AI BASEDDIABETES-PREDICTION PHASE-3

Coding for Diabetes Prediction System

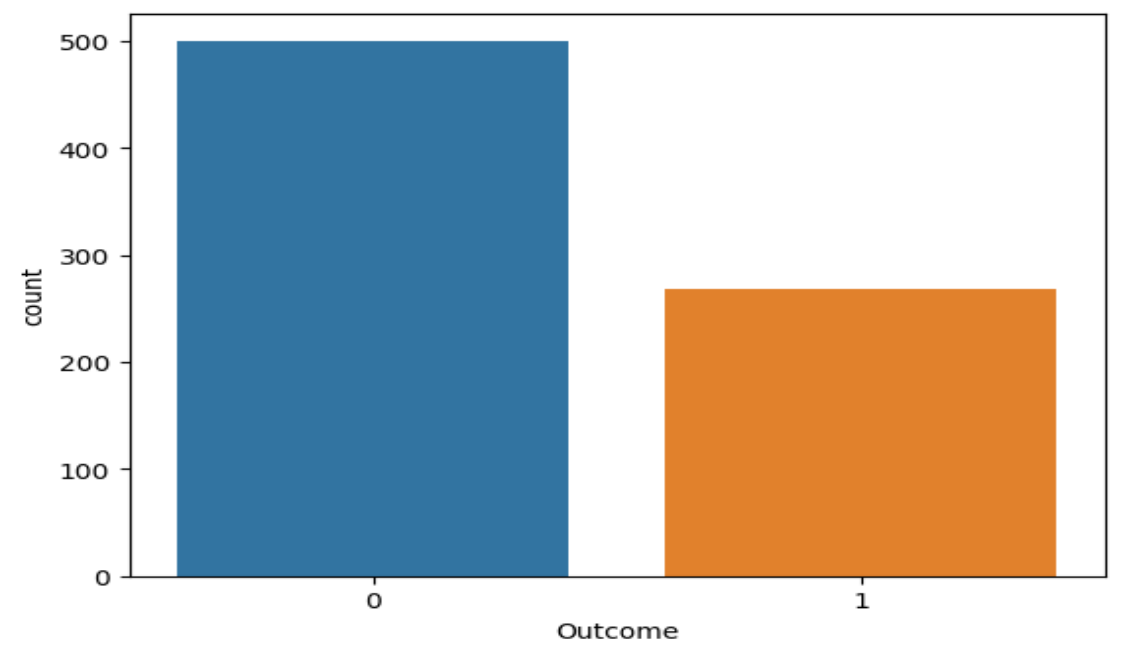
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
dataset=pd.read_csv("/kaggle/input/diabetes-data-
set/diabetes.csv“)
dataset.head()
```

Dataset::

Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	Outcome
0	6	556	34.45	90.00	45.90
1	1	234	56.78	848.0	09.76
2	8	865	23.89	09.8	87.78
3	1	846	09.89	45.0	87.56

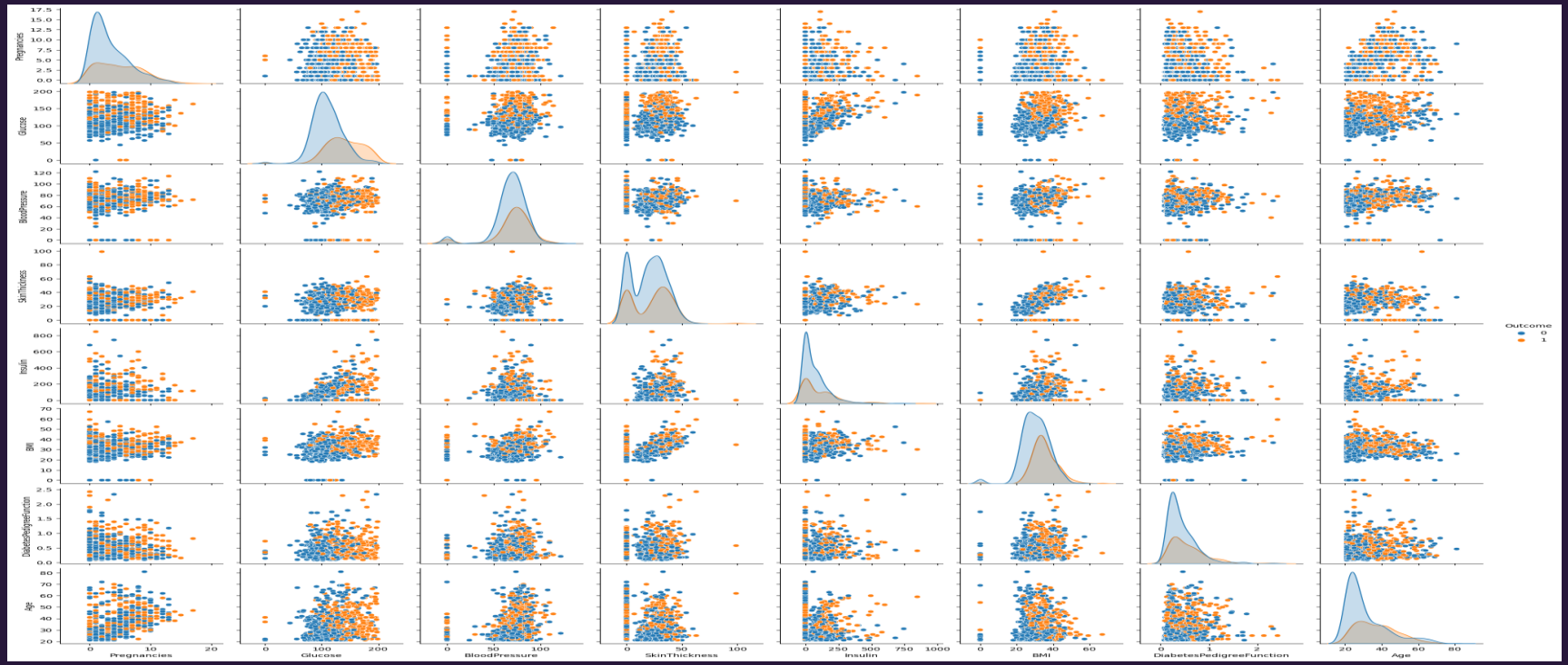
#**data visualization**

sns.countplot(x = 'Outcome',data = dataset)



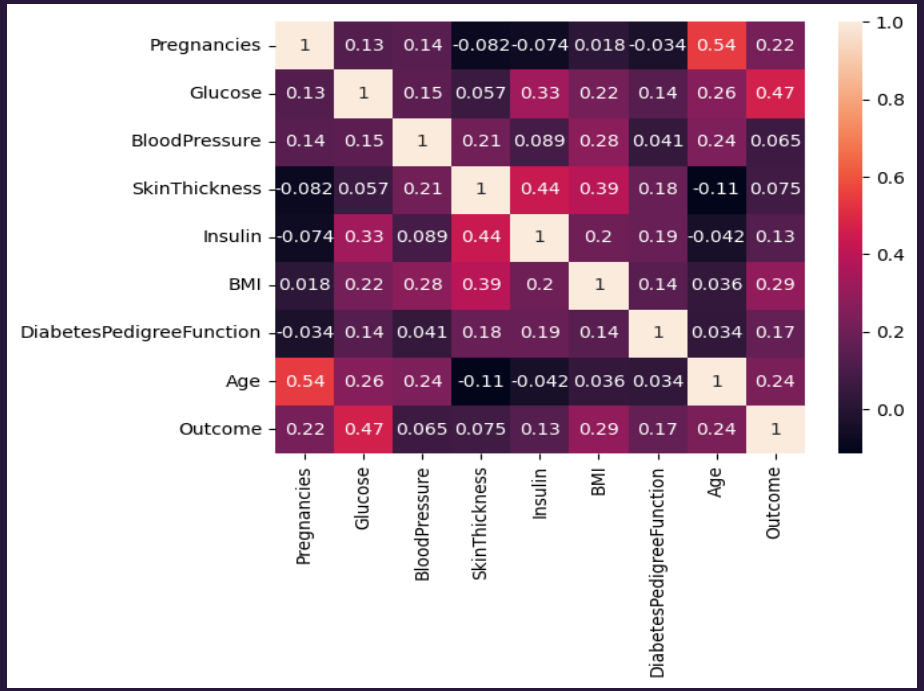
Pairplot

```
sns.pairplot(data = dataset, hue = 'Outcome')
plt.show()
```



Heatmap

```
sns.heatmap(dataset.corr(), annot = True)
plt.show()
```



```
# 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)
# Count of NaN
dataset_new.isnull().sum()
```

Pregnancies	0
Glucose	5
BloodPressure	35
SkinThickness	227
Insulin	374
BMI	11
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

```
# 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)
dataset_new.isnull().sum()
```

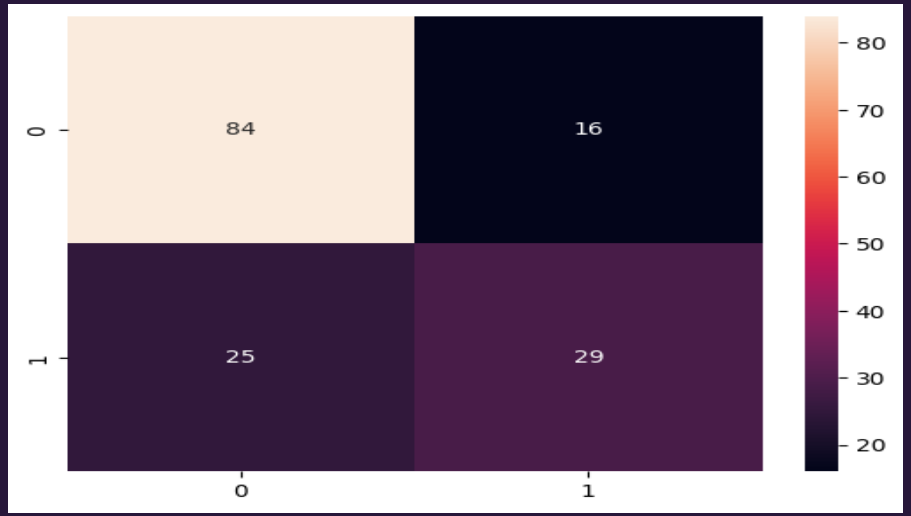
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
#Logistic regression
y = dataset_new['Outcome']
X = dataset_new.drop('Outcome', axis=1)
# 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'] )
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, Y_train)
y_predict = model.predict(X_test)
print(y_predict)
```

```
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, 0, 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, 1, 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, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0])
```

```
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, y_predict)
cm
array([[84, 16],
       [25, 29]])
# Heatmap of Confusion matrix
sns.heatmap(pd.DataFrame(cm), annot=True

)
```



```
from sklearn.metrics import accuracy_score
accuracy =accuracy_score(Y_test, y_predict)
accuracy
0.7337662337662337
```
#Example: Let's check whether the person have diabetes or not using some random values

```
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")
```