PHASE-3

Project Title**: AI-BASED DIABETES PREDICTION SYSTEM**

Phase 3:Data visualization

ARTIFICIAL INTELLIGENCE

INTRODUCTION:

Data virtualization in artificial intelligence (AI) typically involves the use of techniques and tools to create a unified, abstracted view of data from various sources, making it easier for AI systems to access, analyze, and process data.

We will select relevant features that can impact diabetes risk prediction.

Data Virtualization Algorithm:

1. Gather Real Data
2. Data Analysis
3. Choose Data Virtualization Technique
4. Train the Virtualization Model
5. Generate Synthetic Data

DATASET:

Dataset Link: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

Algorithm: Data Preprocessing, Classification, and visualisation.

1. Import the necessary libraries (NumPy, pandas, scikit-learn, and Matplotlib).
2. Define a list of relevant features (e.g., 'Glucose', 'BloodPressure', 'BMI', 'Age').
3. Load the diabetes dataset from a CSV file, and subset it to include relevant features and the 'Outcome' target variable.
4. Standardize the feature data using StandardScaler.
5. Split the data into training and testing sets.
6. Create a Support Vector Machine (SVM) classifier with a linear kernel and train it on the training data.
7. Make predictions on both the training and testing sets and calculate accuracy scores.
8. Visualize accuracy using a bar chart.
9. Create a histogram to visualize the distribution of the "Glucose" feature.
10. Make predictions on a sample input data point (e.g., a hypothetical patient's data).

11. Generate synthetic data:

    a. Determine the number of synthetic samples to generate.

    b. Initialize lists to store synthetic data points and labels.

    c. For each synthetic sample, randomly select an index from the real data.

    d. Select a real data point and its label.

    e. Create a slightly modified version of the real data point with random noise. f. Append the modified data point and its label to the synthetic data.

12. Combine real and synthetic data.

13. Print the first 5 samples of the synthetic data and their corresponding labels.

PROGRAM:

```
# Import necessary libraries

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn.metrics import accuracy_score

import matplotlib.pyplot as plt

# Define a list of relevant features

relevant_features = ['Glucose', 'BloodPressure', 'BMI', 'Age']

# Load the diabetes dataset from a CSV file (Replace 'diabetes.csv' with the actual dataset path)

diabetes_dataset = pd.read_csv('diabetes.csv')

# Subset the dataset to include relevant features and the 'Outcome' target variable

diabetes_dataset = diabetes_dataset[relevant_features + ['Outcome']]

# Split the data into features (X) and the target (Y)

X = diabetes_dataset[relevant_features]

Y = diabetes_dataset['Outcome']

# Standardize the feature data using StandardScaler

scaler = StandardScaler()

X = scaler.fit_transform(X)
```

```
# Split the data into training and testing sets

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)

# Create an SVM classifier with a linear kernel

classifier = svm.SVC(kernel='linear')

classifier.fit(X_train, Y_train)

# Make predictions on the training and testing sets

Y_train_pred = classifier.predict(X_train)

Y_test_pred = classifier.predict(X_test)

# Calculate accuracy scores for training and testing data

training_data_accuracy = accuracy_score(Y_train_pred, Y_train)

test_data_accuracy = accuracy_score(Y_test_pred, Y_test)

# Visualize accuracy using a bar chart

accuracies = [training_data_accuracy, test_data_accuracy]

labels = ['Training Data', 'Test Data']

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

plt.bar(labels, accuracies, width=0.4, align='center', alpha=0.5, color=['blue', 'green'])

plt.xlabel('Data Split')

plt.ylabel('Accuracy')

plt.title('Training and Test Accuracies')

# Data visualization section: Histogram of the "Glucose" feature

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

plt.hist(X[:, 0], bins=20, color='blue', alpha=0.7)

plt.xlabel('Glucose Level')

plt.ylabel 'Frequency')

plt.title('Distribution of Glucose Levels')

# Make predictions on a sample input data point

input_data = np.array([1, 85, 66, 29, 0, 26.6, 0.351, 31]).reshape(1, -1)

input_data = scaler.transform(input_data)

prediction = classifier.predict(input_data)
```

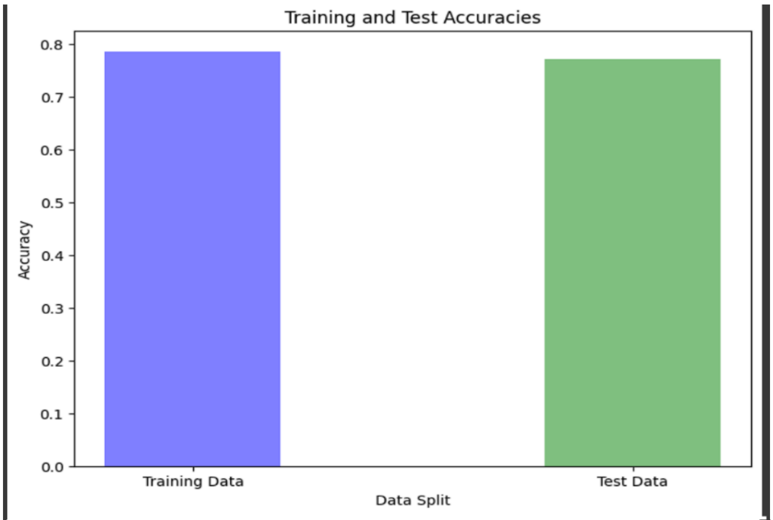
```
# Number of synthetic samples to generate
num_samples_to generate = 500
# Initialize lists to store synthetic data
synthetic_data = []
synthetic_labels = []
# Generate synthetic data points based on existing data
for _ in range(num_samples_to_generate):
    random_index = np.random.randint(0, len(X))
    real_data_point = X[random_index]
    real_label = Y[random_index]
    modified_data_point = real_data_point + np.random.normal(0, 0.1, size=real_data_point.shape)
    synthetic_data.append(modified_data_point)
    synthetic_labels.append(real_label)
# Combine real and synthetic data
X_synthetic = np.vstack([X, np.array(synthetic_data)])
Y_synthetic = np.concatenate([Y, np.array(synthetic_labels)])
# Print the first 5 samples of the synthetic data
print("Synthetic Data (X_synthetic):")
print(X_synthetic[:5])
# Print the corresponding labels for the first 5 samples
print("Synthetic Labels (Y_synthetic):")
print(Y_synthetic[:5])
```

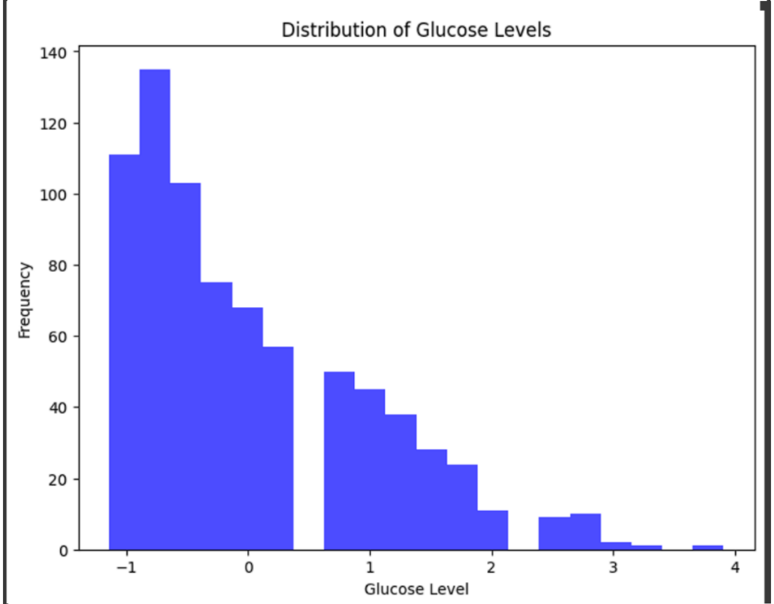
OUTPUT:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
(768, 8) (614, 8) (154, 8)
Accuracy score of the training data :  0.7866449511400652
```






```
Accuracy score of the test data :  0.7727272727272727
[[1.00000000e+00 8.50000000e+01 6.60000000e+01 2.90000000e+01
  3.00685403e-17 2.66000000e+01 3.51000000e-01 3.10000000e+01]]
[1]
The person is diabetic
Synthetic Data (X_synthetic):
[[ 0.63994726  0.84832379  0.14964075  0.90726993 -0.69289057  0.20401277
   0.46849198  1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575  0.53090156 -0.69289057 -0.68442195
  -0.36506078 -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 -1.28821221 -0.69289057 -1.10325546
   0.60439732 -0.10558415]
 [-0.84488505 -0.99820778 -0.16054575  0.15453319  0.12330164 -0.49404308
  -0.92076261 -1.04154944]
 [-1.14185152  0.5040552  -1.50468724  0.90726993  0.76583594  1.4097456
   5.4849091  -0.0204964 ]]
Synthetic Labels (Y_synthetic):
[1 0 1 0 1]
```

Result:

The code preprocesses a diabetes dataset, trains a support vector machine (SVM) classifier, visualizes the distribution of glucose levels(visualization), and augments the dataset with synthetic data for classification and analysis. Specific results depend on the dataset and problem domain, including classifier accuracy and improved data quality for enhanced model performance.