diabetes_KNN

November 7, 2023

[1]:
```
import numpy as np
import pandas as pd
```

[3]:
```
df = pd.read_csv(r"C:\Users\lalit\OneDrive\Desktop\practicals\machine learning\practical3\diabetes.csv")
```

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

[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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

[5]:
```
df.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   Pedigree       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
```

[6]:
```
df.head(20)
```

[6]:

	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	
5	5	116	74	0	0	25.6	
6	3	78	50	32	88	31.0	
7	10	115	0	0	0	35.3	
8	2	197	70	45	543	30.5	
9	8	125	96	0	0	0.0	
10	4	110	92	0	0	37.6	
11	10	168	74	0	0	38.0	
12	10	139	80	0	0	27.1	
13	1	189	60	23	846	30.1	
14	5	166	72	19	175	25.8	
15	7	100	0	0	0	30.0	
16	0	118	84	47	230	45.8	
17	7	107	74	0	0	29.6	
18	1	103	30	38	83	43.3	
19	1	115	70	30	96	34.6	

	Pedigree	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	0.201	30	0
6	0.248	26	1
7	0.134	29	0
8	0.158	53	1
9	0.232	54	1
10	0.191	30	0
11	0.537	34	1
12	1.441	57	0
13	0.398	59	1
14	0.587	51	1
15	0.484	32	1
16	0.551	31	1
17	0.254	31	1
18	0.183	33	0

```
19     0.529   32        1
```

[7]:
```
df.isnull()
```

[7]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI
0          False    False          False          False    False  False  \
1          False    False          False          False    False  False
2          False    False          False          False    False  False
3          False    False          False          False    False  False
4          False    False          False          False    False  False
..           …      …              …              …        …      …
763        False    False          False          False    False  False
764        False    False          False          False    False  False
765        False    False          False          False    False  False
766        False    False          False          False    False  False
767        False    False          False          False    False  False

     Pedigree    Age  Outcome
0       False  False    False
1       False  False    False
2       False  False    False
3       False  False    False
4       False  False    False
..        …    …       …
763     False  False    False
764     False  False    False
765     False  False    False
766     False  False    False
767     False  False    False

[768 rows x 9 columns]
```

[10]:
```
df.shape
```

[10]:
```
(768, 9)
```

[11]:
```
df.dtypes
```

[11]:
```
Pregnancies        int64
Glucose            int64
BloodPressure      int64
SkinThickness      int64
Insulin            int64
BMI              float64
Pedigree         float64
Age                int64
Outcome            int64
```

```
dtype: object
```

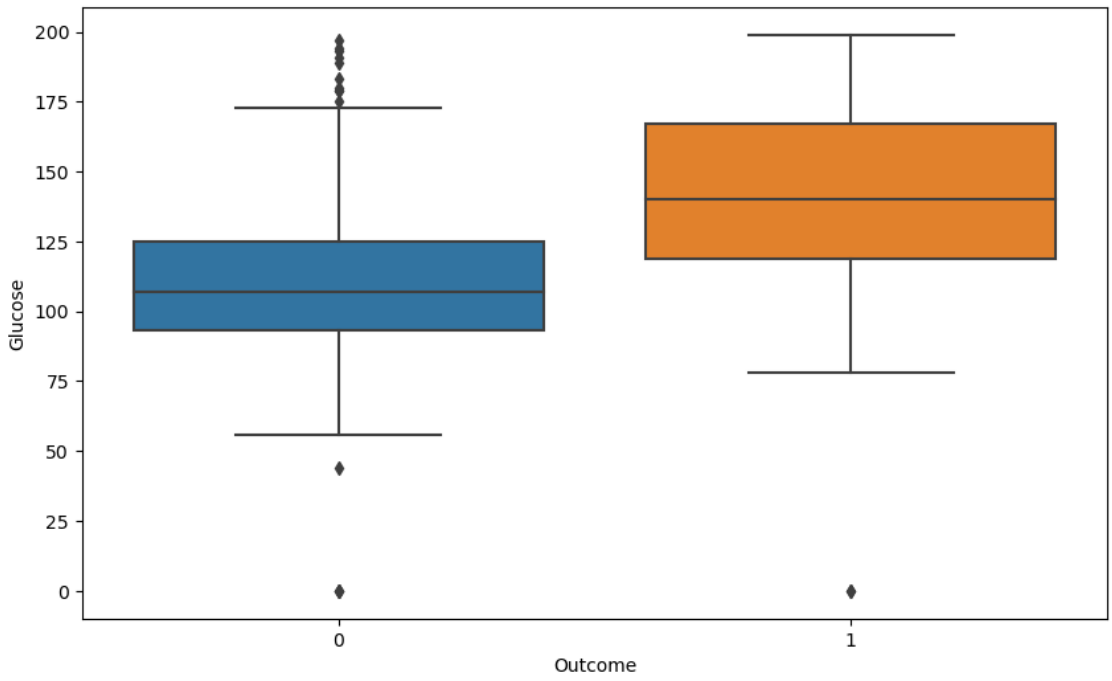
[12]:
```
df.isnull().sum()
```

[12]:
```
Pregnancies      0
Glucose          0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
Pedigree         0
Age              0
Outcome          0
dtype: int64
```

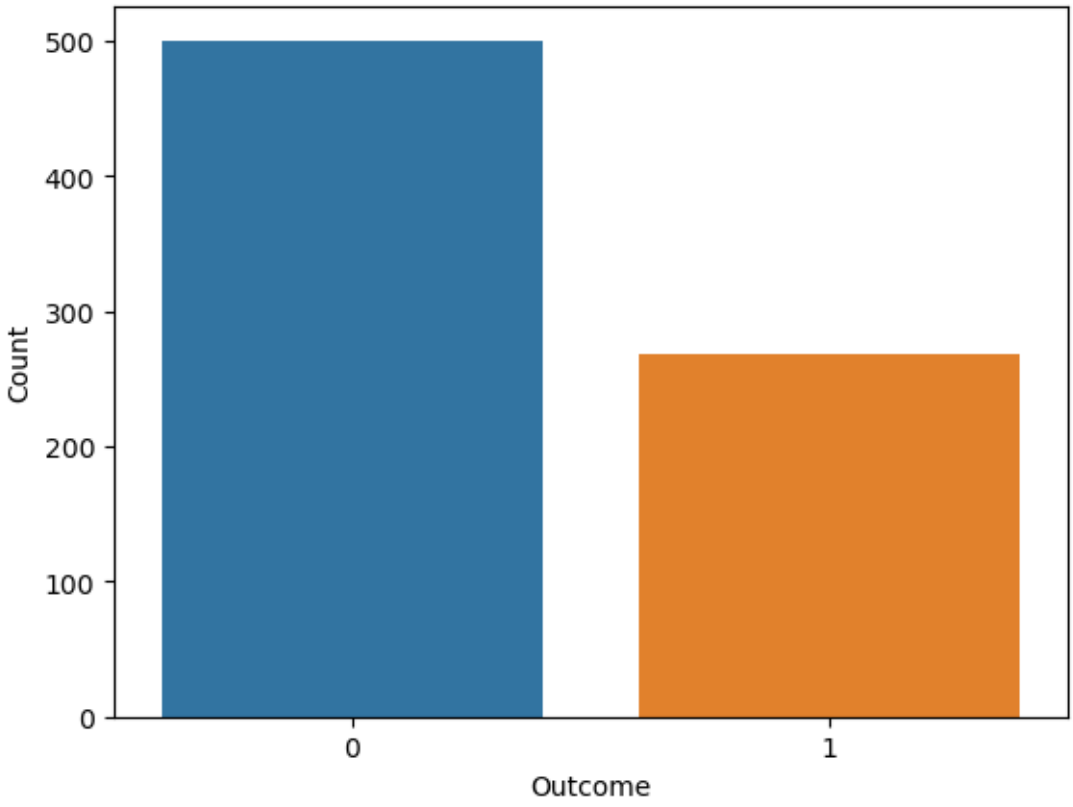
[26]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[30]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x="Outcome", y="Glucose", data=df)
plt.xlabel("Outcome")
plt.ylabel("Glucose")
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



[31]:
```
sns.countplot(x="Outcome", data=df)
plt.xlabel("Outcome")
plt.ylabel("Count")
plt.show()
```



[14]:
```
# Assuming your data is in a DataFrame called 'data'
X = df[["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin",␣
 ↪"BMI", "Pedigree", "Age"]]
y = df["Outcome"]
```

[15]:
```
from sklearn.model_selection import train_test_split
# Split the data into a training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[16]:
```
from sklearn.neighbors import KNeighborsClassifier

# Choose the value of K (number of neighbors)
k = 5
knn = KNeighborsClassifier(n_neighbors=k)
```

[17]:
```
# Fit the KNN model to the training data
knn.fit(X_train, y_train)
```

[17]:
```
KNeighborsClassifier()
```

[18]:
```
# Make predictions on the test data
y_pred = knn.predict(X_test)
```

```
D:\Anaconda\lib\site-packages\sklearn\neighbors\_classification.py:228:
FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the
default behavior of `mode` typically preserves the axis it acts along. In SciPy
1.11.0, this behavior will change: the default value of `keepdims` will become
False, the `axis` over which the statistic is taken will be eliminated, and the
value None will no longer be accepted. Set `keepdims` to True or False to avoid
this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[21]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score

# Compute the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[70 29]
 [23 32]]
```

[22]:
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.66
```

[23]:
```
# Calculate error rate
error_rate = 1 - accuracy
print(f"Error Rate: {error_rate:.2f}")
```

```
Error Rate: 0.34
```

[24]:
```
# Calculate precision
precision = precision_score(y_test, y_pred)
print(f"Precision: {precision:.2f}")
```

```
Precision: 0.52
```

[25]:
```
# Calculate recall
recall = recall_score(y_test, y_pred)
print(f"Recall: {recall:.2f}")
```

```
Recall: 0.58
```

[ ]: