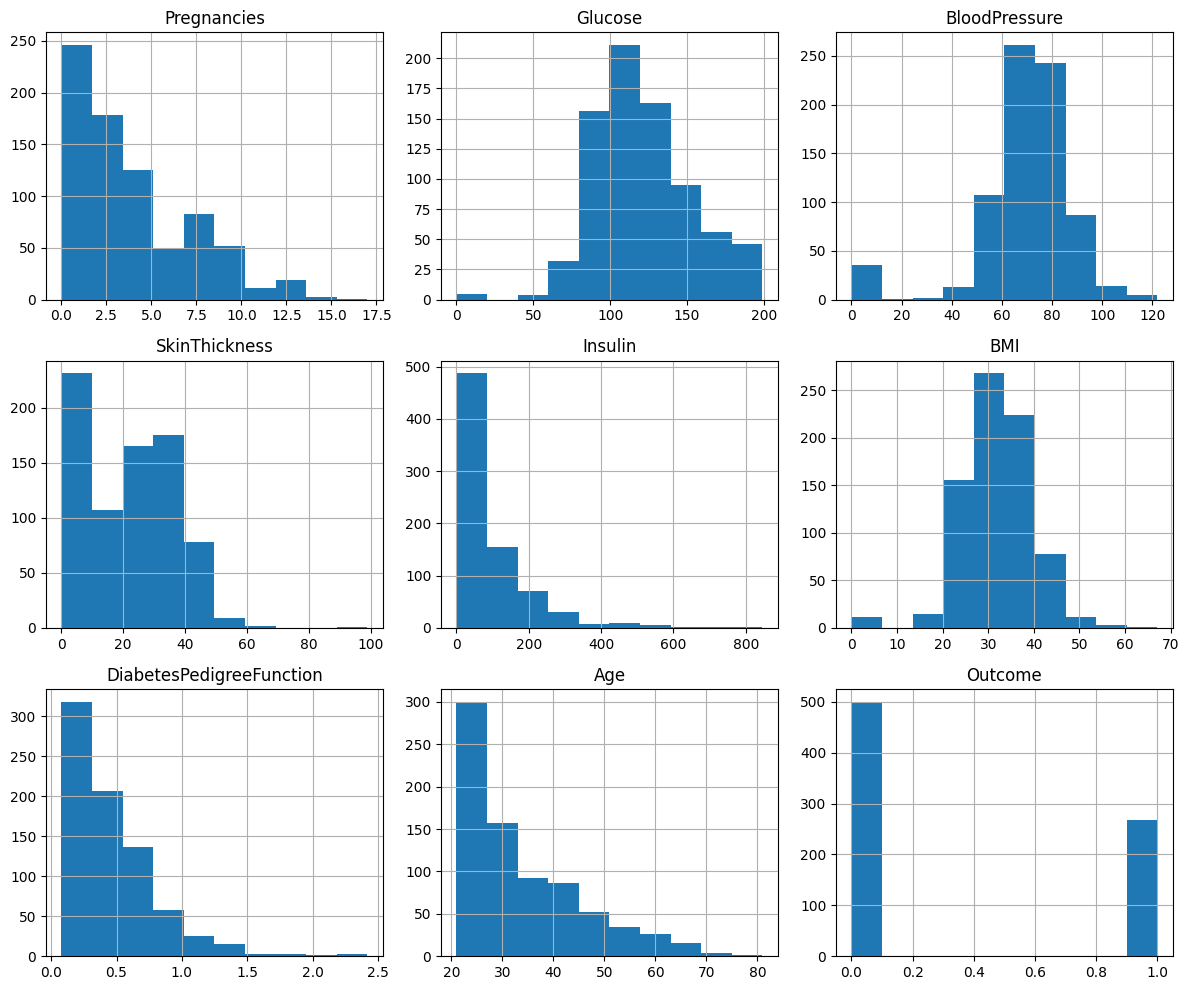
plt.show()

# Box plot for specific features (e.g., Glucose and Age)

sns.boxplot(x='Outcome', y='Glucose', data=data)

plt.show()

sns.boxplot(x='Outcome', y='Age', data=data)

plt.show() 

### 3. Correlation Analysis:

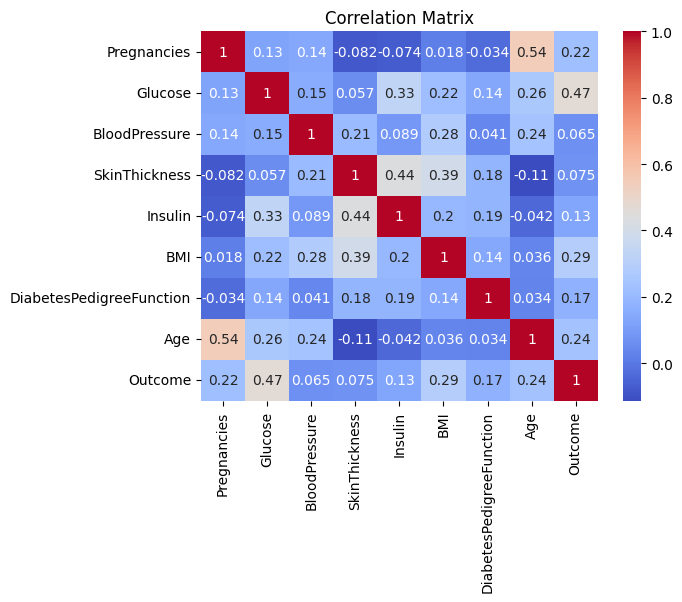
# Correlation matrix

correlation\_matrix = data.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()



### 4. Feature Engineering (Example: BMI Categories):

# Create a new feature 'BMI\_Category'

def categorize\_bmi(bmi):

if bmi < 18.5:

return 'Underweight'

elif 18.5 <= bmi < 25:

return 'Normal'

elif 25 <= bmi < 30:

return 'Overweight'

else:

return 'Obese'

data['BMI\_Category'] = data['BMI'].apply(categorize\_bmi)

### 5. Model Building (Example: Logistic Regression):

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Assuming 'X' contains features and 'y' contains the target variable 'Outcome'

X = data[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]

y = data['Outcome']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train a Logistic Regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Predict using the model

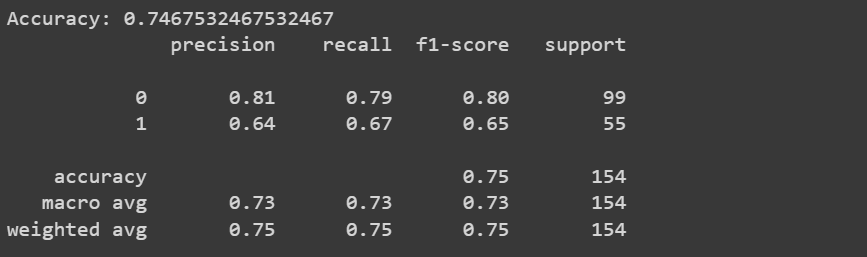
y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

print(classification\_report(y\_test, y\_pred))



### 6. Hyperparameter Tuning:

# Example: Grid Search for Logistic Regression

from sklearn.model\_selection import GridSearchCV

param\_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}

grid\_search = GridSearchCV(LogisticRegression(), param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

best\_model = grid\_search.best\_estimator\_

# Use the best model for predictions

y\_pred = best\_model.predict(X\_test)