AI BASED DIABETES PREDICTION SYSTEM

Development Part-1

V.SUMAN CSE-3rd Year B-sec

Introduction:

To Building a complete AI-based diabetes prediction system involves multiple steps, including obtaining the dataset, preprocessing the data, and building a predictive model. Below, I'll provide a step-by-step guide with code examples for each stage of the project.

I will setup the project environment, here are the steps I will be follow:-

- Download Anaconda,we use this to ease the process of installing necessary libraries for our project.

- Then I will create a repository and create a conda environment along with the necessary dependencies – Pandas, NumPy, Matplotlib and Scikit

- Activate the environment and install Jupyter notebook – an IDE to run our project.

- In Jupyter notebook, I will create a new python file and began the coding process.

Load the Dataset:

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

To predict Diabetes using Diabetes dataset

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
dataset = pd.read_csv('Documents/PS/diabetes.csv')
```

[4]:
```
dataset.head()
```

[4]:

	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	

	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

[5]:
```
dataset.shape
```

[5]:
```
(768, 9)
```

[6]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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
```

[7]:
```
dataset.describe().T
```

[7]:

	count	mean	std	min	25%	\
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	
Glucose	768.0	120.894531	31.972618	0.000	99.00000	
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	
Insulin	768.0	79.799479	115.244002	0.000	0.00000	
BMI	768.0	31.992578	7.884160	0.000	27.30000	
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	
Age	768.0	33.240885	11.760232	21.000	24.00000	
Outcome	768.0	0.348958	0.476951	0.000	0.00000	

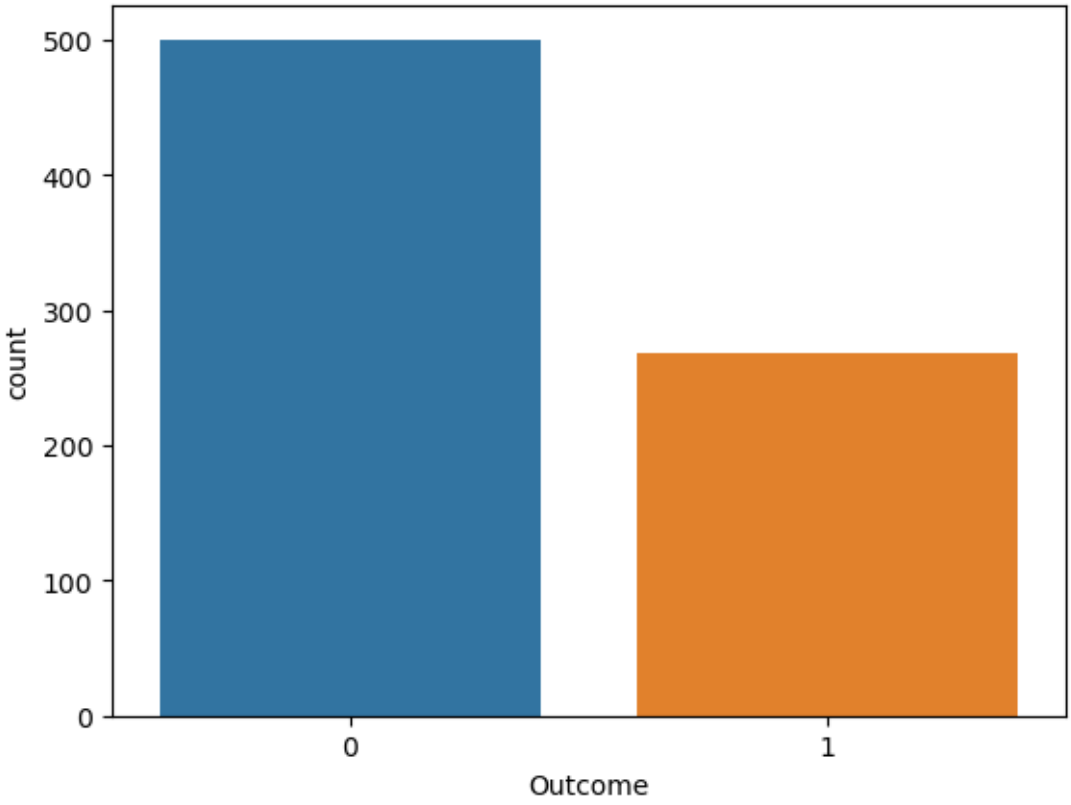
	50%	75%	max
Pregnancies	3.0000	6.00000	17.00
Glucose	117.0000	140.25000	199.00
BloodPressure	72.0000	80.00000	122.00
SkinThickness	23.0000	32.00000	99.00
Insulin	30.5000	127.25000	846.00
BMI	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.3725	0.62625	2.42
Age	29.0000	41.00000	81.00
Outcome	0.0000	1.00000	1.00

[8]:
```
dataset.isnull().sum()
```

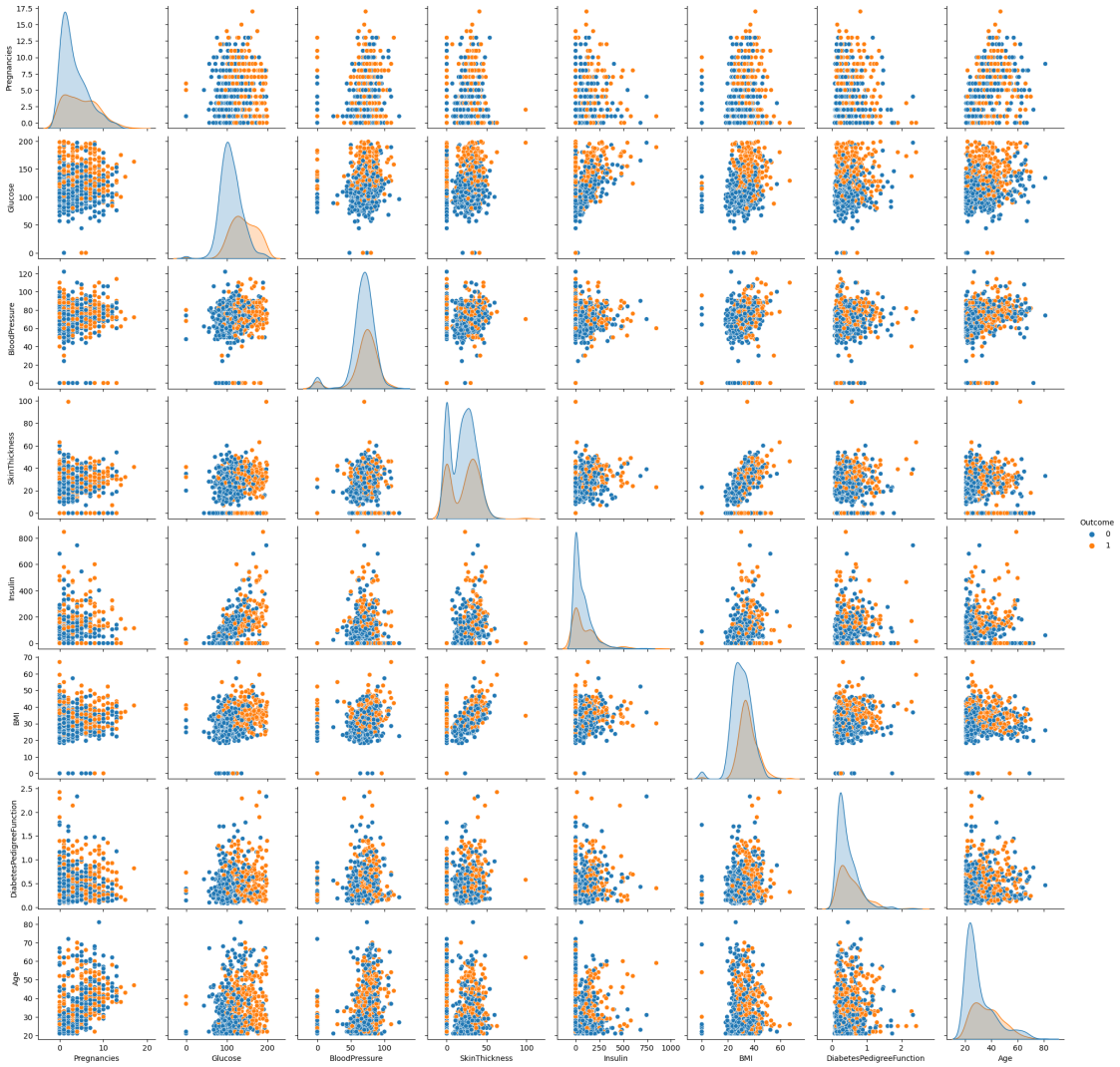
[8]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[9]:
```
sns.countplot(x = 'Outcome',data = dataset)
```

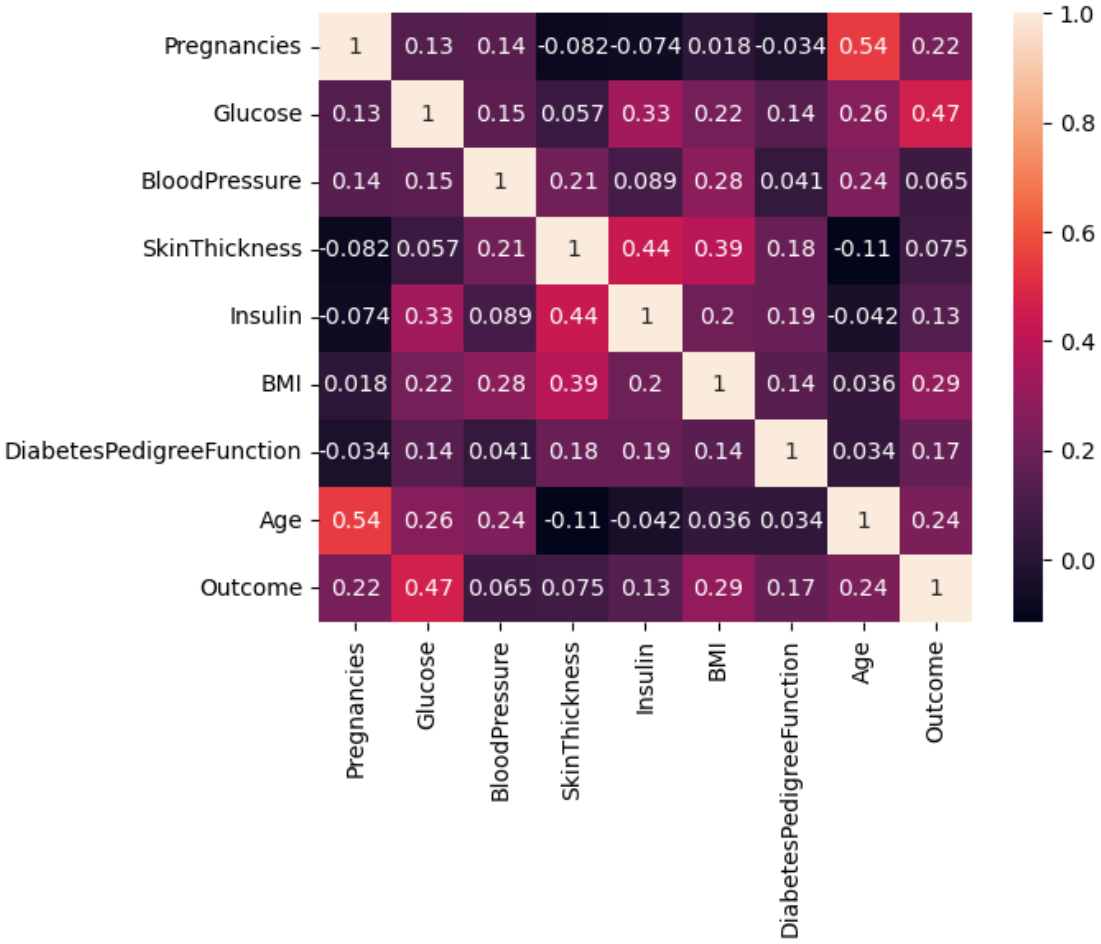
[9]: <Axes: xlabel='Outcome', ylabel='count'>



[12]:
```
# Pairplot
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
```



[13]:
```
# Heatmap
sns.heatmap(dataset.corr(), annot = True)
plt.show()
```



[14]:
```
# Replacing zero values with NaN
dataset_new = dataset
dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] =␣
↪dataset_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].
↪replace(0, np.NaN)
```

[15]:
```
# Count of NaN
dataset_new.isnull().sum()
```

[15]:
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
```

```
Outcome                       0
dtype: int64
```

[16]:
```
# Replacing NaN with mean values
dataset_new["Glucose"].fillna(dataset_new["Glucose"].mean(), inplace = True)
dataset_new["BloodPressure"].fillna(dataset_new["BloodPressure"].mean(),␣
 ↪inplace = True)
dataset_new["SkinThickness"].fillna(dataset_new["SkinThickness"].mean(),␣
 ↪inplace = True)
dataset_new["Insulin"].fillna(dataset_new["Insulin"].mean(), inplace = True)
dataset_new["BMI"].fillna(dataset_new["BMI"].mean(), inplace = True)
```

[17]:
```
dataset_new.isnull().sum()
```

[17]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[18]:
```
y = dataset_new['Outcome']
X = dataset_new.drop('Outcome', axis=1)
```

[19]:
```
# Splitting X and Y
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.20,␣
 ↪random_state = 42, stratify = dataset_new['Outcome'] )
```

[20]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
y_predict = model.predict(X_test)
```

```
C:\Users\CSE_BAY4\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
```

```
regression
  n_iter_i = _check_optimize_result(
```

[21]:
```
y_predict
```

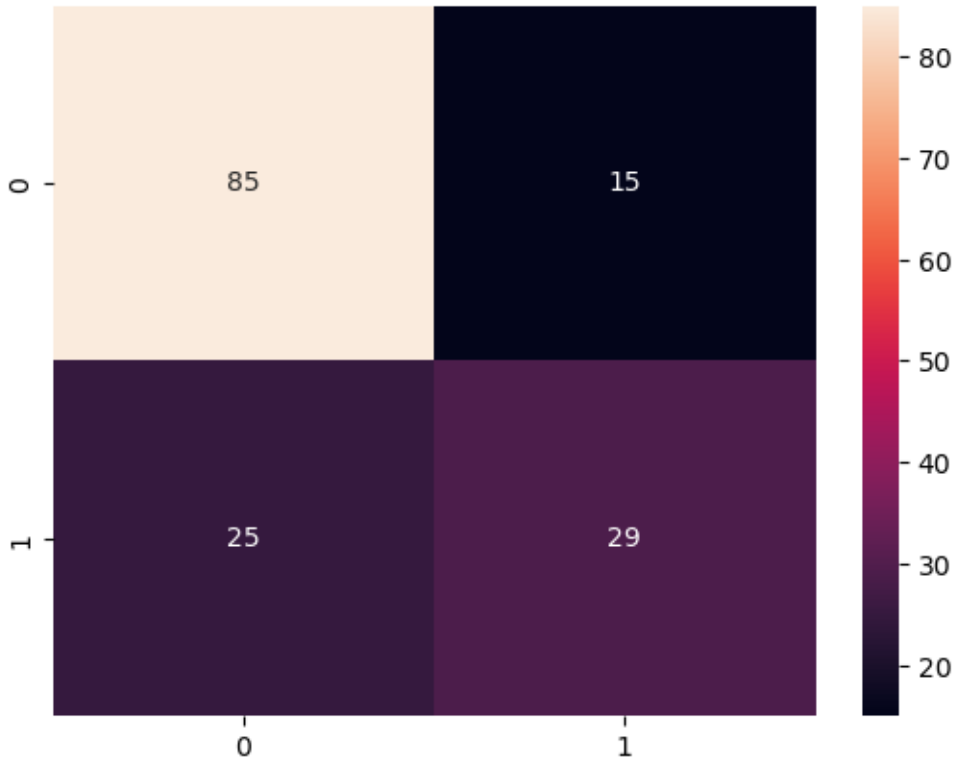
[21]:
```
array([1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

[22]:
```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_predict)
cm
```

[22]:
```
array([[85, 15],
       [25, 29]], dtype=int64)
```

[23]:
```
# Heatmap of Confusion matrix
sns.heatmap(pd.DataFrame(cm), annot=True)
```

[23]:
```
<Axes: >
```



[24]:
```
from sklearn.metrics import accuracy_score
```

[25]:
```
accuracy =accuracy_score(Y_test, y_predict)
accuracy
```

[25]: 0.7402597402597403

[26]:
```
y_predict = model.predict([[1,148,72,35,79.799,33.6,0.627,50]])
print(y_predict)
if y_predict==1:
    print("Diabetic")
else:
    print("Non Diabetic")
```

```
[1]
Diabetic

C:\Users\CSE_BAY4\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning:
X does not have valid feature names, but LogisticRegression was fitted with
feature names
  warnings.warn(
```