PHASE 5:DOCUMENTATION

Creating an AI-based diabetes prediction system involves several key steps, from defining the problem to developing a model and evaluating its performance. Here is a high-level overview of the process:

1. Problem Statement:

Define the problem you want to address. In this case, the problem is diabetes prediction. The goal is to develop a machine learning model that can predict whether a person is likely to have diabetes based on certain features.

2. Design Thinking Process:

Design thinking involves understanding the needs and perspectives of the end-users and stakeholders. It helps in creating a solution that addresses real-world issues effectively. Engage with healthcare professionals, patients, and other stakeholders to gain insights into the problem and its context.

3. Phases of Development:

a. Data Collection: Gather relevant data for the problem. This may include medical records, lab results, lifestyle data, and more.
b. Data Preprocessing: Clean, preprocess, and prepare the data for model training. This may involve handling missing values, normalizing or scaling features, and encoding categorical variables.
c. Feature Selection: Choose the most relevant features from the dataset to improve model performance and reduce dimensionality.
d. Model Selection: Select an appropriate machine learning algorithm for the task. Common choices include logistic regression, decision trees, random forests, support vector machines, or neural networks.
e. Model Training: Train the selected machine learning model using a portion of the dataset. Split the data into training and validation sets.

f. Hyperparameter Tuning: Optimize the model's hyperparameters to improve its performance.
g. Model Evaluation: Evaluate the model's performance using appropriate metrics and cross-validation techniques.
h. Model Interpretation: Understand the model's predictions and interpretability, especially in healthcare applications.
i. Deployment: Deploy the model in a healthcare setting, ensuring data privacy and security compliance.

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

I have used dataset which is mentioned above.

5. Data Preprocessing Steps:

a. Data Cleaning: Handle missing values, outliers, and errors in the data.
b. Feature Engineering: Create new features or transform existing ones to extract more meaningful information.
c. Normalization/Scaling: Standardize numerical features to have zero mean and unit variance.
d. Categorical Encoding: Convert categorical variables into numerical format using techniques like one-hot encoding.
6. Feature Selection Techniques:
a. Correlation Analysis: Identify features that are highly correlated with the target variable.
b. Feature Importance from Models: Use algorithms like Random Forest to estimate feature importance.
c. Univariate Feature Selection: Select features based on statistical tests such as chi-squared or ANOVA.
7. Machine Learning Algorithm:
Common algorithms for binary classification tasks like diabetes prediction include Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, and Neural Networks.
8. Model Training:

a. Split the dataset into training, validation, and test sets. b. Train the chosen model on the training data. c. Validate the model's performance on the validation set. d. Fine-tune hyperparameters to optimize model performance.

9. Evaluation Metrics:

Common evaluation metrics for a diabetes prediction model include:

a. Accuracy: The proportion of correctly predicted cases. b. Precision: The ratio of true positive predictions to the total positive predictions. c. Recall: The ratio of true positive predictions to all actual positive cases. d. **F1-Score:** The harmonic mean of precision and recall. e. AUC-ROC: Area Under the Receiver Operating Characteristic curve. f. Confusion Matrix: Providing detailed information about true positives, true negatives, false positives, and false negatives.

PROGRAM:

1.Importing the necessary packages:

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

Pandas-Pandas is a Python library used for working with data sets.
sklearn-Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling.

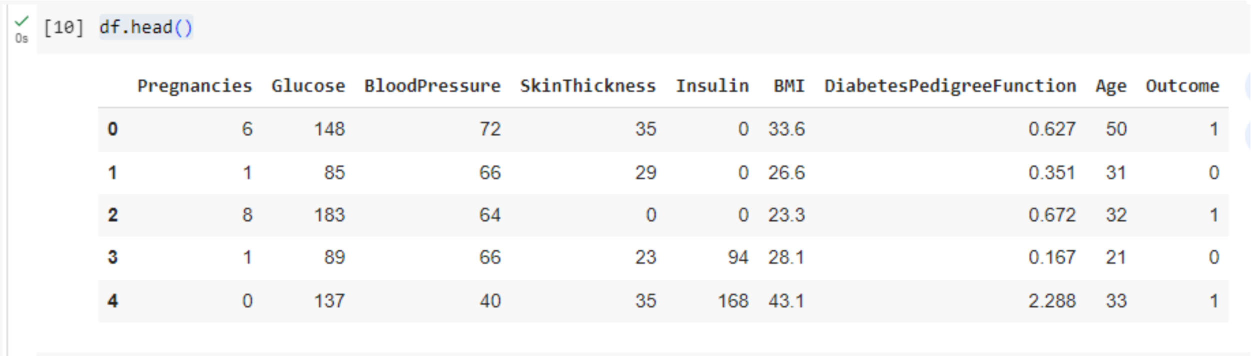
2.Loading the dataset:

```
df=pd.read_csv('/content/diabetes.csv')
```

Read_csv-read_csv is a method in pandas module, which is used to read the csv files.

3.Exploratory Data Analysis:

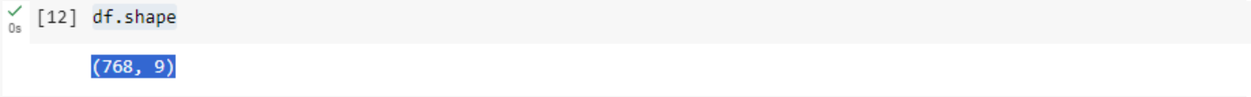
```
df.head()
```



```
[10] df.head()
```

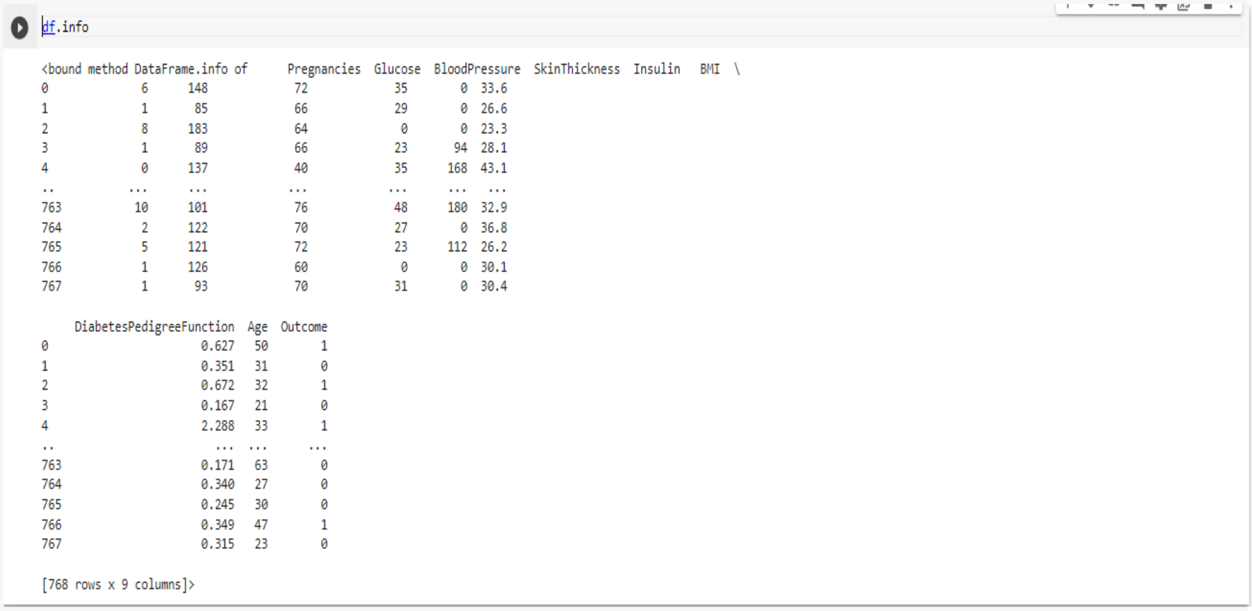
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

df.shape



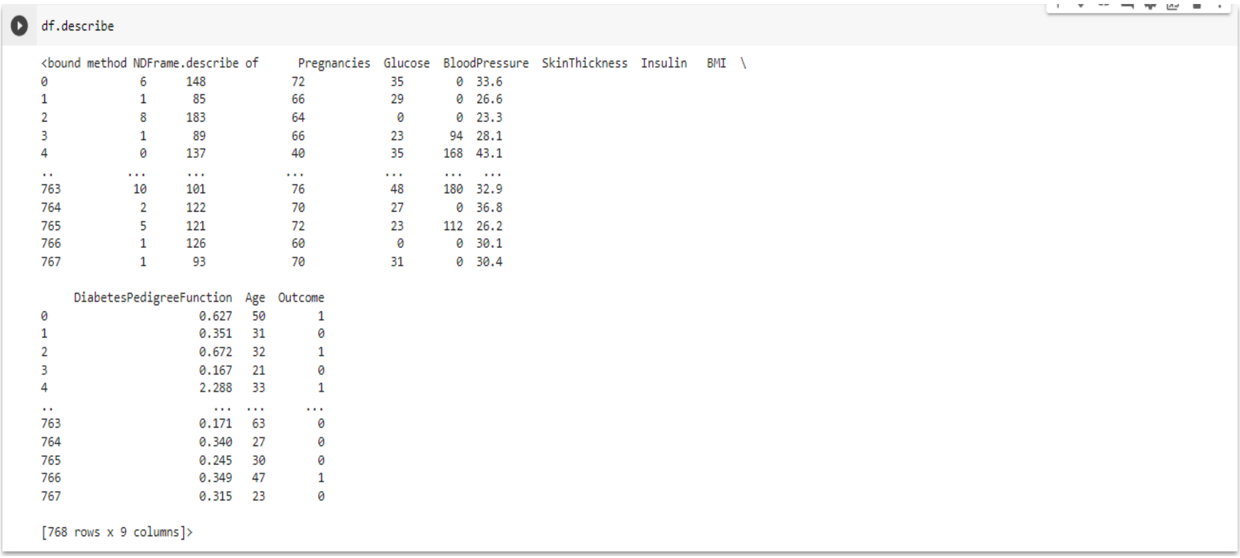
```
[12] df.shape

(768, 9)
```

df.info



```
df.info

<bound method DataFrame.info of      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  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

df.describe



```
df.describe
```

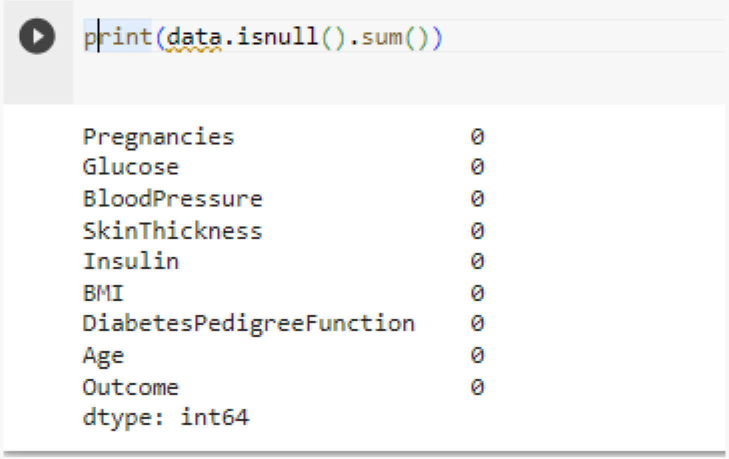
```
<bound method NDFrame.describe of      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              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  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

4.Separating Dataset into X and Y:

```
X=data.drop('Outcome',axis=1)
Y=data['Outcome']
X-Which doesn't store the 'outcome' field
Y-It stores only 'Outcome' field
```

5.Checking for Null values

```
print(data.isnull().sum())
```



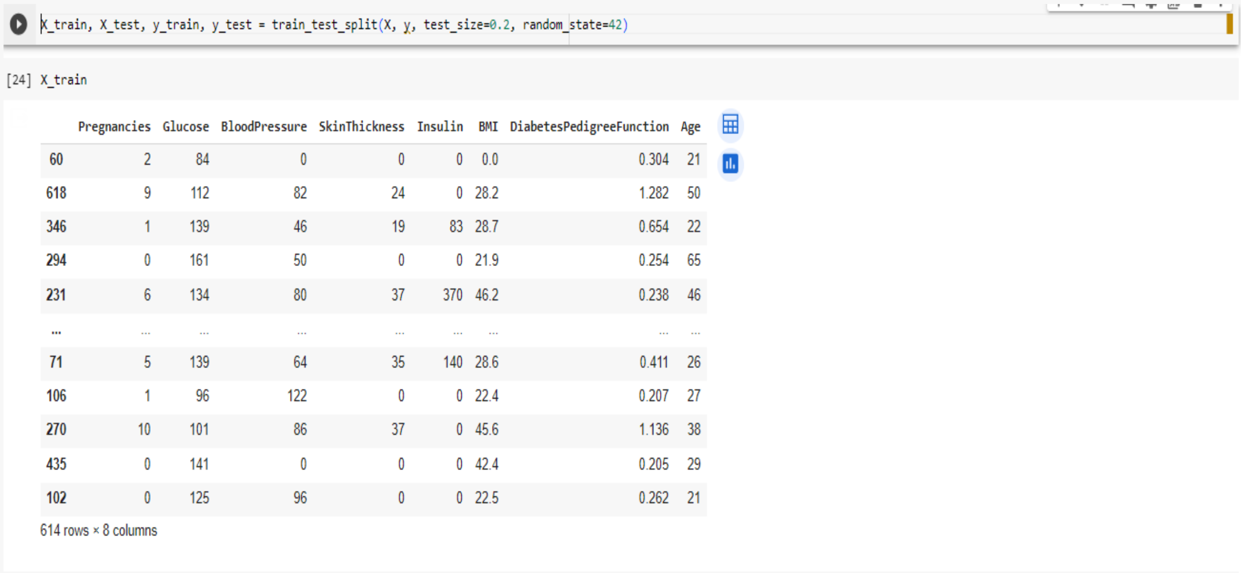
```
print(data.isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
There is no null values in the dataset.
```

6. Spliting dataset into test and training data:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```



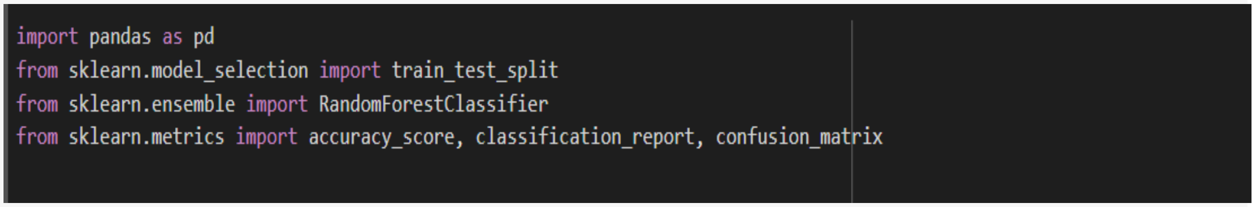
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[24] X_train
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
60	2	84	0	0	0	0.0	0.304	21
618	9	112	82	24	0	28.2	1.282	50
346	1	139	46	19	83	28.7	0.654	22
294	0	161	50	0	0	21.9	0.254	65
231	6	134	80	37	370	46.2	0.238	46
...	...	...	...	...	...	...	...	...
71	5	139	64	35	140	28.6	0.411	26
106	1	96	122	0	0	22.4	0.207	27
270	10	101	86	37	0	45.6	1.136	38
435	0	141	0	0	0	42.4	0.205	29
102	0	125	96	0	0	22.5	0.262	21

614 rows × 8 columns

7.Selecting the Machine learning model

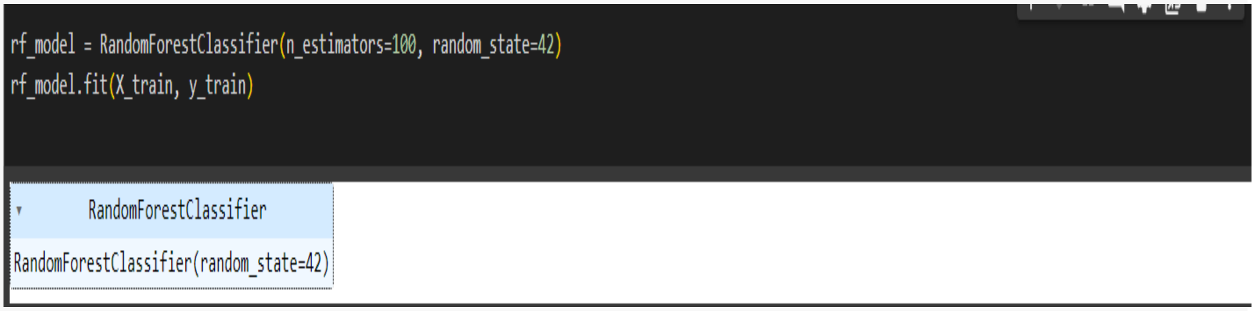


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Machine learning model=Random Forest

Random Forest grows multiple decision trees which are merged together for a more accurate prediction

8.Training the Model:



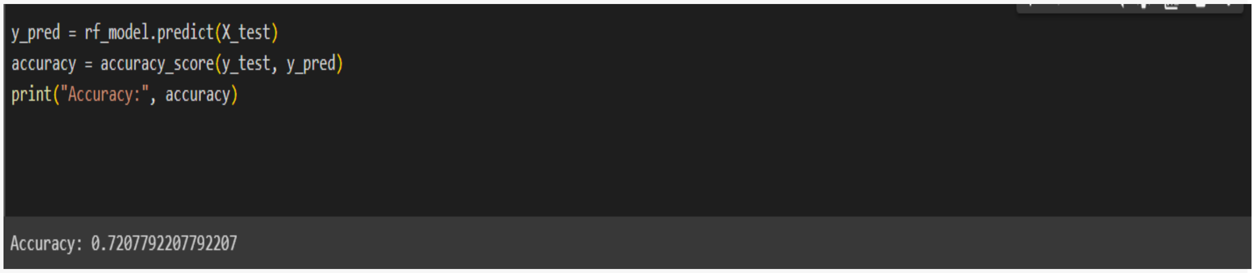
```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In above code, I have trained the model by using fit function of sklearn.ensemble module.

n_estimators=the number of trees you want to build before taking the maximum voting or averages of predictions.

9.Evaluating its performance:



```
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7207792207792207
```

This code explains that, by using X_test the model was predicting. And the by using accuracy_score, the performance is evaluated.