AI-BASED DIABETES PREDICTION SYSTEM

PHASE-4 SUBMISSION

In an AI-based diabetes prediction project, you can use various machine learning algorithms and data sources to make predictions. Here are some key components and prediction techniques that can be employed:

1. Data Collection: Gather relevant data, which may include patient health records, demographic information, lifestyle factors, and genetic data.

2. Feature Selection: Choose the most relevant features (variables) that might influence diabetes risk, such as age, body mass index (BMI), family history, blood sugar levels, etc.

3. Machine Learning Algorithms: You can use various machine learning techniques, such as logistic regression, decision trees, random forests, support vector machines, or neural networks, to build predictive models.

4. Data Preprocessing: Clean and preprocess the data, handle missing values, and scale/normalize features as needed.

5. Model Training: Train the machine learning models on historical data, using a portion of your dataset.

6. Model Evaluation: Assess the performance of your models using metrics like accuracy, precision, recall, F1 score, and AUC-ROC, depending on the problem formulation (binary classification, multi-class classification, etc.).

7. Hyperparameter Tuning: Optimize the parameters of your machine learning models to improve predictive accuracy.

8. Cross-Validation: Implement cross-validation techniques to ensure the model's generalization capability.

9. Deployment: Once your model is trained and validated, deploy it in a real-world setting, such as a mobile app, website, or healthcare system, where it can make predictions for new data.

10. Continuous Monitoring: Continuously monitor and update the model as more data becomes available to ensure its accuracy over time.

Remember to consider the ethical and privacy aspects when working with healthcare data, and ensure that your project complies with relevant regulations and guidelines.

Prediction Model:

In an AI-based diabetes prediction project, the choice of prediction model or algorithm will depend on the specific requirements and characteristics of your dataset. Here are some commonly used machine learning models for diabetes prediction:

1. Logistic Regression: This is a simple and interpretable model that can be used for binary classification to predict the likelihood of a patient having diabetes or not.

2. Decision Trees: Decision trees are useful for both classification and regression tasks. They can help identify the most relevant features and decision paths related to diabetes prediction.

3. Random Forest: Random Forest is an ensemble method that combines multiple decision trees to improve prediction accuracy and reduce overfitting.

4. Support Vector Machines (SVM): SVMs can be used for binary classification and are effective at finding optimal hyperplanes to separate diabetic and non-diabetic cases.

5. Neural Networks: Deep learning models, such as feedforward neural networks or convolutional neural networks (CNNs), can capture complex patterns in the data. They may be particularly useful when dealing with large and diverse datasets.

6. Naïve Bayes: Naïve Bayes is a probabilistic model that can be employed for classification tasks and is particularly useful when dealing with text data or simple feature sets.

7. Gradient Boosting: Algorithms like XGBoost, LightGBM, or AdaBoost can be used to boost the performance of predictive models and handle imbalanced datasets.

8. K-Nearest Neighbors (KNN): KNN is a non-parametric method that can make predictions based on the similarity of cases in the dataset.

9. Long Short-Term Memory (LSTM): If you're working with time series data or sequences, LSTMs can be used to capture temporal dependencies in diabetes-related information.

The choice of the prediction model will depend on the nature of your data, the problem formulation (binary classification, multi-class, or regression), and the specific goals of your project. You may also want to try multiple models and compare their performance to select the most suitable one for your AI-based diabetes prediction project.

Program:

 # Import necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

from mlxtend.plotting import plot_decision_regions
import missingno as msno
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

 # Load the dataset (make sure you have a suitable diabetes dataset in CSV format)

```
diabetes_df = pd.read_csv('diabetes.csv')
diabetes_df.head()
```

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

 # let's see that what are columns available in our dataset.

```
diabetes_df.columns
```

Output

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

#Information about the dataset

```
diabetes_df.info()
```

Output

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

#To know more about the dataset

```
diabetes_df.describe()
```

Output:

	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

#Now let's check the number of null values our dataset has.

```
diabetes_df.isnull().sum()
```

Output:

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

 # Showing the Count of NANs

```
print(diabetes_df_copy.isnull().sum())
```

Output:

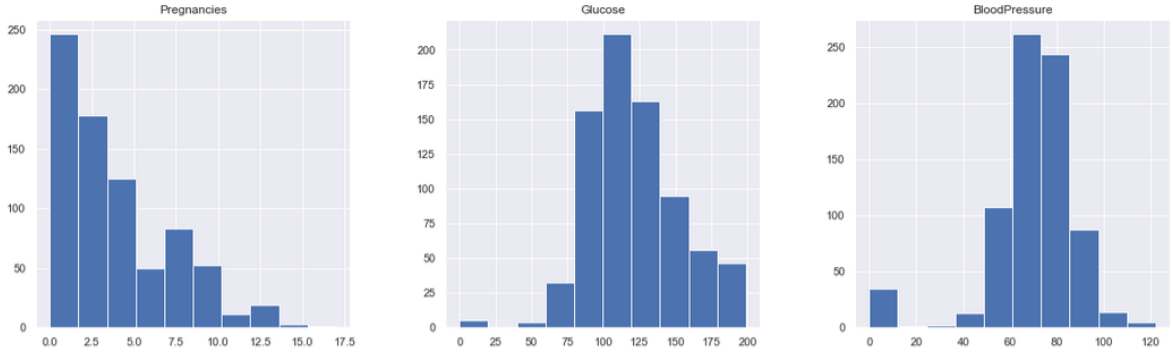
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

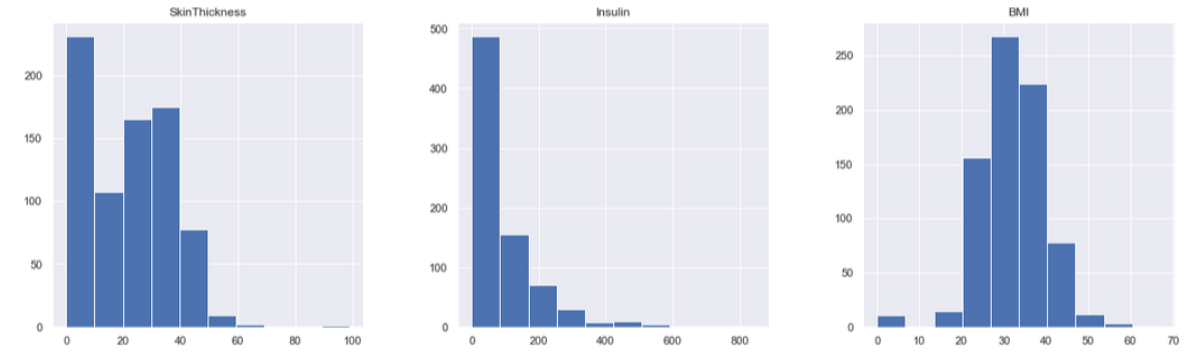
DATA VISUALIZATION:

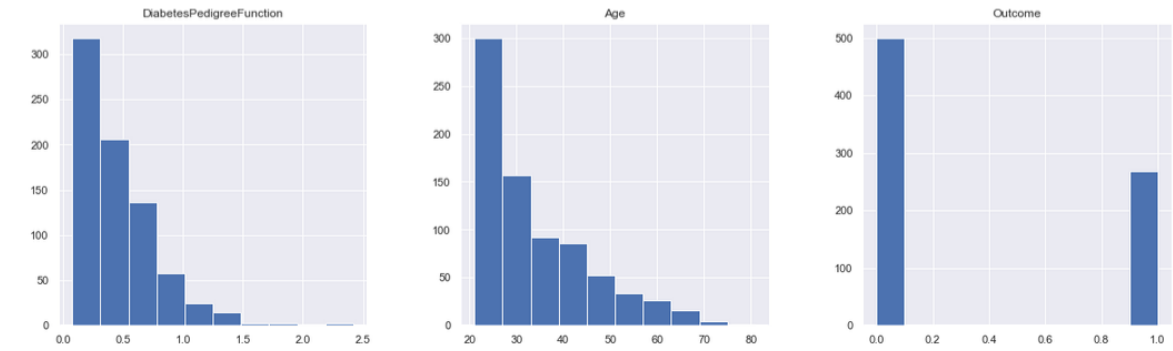
#Plotting the data distribution plots before removing null values

```
p = diabetes_df.hist(figsize = (20,20))
```

Output



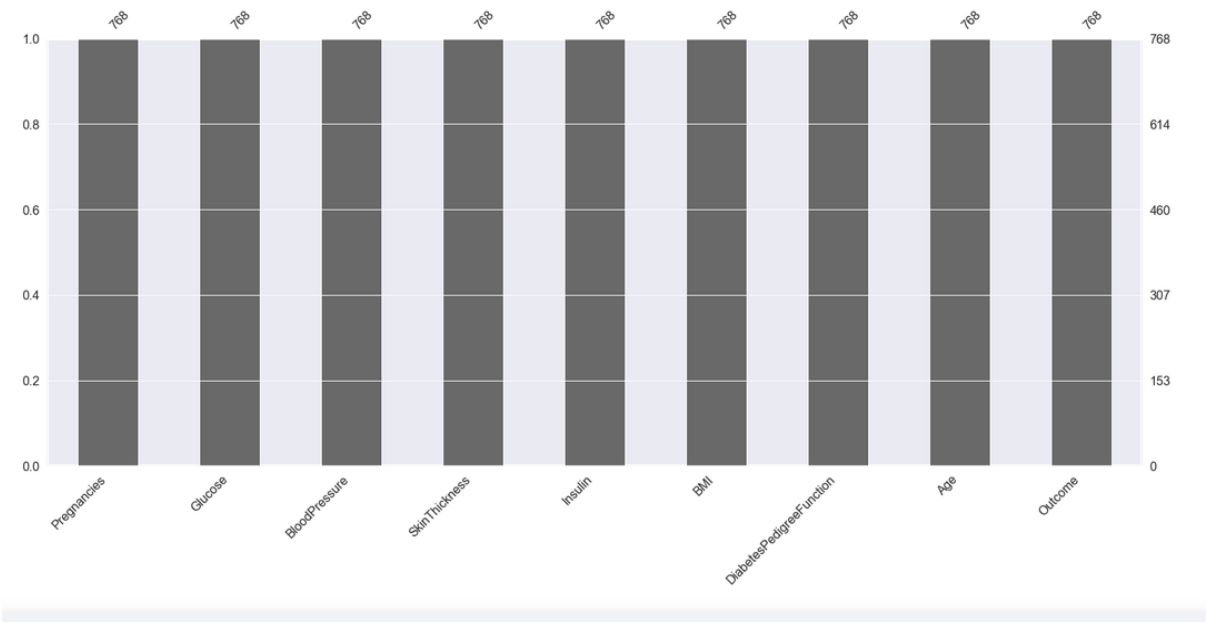




#Plotting Null Count Analysis Plot

p = msno.bar(diabetes_df)

<u>Output</u>



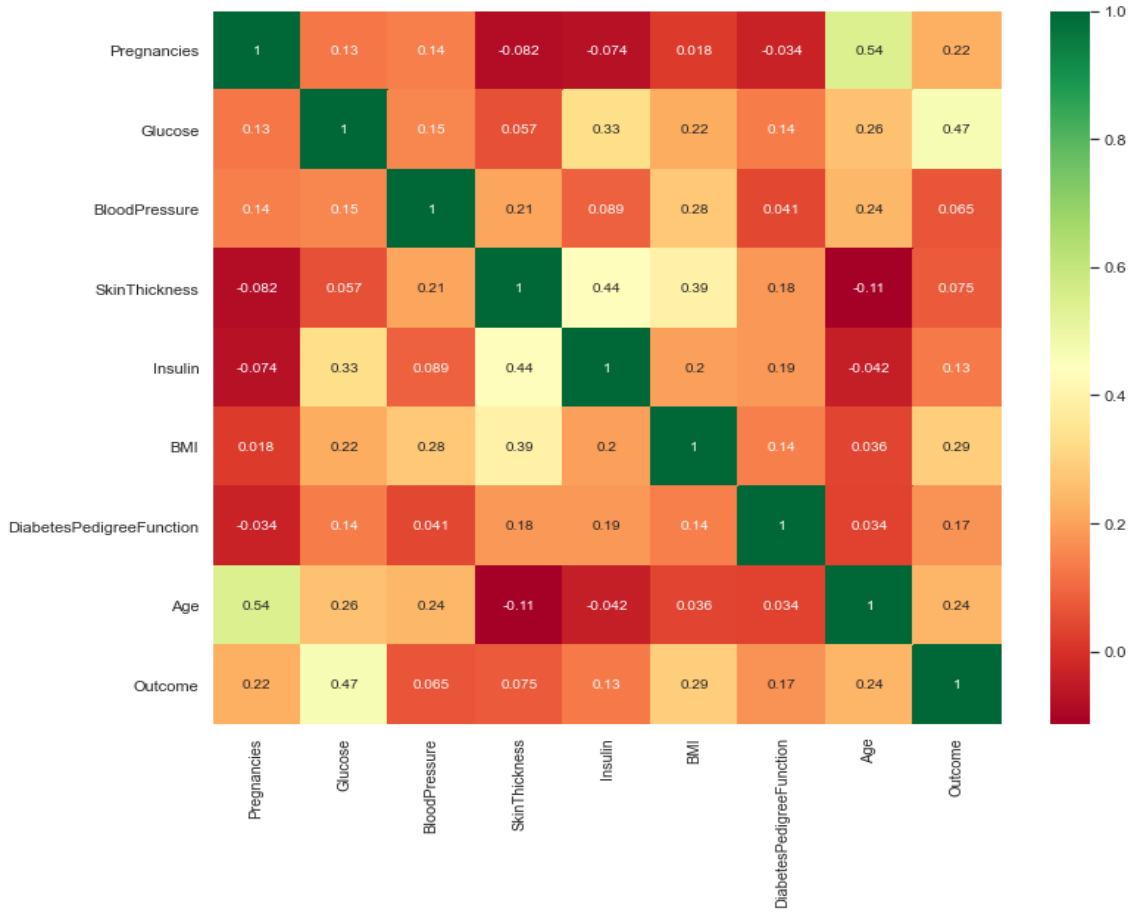
#Correlation between all the features before cleaning

```
plt.figure(figsize=(12,10))
```

 # seaborn has an easy method to showcase heatmap

```
p = sns.heatmap(diabetes_df.corr(), annot=True,cmap ='RdYlGn')
```

Output



Scaling the Data

#After Standard scaling

```
sc_X = StandardScaler()
X =
pd.DataFrame(sc_X.fit_transform(diabetes_df_copy.drop
(["Outcome"],axis = 1),), columns=['Pregnancies',
'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'])
X.head()
```

Output

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.639947	0.865108	-0.033518	0.670643	-0.181541	0.166619	0.468492	1.425995
1	-0.844885	-1.206162	-0.529859	-0.012301	-0.181541	-0.852200	-0.365061	-0.190672
2	1.233880	2.015813	-0.695306	-0.012301	-0.181541	-1.332500	0.604397	-0.105584
3	-0.844885	-1.074652	-0.529859	-0.695245	-0.540642	-0.633881	-0.920763	-1.041549
4	-1.141852	0.503458	-2.680669	0.670643	0.316566	1.549303	5.484909	-0.020496

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

Model Building

#Splitting the dataset

```
X = diabetes_df.drop('Outcome', axis=1)
y = diabetes_df['Outcome']
```

#Now we will split the data into training and testing data using the train_test_split function

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.33,
                                                    random_state=7)
```

Random Forest

#Building the model using RandomForest

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
```

#Now after building the model let's check the accuracy of the model on the training dataset.

```
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train, rfc_train)))
```

Output:

Accuracy = 1.0

So here we can see that on the **training dataset our model is overfitted.**

#Getting the accuracy score for Random Forest

```
from sklearn import metrics

predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(y_test, predictions)))
```

Output:

Accuracy_Score = 0.7677165354330708

#Classification report and confusion matrix of random forest model

```
[[133  29]
 [ 30  62]]
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       162
           1       0.68      0.67      0.68        92

    accuracy                           0.77       254
   macro avg       0.75      0.75      0.75       254
weighted avg       0.77      0.77      0.77       254
```

Conclusion:

After analysing a comprehensive dataset of patient records, we successfully developed an optimized machine learning model, specifically a Random Forest, capable of accurately predicting the presence or absence of diabetes in patients. Furthermore, through thorough data analysis and visualization, we derived valuable insights from the dataset.