import pandas as pd  
import numpy as np  
import seaborn as sns  
from matplotlib import pyplot as plt

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score,roc\_auc\_score  
from sklearn.model\_selection import GridSearchCV, cross\_validate  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import LabelEncoder  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import StandardScaler

def load():  
 df = pd.read\_csv("datasets/diabetes.csv")  
 return df

df = load()

df.head()

|  | 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 |

df.shape

(768, 9)

df.duplicated()

0 False  
1 False  
2 False  
3 False  
4 False  
 ...   
763 False  
764 False  
765 False  
766 False  
767 False  
Length: 768, dtype: bool

df[df.duplicated]

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

df.isnull().sum()

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

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 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

def outlier\_tresholds(df, col\_name, q1=0.25, q3=0.75):  
 quartile1 = df[col\_name].quantile(q1)  
 quartile3 = df[col\_name].quantile(q3)  
 interquantile\_range = quartile3 - quartile1  
 up\_limit = quartile3 + 1.5 \* interquantile\_range  
 low\_limit = quartile1 - 1.5 \* interquantile\_range  
 return low\_limit, up\_limit

outlier\_tresholds(df, "Age")

(-1.5, 66.5)

low, up = outlier\_tresholds(df, "Age")

df[(df["Age"] < low) | (df["Age"] > up)]

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 123 | 5 | 132 | 80 | 0 | 0 | 26.8 | 0.186 | 69 | 0 |
| 363 | 4 | 146 | 78 | 0 | 0 | 38.5 | 0.520 | 67 | 1 |
| 453 | 2 | 119 | 0 | 0 | 0 | 19.6 | 0.832 | 72 | 0 |
| 459 | 9 | 134 | 74 | 33 | 60 | 25.9 | 0.460 | 81 | 0 |
| 489 | 8 | 194 | 80 | 0 | 0 | 26.1 | 0.551 | 67 | 0 |
| 537 | 0 | 57 | 60 | 0 | 0 | 21.7 | 0.735 | 67 | 0 |
| 666 | 4 | 145 | 82 | 18 | 0 | 32.5 | 0.235 | 70 | 1 |
| 674 | 8 | 91 | 82 | 0 | 0 | 35.6 | 0.587 | 68 | 0 |
| 684 | 5 | 136 | 82 | 0 | 0 | 0.0 | 0.640 | 69 | 0 |

def check\_outlier(df, col):  
 low, up = outlier\_tresholds(df, col)  
 if(df[(df[col] < low) | (df[col] > up)].any(axis = None)):  
 return True  
 else:  
 return False

check\_outlier(df, "Age")

True

[col for col in df.columns if check\_outlier(df, col)]

['Pregnancies',  
 'Glucose',  
 'BloodPressure',  
 'SkinThickness',  
 'Insulin',  
 'BMI',  
 'DiabetesPedigreeFunction',  
 'Age']

def grab\_col\_names(df, cat\_th = 10, car\_th=20):  
 # cat\_cols, cat\_but\_car  
 cat\_cols = [col for col in df.columns if df[col].dtypes == "0"]  
 num\_but\_cat = [col for col in df.columns if df[col].nunique() < 10 and df[col].dtypes != "0"]  
  
 cat\_but\_car = [col for col in df.columns if df[col].dtypes == "0" and   
 df[col].nunique() > car\_th]  
 cat\_cols = cat\_cols + num\_but\_cat  
 cat\_cols = [col for col in cat\_cols if col not in cat\_but\_car]  
  
 # num cols   
 num\_cols = [col for col in df.columns if df[col].dtypes != "0"]  
 num\_cols = [col for col in num\_cols if col not in num\_but\_cat]  
  
 print(f"Observations: {df.shape[0]}")  
 print(f"Variables: {df.shape[1]}")  
 print(f"cat\_cols: {len(cat\_cols)}")  
 print(f"num\_cols: {len(num\_cols)}")  
 print(f"cat\_but\_car: {len(cat\_but\_car)}")  
 print(f"num\_but\_cat: {len(num\_but\_cat)}")  
 return cat\_cols, num\_cols, cat\_but\_car

cat\_cols, num\_cols, cat\_but\_car = grab\_col\_names(df)

Observations: 768  
Variables: 9  
cat\_cols: 1  
num\_cols: 8  
cat\_but\_car: 0  
num\_but\_cat: 1

def cat\_summary(dataframe, col\_name, plot=False):  
 print(pd.DataFrame({col\_name: dataframe[col\_name].value\_counts(),  
 "Ratio": 100 \* dataframe[col\_name].value\_counts() / len(dataframe)}))  
 print("##########################################")  
 if plot:  
 sns.countplot(x=dataframe[col\_name], data=dataframe)  
 plt.show()

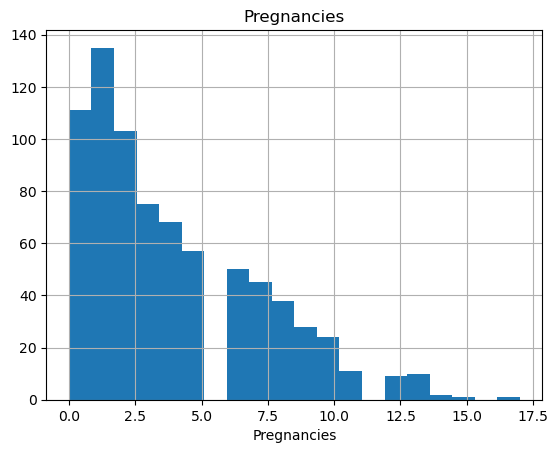
cat\_summary(df, "Outcome")

Outcome Ratio  
Outcome   
0 500 65.104167  
1 268 34.895833  
##########################################

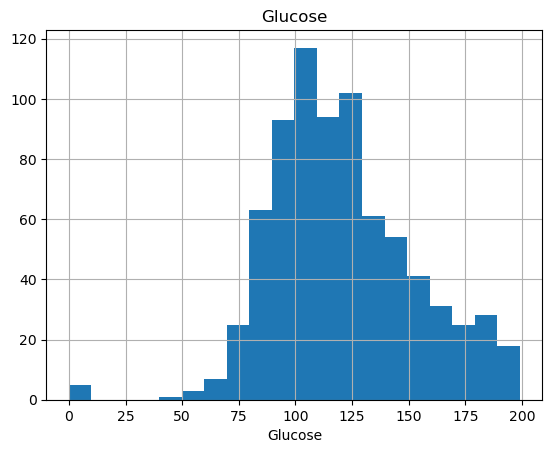
def num\_summary(df, numerical\_col, plot=False):  
 quantiles = [0.05, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 0.95, 0.99]  
 print(df[numerical\_col].describe(quantiles).T)  
  
 if plot:  
 df[numerical\_col].hist(bins=20)  
 plt.xlabel(numerical\_col)  
 plt.title(numerical\_col)  
 plt.show(block=True)

[num\_summary(df, col, True) for col in num\_cols]

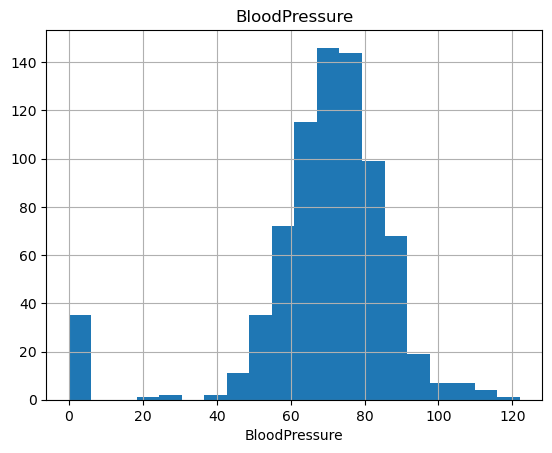
count 768.000000  
mean 3.845052  
std 3.369578  
min 0.000000  
5% 0.000000  
10% 0.000000  
20% 1.000000  
30% 1.000000  
40% 2.000000  
50% 3.000000  
60% 4.000000  
70% 5.000000  
80% 7.000000  
90% 9.000000  
95% 10.000000  
99% 13.000000  
max 17.000000  
Name: Pregnancies, dtype: float64



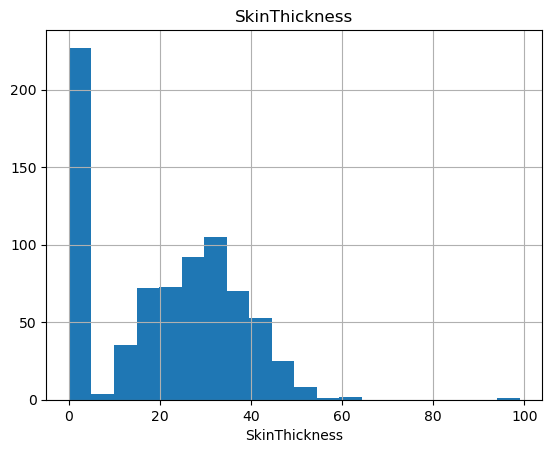
count 768.000000  
mean 120.894531  
std 31.972618  
min 0.000000  
5% 79.000000  
10% 85.000000  
20% 95.000000  
30% 102.000000  
40% 109.000000  
50% 117.000000  
60% 125.000000  
70% 134.000000  
80% 147.000000  
90% 167.000000  
95% 181.000000  
99% 196.000000  
max 199.000000  
Name: Glucose, dtype: float64



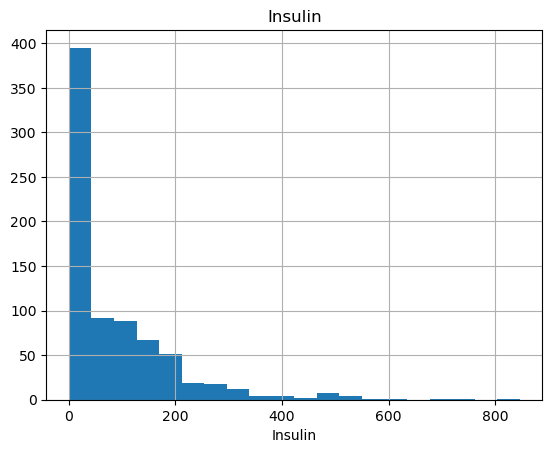
count 768.000000  
mean 69.105469  
std 19.355807  
min 0.000000  
5% 38.700000  
10% 54.000000  
20% 60.000000  
30% 64.000000  
40% 68.000000  
50% 72.000000  
60% 74.000000  
70% 78.000000  
80% 82.000000  
90% 88.000000  
95% 90.000000  
99% 106.000000  
max 122.000000  
Name: BloodPressure, dtype: float64



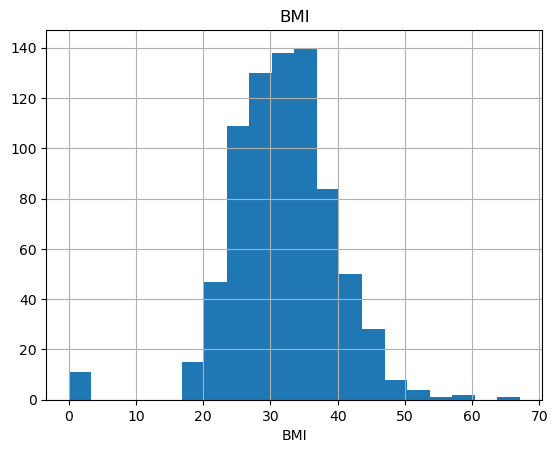
count 768.000000  
mean 20.536458  
std 15.952218  
min 0.000000  
5% 0.000000  
10% 0.000000  
20% 0.000000  
30% 8.200000  
40% 18.000000  
50% 23.000000  
60% 27.000000  
70% 31.000000  
80% 35.000000  
90% 40.000000  
95% 44.000000  
99% 51.330000  
max 99.000000  
Name: SkinThickness, dtype: float64



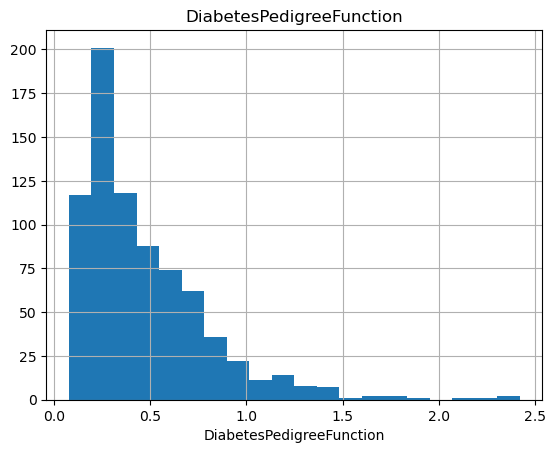
count 768.000000  
mean 79.799479  
std 115.244002  
min 0.000000  
5% 0.000000  
10% 0.000000  
20% 0.000000  
30% 0.000000  
40% 0.000000  
50% 30.500000  
60% 72.200000  
70% 106.000000  
80% 150.000000  
90% 210.000000  
95% 293.000000  
99% 519.900000  
max 846.000000  
Name: Insulin, dtype: float64



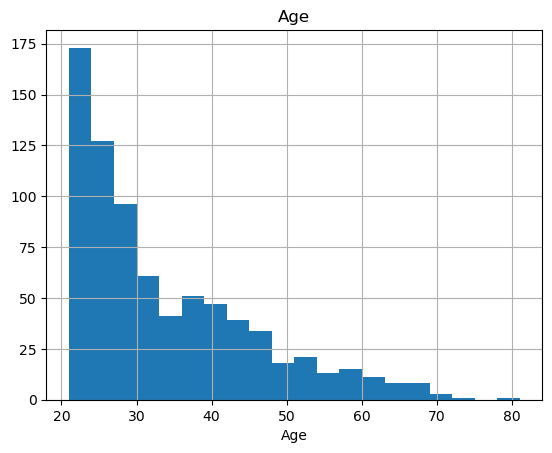
count 768.000000  
mean 31.992578  
std 7.884160  
min 0.000000  
5% 21.800000  
10% 23.600000  
20% 25.900000  
30% 28.200000  
40% 30.100000  
50% 32.000000  
60% 33.700000  
70% 35.490000  
80% 37.800000  
90% 41.500000  
95% 44.395000  
99% 50.759000  
max 67.100000  
Name: BMI, dtype: float64



count 768.000000  
mean 0.471876  
std 0.331329  
min 0.078000  
5% 0.140350  
10% 0.165000  
20% 0.219400  
30% 0.259000  
40% 0.302800  
50% 0.372500  
60% 0.454200  
70% 0.563700  
80% 0.687000  
90% 0.878600  
95% 1.132850  
99% 1.698330  
max 2.420000  
Name: DiabetesPedigreeFunction, dtype: float64



count 768.000000  
mean 33.240885  
std 11.760232  
min 21.000000  
5% 21.000000  
10% 22.000000  
20% 23.000000  
30% 25.000000  
40% 27.000000  
50% 29.000000  
60% 33.000000  
70% 38.000000  
80% 42.600000  
90% 51.000000  
95% 58.000000  
99% 67.000000  
max 81.000000  
Name: Age, dtype: float64



num\_cols

['Pregnancies',  
 'Glucose',  
 'BloodPressure',  
 'SkinThickness',  
 'Insulin',  
 'BMI',  
 'DiabetesPedigreeFunction',  
 'Age']

for col in num\_cols:  
 print(col, check\_outlier(df, col))

Pregnancies True  
Glucose True  
BloodPressure True  
SkinThickness True  
Insulin True  
BMI True  
DiabetesPedigreeFunction True  
Age True

def target\_summary\_with\_num(dataframe, target, numerical\_col):  
 print(dataframe.groupby(target).agg({numerical\_col: "mean"}), end="\n\n\n")

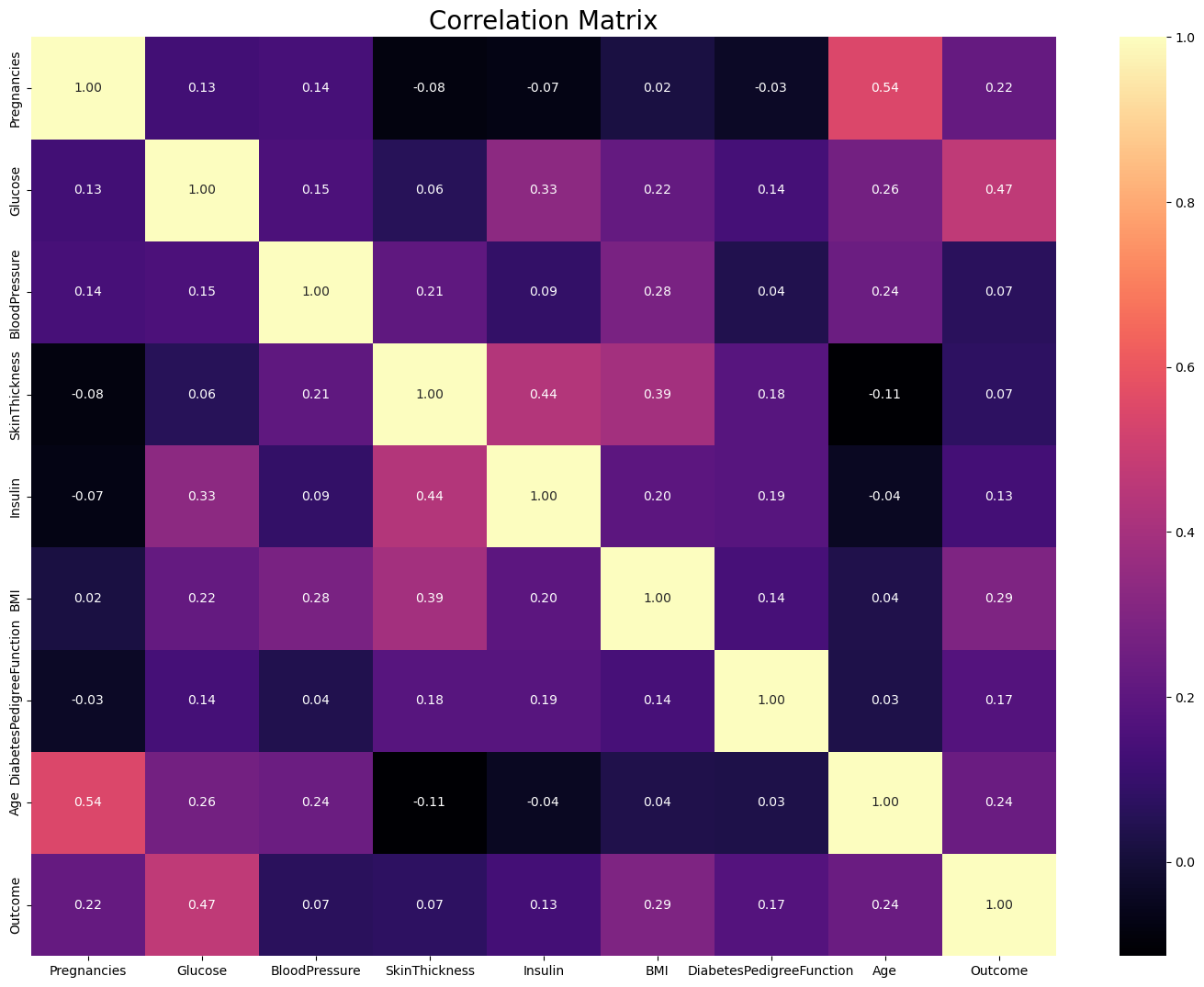
for col in num\_cols:  
 target\_summary\_with\_num(df, "Outcome", col)

Pregnancies  
Outcome   
0 3.298000  
1 4.865672  
  
  
 Glucose  
Outcome   
0 109.980000  
1 141.257463  
  
  
 BloodPressure  
Outcome   
0 68.184000  
1 70.824627  
  
  
 SkinThickness  
Outcome   
0 19.664000  
1 22.164179  
  
  
 Insulin  
Outcome   
0 68.792000  
1 100.335821  
  
  
 BMI  
Outcome   
0 30.304200  
1 35.142537  
  
  
 DiabetesPedigreeFunction  
Outcome   
0 0.429734  
1 0.550500  
  
  
 Age  
Outcome   
0 31.190000  
1 37.067164

df.corr()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 | 0.263514 | 0.466581 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238356 | 1.000000 |

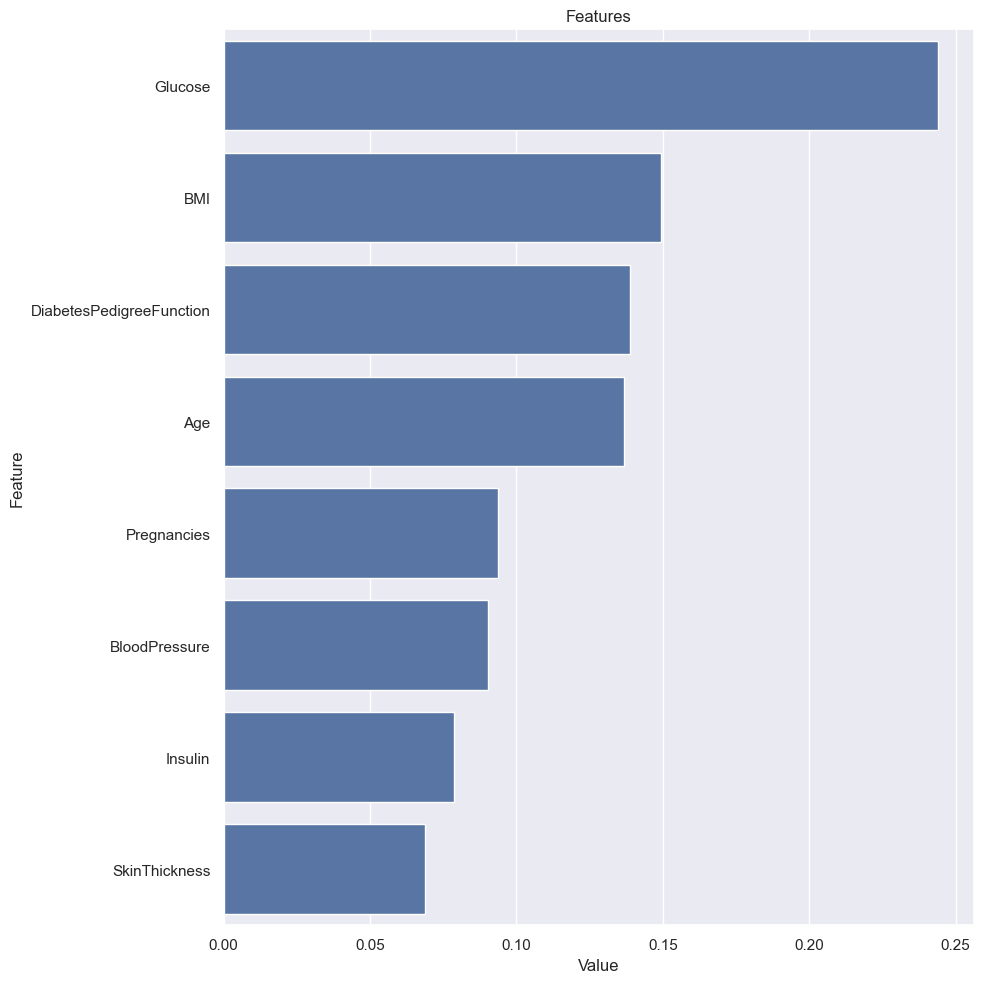
f, ax = plt.subplots(figsize=[18, 13])  
sns.heatmap(df.corr(), annot=True, fmt=".2f", ax=ax, cmap="magma")  
ax.set\_title("Correlation Matrix", fontsize=20)  
plt.show()



y = df["Outcome"]  
X = df.drop("Outcome", axis=1)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=17)  
  
rf\_model = RandomForestClassifier(random\_state=46).fit(X\_train, y\_train)  
y\_pred = rf\_model.predict(X\_test)  
  
print(f"Accuracy: {round(accuracy\_score(y\_pred, y\_test), 2)}")  
print(f"Recall: {round(recall\_score(y\_pred,y\_test),3)}")  
print(f"Precision: {round(precision\_score(y\_pred,y\_test), 2)}")  
print(f"F1: {round(f1\_score(y\_pred,y\_test), 2)}")  
print(f"Auc: {round(roc\_auc\_score(y\_pred,y\_test), 2)}")

Accuracy: 0.77  
Recall: 0.706  
Precision: 0.59  
F1: 0.64  
Auc: 0.75

def plot\_importance(model, features, num=len(X), save=False):  
 feature\_imp = pd.DataFrame({'Value': model.feature\_importances\_, 'Feature': features.columns})  
 plt.figure(figsize=(10, 10))  
 sns.set(font\_scale=1)  
 sns.barplot(x="Value", y="Feature", data=feature\_imp.sort\_values(by="Value",  
 ascending=False)[0:num])  
 plt.title('Features')  
 plt.tight\_layout()  
 plt.show()  
 if save:  
 plt.savefig('importances.png')  
  
plot\_importance(rf\_model, X)



df[df["Glucose"] == 0]["Glucose"]

75 0  
182 0  
342 0  
349 0  
502 0  
Name: Glucose, dtype: int64

df["Glucose"] = df["Glucose"].replace(0, np.nan)

df.head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 6 | 148.0 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85.0 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183.0 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89.0 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137.0 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

df=load()

df.head()

|  | 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 |

df["Glucose"] = df["Glucose"].replace(0, np.nan)

df.head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 6 | 148.0 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85.0 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183.0 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89.0 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137.0 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

df.head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 6 | 148.0 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85.0 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183.0 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89.0 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137.0 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

df[(df["Glucose"].isnull()) & (df["Outcome"] == 0)].head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 75 | 1 | NaN | 48 | 20 | 0 | 24.7 | 0.140 | 22 | 0 |
| 182 | 1 | NaN | 74 | 20 | 23 | 27.7 | 0.299 | 21 | 0 |
| 342 | 1 | NaN | 68 | 35 | 0 | 32.0 | 0.389 | 22 | 0 |

df.describe()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 768.000000 | 763.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 3.845052 | 121.686763 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 3.369578 | 30.535641 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| min | 0.000000 | 44.000000 | 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 | 141.000000 | 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 |

age\_filtered\_mean = df[(df["Age"] > 20) & (df["Age"] < 30)]["Glucose"].mean()

df.loc[(df["Glucose"].isnull()) & (df["Outcome"] == 0), "Glucose"] = age\_filtered\_mean

df[(df["Glucose"].isnull())].head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 349 | 5 | NaN | 80 | 32 | 0 | 41.0 | 0.346 | 37 | 1 |
| 502 | 6 | NaN | 68 | 41 | 0 | 39.0 | 0.727 | 41 | 1 |

y = df["Outcome"]  
X = df.drop("Outcome", axis=1)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=17)  
  
rf\_model = RandomForestClassifier(random\_state=46).fit(X\_train, y\_train)  
y\_pred = rf\_model.predict(X\_test)  
  
  
print(f"Accuracy: {round(accuracy\_score(y\_pred, y\_test), 2)}")  
print(f"Recall: {round(recall\_score(y\_pred,y\_test),3)}")  
print(f"Precision: {round(precision\_score(y\_pred,y\_test), 2)}")  
print(f"F1: {round(f1\_score(y\_pred,y\_test), 2)}")  
print(f"Auc: {round(roc\_auc\_score(y\_pred,y\_test), 2)}")

Accuracy: 0.76  
Recall: 0.691  
Precision: 0.58  
F1: 0.63  
Auc: 0.74

age\_filtered\_mean = df[(df["Age"] > 30) & (df["Age"] < 40)]["Glucose"].mean()

df.loc[(df["Glucose"].isnull()) & (df["Outcome"] == 1), "Glucose"] = age\_filtered\_mean

df[(df["Glucose"].isnull())].head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

df.head()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 6 | 148.0 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85.0 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183.0 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89.0 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137.0 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

df.groupby("Pregnancies").agg({"Outcome": "count"})

|  | Outcome |
| --- | --- |
| Pregnancies |  |
| 0 | 111 |
| 1 | 135 |
| 2 | 103 |
| 3 | 75 |
| 4 | 68 |
| 5 | 57 |
| 6 | 50 |
| 7 | 45 |
| 8 | 38 |
| 9 | 28 |
| 10 | 24 |
| 11 | 11 |
| 12 | 9 |
| 13 | 10 |
| 14 | 2 |
| 15 | 1 |
| 17 | 1 |

df.groupby("Glucose").agg({"Outcome": "count"})

|  | Outcome |
| --- | --- |
| Glucose |  |
| 44.0 | 1 |
| 56.0 | 1 |
| 57.0 | 2 |
| 61.0 | 1 |
| 62.0 | 1 |
| ... | ... |
| 195.0 | 2 |
| 196.0 | 3 |
| 197.0 | 4 |
| 198.0 | 1 |
| 199.0 | 1 |

df.corr()

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 1.000000 | 0.128869 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.128869 | 1.000000 | 0.158622 | 0.075190 | 0.331309 | 0.236008 | 0.137660 | 0.267598 | 0.494102 |
| BloodPressure | 0.141282 | 0.158622 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |
| SkinThickness | -0.081672 | 0.075190 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331309 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.236008 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | -0.033523 | 0.137660 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.267598 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.494102 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238356 | 1.000000 |

plt.figure(figsize=(10, 6))  
sns.barplot(data=df, x="Glucose", y="Outcome", estimator=sum, ci=None)  
  
# Grafik düzenleme  
plt.title("Glucose Değerlerine Göre Şeker Hastalığı Durumu", fontsize=14)  
plt.xlabel("Glucose Değerleri", fontsize=12)  
plt.ylabel("Şeker Hastası Olan Kişi Sayısı", fontsize=12)  
plt.xticks(rotation=45)  
plt.show()

/var/folders/st/g578wh254q58gyrbtjhdkz380000gn/T/ipykernel\_4165/1172269381.py:2: FutureWarning:   
  
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.  
  
 sns.barplot(data=df, x="Glucose", y="Outcome", estimator=sum, ci=None)

