PHASE 3: DEVELOPMENT PHASE PART 1

COURSE	Artificial Intelligence
PROJECT	AI Based Diabetes Prediction System
DATE	22-10-2023

INTRODUCTION:

In an era defined by remarkable advancements in artificial intelligence and healthcare technology, the AI-Based Diabetes Prediction System represents a significant step towards enhancing healthcare services and improving the lives of individuals at risk of diabetes. The development phase of this project marks a pivotal stage in transforming our initial concept into a practical and functional solution.

PROJECT OVERVIEW

The AI-Based Diabetes Prediction System is a data-driven healthcare solution designed to predict the likelihood of an individual developing diabetes based on a set of key medical and demographic features. The project's primary objectives include:

- Data Preprocessing
- Model Selection
- Model Evaluation
- Model Deployment

DEVELOPMENT STEPS:

To begin building our project, we'll need to start with data loading, preprocessing and model selection. Below are the steps you can follow:

1.DATA COLLECTION AND LOADING

The very first step is to choose the dataset for our model .This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Now we have to set the development environment to build our project. Import the necessary libraries to built the model. We are ensembleing the random forest ,SVM and logistic regression model .

Code:

```
import numpy as np
import pandas as pd
```

Here's a quick overview of the libraries we've imported and what each of them is used for:

- numpy (import numpy as np): NumPy is a fundamental library for scientific computing in Python.
- pandas (import pandas as pd): Pandas is a widely used library for data manipulation and analysis. It provides data structures like DataFrame .

Code:

```
# Load the dataset
data = pd.read_csv("/content/sample_data/diabetes.csv")
```

- The primary task is to read the data from the "diabetes.csv" file. This can be done using the read_csv function provided by pandas. The loaded data is stored in a pandas DataFrame, which is a two-dimensional, size-mutable, and potentially heterogeneous tabular data structure with labeled axes (rows and columns). In this case, the DataFrame is assigned to the variable "data."
- After running this code, the "data" variable will contain the contents of the "diabetes.csv" file as a DataFrame, and we can use various pandas methods and functions to analyze, manipulate, and visualize the data as needed for our project or analysis.

2.DATA_EXPLORATION:

Exploring the data is a crucial step in understanding the dataset, identifying patterns, and gaining insights that can guide our modeling process. Here are some common data exploration techniques :

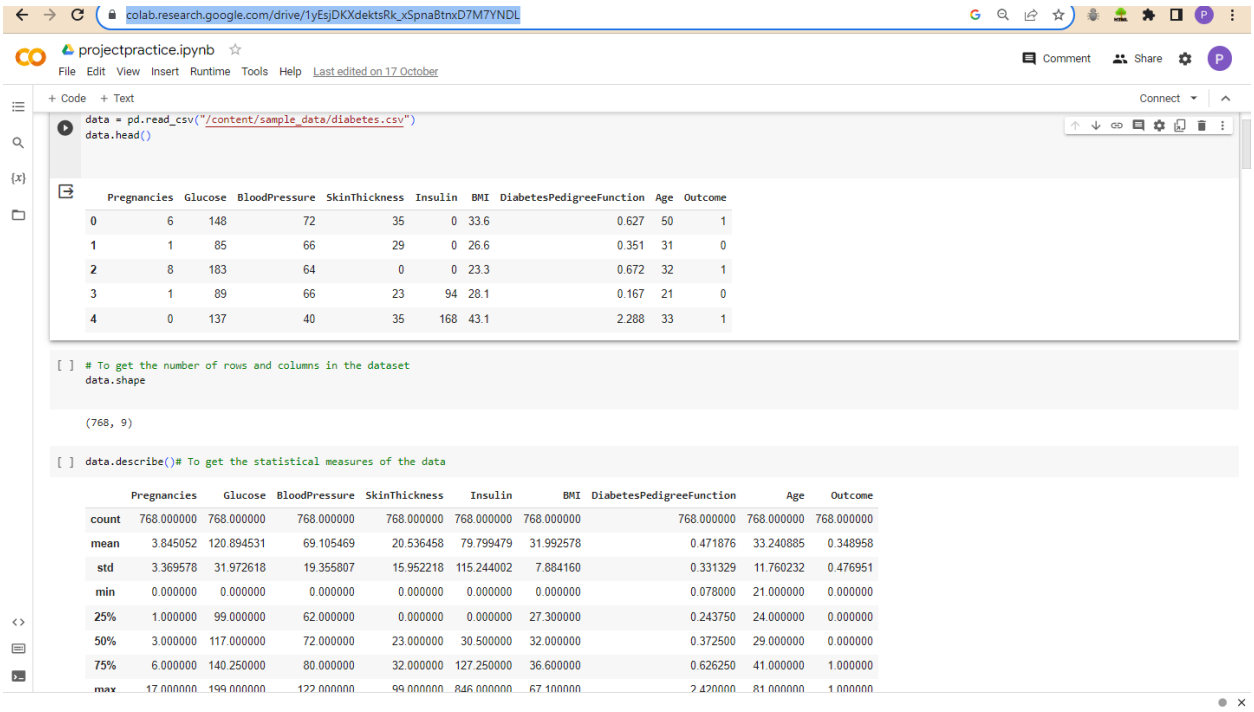
Code:

```
data.head()
# To get the number of rows and columns in the dataset
data.shape
# To get the statistical measures of the data
data.describe()# To get the statistical measures of the data

data.columns
```

- **data.head()**: It gives you a quick preview of what your data looks like, including the first five rows by default.
- **data.shape**: This will returns the number of rows and columns in your dataset. The output will be in the format (number of rows, number of columns).
- **data.describe()**: This code provides statistical summary measures for your dataset.
- **data.columns**: This code returns the column names (features) of your dataset.

Output:



	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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

3.DATA PREPROCESSING AND SPLITTING:

Here are some common data preprocessing steps we can follow ,

- **Handling Missing Values:**

  Check for missing values in your dataset and decide how to handle them. Checking for missing values is an essential step in data preprocessing to ensure that the dataset is clean and ready for analysis or modeling.

Code:

```
missing_values = data.isnull().sum()
print(missing_values)   # Check for missing values


data.isnull().values.any()
```

The code **data.isnull().values.any()** is used to check whether there are any missing (NaN) values in the DataFrame **data**. If the code returns **True**, it means that there are missing values in the DataFrame **data**. If it returns **False**, it means that there are no missing values in the DataFrame.

Output:

```
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values)
```

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

```
# Example: Impute missing values with the mean

data.isnull().values.any()
```

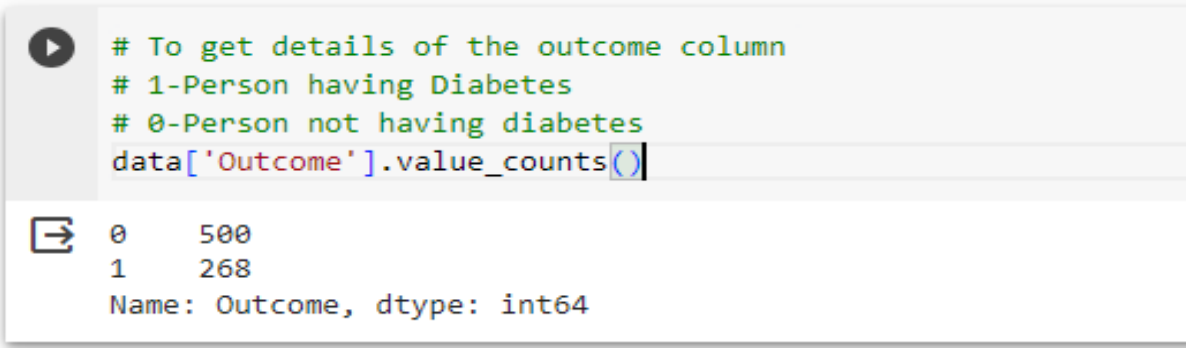
```
False
```

Code:

```
# To get details of the outcome column
# 1-Person having Diabetes
# 0-Person not having diabetes

data['Outcome'].value_counts()
```

Output:



Code:

```
# separating the data and labels
X = data.drop(columns = 'Outcome', axis=1)
Y = data['Outcome']

# To print the independent variables
print(X)
# To print the outcome variable
print(Y)
```

In this code, you are separating your dataset into two essential components for machine learning:

1. X: This contains the independent variables (features) and is used to store all the data attributes that will be used as input for your machine learning model.
2. Y: This contains the dependent variable (outcome or label) and is used to store the target you want to predict or classify.

Output:

SPLITTING THE DATA

```
# separating the data and labels
X = data.drop(columns = 'Outcome', axis=1)
Y = data['Outcome']

# To print the independent variables
print(X)
```

```
     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]
```

```
# To print the outcome variable
print(Y)
```

```
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
```

CHECK OUT THE CODE :

https://colab.research.google.com/drive/1llxazFt1201IWkvuMcy29G0KOArcgw6u#scrollTo=L06tBl2x0jOI

CONCLUSION:

In the first part of our project, we've covered the essential steps for handling our diabetes dataset. We started by loading the data from a CSV file and creating a DataFrame. Then, we addressed data preprocessing tasks, such as checking for missing values and ensuring the data is in a suitable format. Additionally, we split the data into training and testing sets for model development. Now, as we conclude this phase, we'll move on to the second part of our project. In this next phase, we will focus on selecting a machine learning model and training it using our preprocessed data.