In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
diabetes_df=pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\ML\Amit Mi
```

In [3]:
```
diabetes_df
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

Problem Statement: Diabetes classification based on Ensemble Learning

- Since it is supervised learning that is why output is given

Data Wrangling

In [4]:
```
diabetes_df.isnull().sum()
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [5]:
```
diabetes_df.shape
```

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

In [6]:
```
diabetes_df.describe()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

- From above data - Bp, skin thickness is 0 not possible, means missing data is replaced by 0

In [7]:
```
diabetes = diabetes_df.drop(["Pregnancies","Outcome"],axis=1)
```

In [8]:
```
diabetes.replace(0, np.nan, inplace=True)
```

In [9]:
```
diabetes.isnull().sum()
```

Out[9]:
```
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
dtype: int64
```

Replace missing values with Central Tendencies - Mean, Median and Mode

- We will do here in simple imputer method
- We can replace the missing values with mean, median and mode- with the help of function replace, fillna

In [10]:
```
diabetes["Insulin"].unique()
```

Out[10]:
```
array([ nan,  94., 168.,  88., 543., 846., 175., 230.,  83.,  96., 235.,
       146., 115., 140., 110., 245.,  54., 192., 207.,  70., 240.,  82.,
        36.,  23., 300., 342., 304., 142., 128.,  38., 100.,  90., 270.,
        71., 125., 176.,  48.,  64., 228.,  76., 220.,  40., 152.,  18.,
       135., 495.,  37.,  51.,  99., 145., 225.,  49.,  50.,  92., 325.,
        63., 284., 119., 204., 155., 485.,  53., 114., 105., 285., 156.,
        78., 130.,  55.,  58., 160., 210., 318.,  44., 190., 280.,  87.,
       271., 129., 120., 478.,  56.,  32., 744., 370.,  45., 194., 680.,
       402., 258., 375., 150.,  67.,  57., 116., 278., 122., 545.,  75.,
        74., 182., 360., 215., 184.,  42., 132., 148., 180., 205.,  85.,
       231.,  29.,  68.,  52., 255., 171.,  73., 108.,  43., 167., 249.,
       293.,  66., 465.,  89., 158.,  84.,  72.,  59.,  81., 196., 415.,
       275., 165., 579., 310.,  61., 474., 170., 277.,  60.,  14.,  95.,
       237., 191., 328., 250., 480., 265., 193.,  79.,  86., 326., 188.,
       106.,  65., 166., 274.,  77., 126., 330., 600., 185.,  25.,  41.,
       272., 321., 144.,  15., 183.,  91.,  46., 440., 159., 540., 200.,
       335., 387.,  22., 291., 392., 178., 127., 510.,  16., 112.])
```

In [11]:
```
np.round(diabetes["Insulin"].mean())
# we didn't use simple imputation because the diffrence btwn Mean and Median is ver
```

Out[11]:
```
156.0
```

In [12]:
```
diabetes["Insulin"].replace(np.nan,np.round(diabetes["Insulin"].mean()),inplace=Tru
```

Imputation - Simple Imputer

- Where to implement simpe imputer strategy - when you have got the lot of columns missing and you want to replace with either mean, median or mode
- Two options either you do manually or you just do simple imputer

In [13]:
```
from sklearn.impute import SimpleImputer
# strategy = mean, median, most-frequent
imputer = SimpleImputer (strategy = "median")
```

In [14]:
```
X_data=imputer.fit_transform(diabetes)
X_data
```

Out[14]:
```
array([[148.   ,  72.   ,  35.   , ...,  33.6  ,   0.627,  50.   ],
       [ 85.   ,  66.   ,  29.   , ...,  26.6  ,   0.351,  31.   ],
       [183.   ,  64.   ,  29.   , ...,  23.3  ,   0.672,  32.   ],
       ...,
       [121.   ,  72.   ,  23.   , ...,  26.2  ,   0.245,  30.   ],
       [126.   ,  60.   ,  29.   , ...,  30.1  ,   0.349,  47.   ],
       [ 93.   ,  70.   ,  31.   , ...,  30.4  ,   0.315,  23.   ]])
```

In [15]:
```
diabetes_df2= pd.DataFrame(X_data, columns = diabetes.columns)
```

In [16]:
```
diabetes_df2
```

Out[16]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	148.0	72.0	35.0	156.0	33.6	0.627	50.0
1	85.0	66.0	29.0	156.0	26.6	0.351	31.0
2	183.0	64.0	29.0	156.0	23.3	0.672	32.0
3	89.0	66.0	23.0	94.0	28.1	0.167	21.0
4	137.0	40.0	35.0	168.0	43.1	2.288	33.0
...	...	...	...	...	...	...	...
763	101.0	76.0	48.0	180.0	32.9	0.171	63.0
764	122.0	70.0	27.0	156.0	36.8	0.340	27.0
765	121.0	72.0	23.0	112.0	26.2	0.245	30.0
766	126.0	60.0	29.0	156.0	30.1	0.349	47.0
767	93.0	70.0	31.0	156.0	30.4	0.315	23.0

768 rows × 7 columns

In [17]:

```
# now we had to add rest of two columns to our existing df
diabetes_df2['Pregnancies'] = diabetes_df.Pregnancies
diabetes_df2['Outcome'] = diabetes_df.Outcome
```

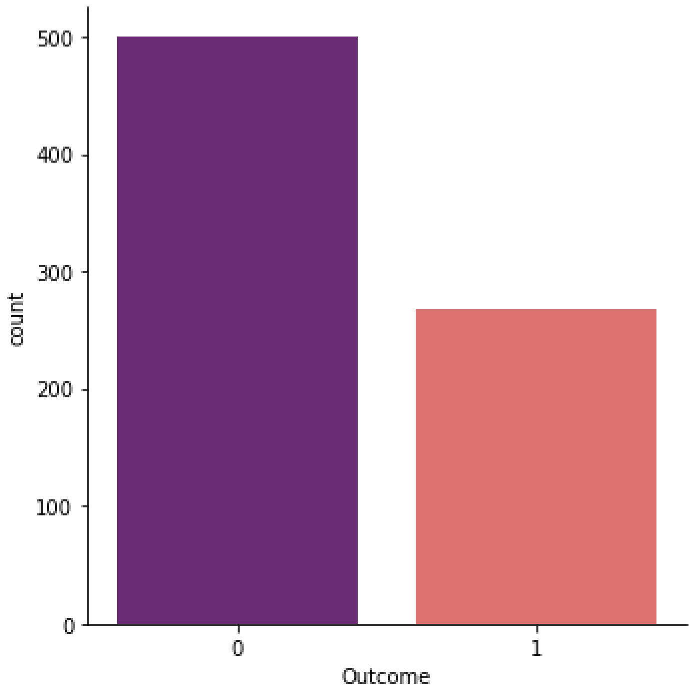
Output Distribution

In [18]:

```
sns.catplot(x="Outcome", kind="count", data=diabetes_df2, palette= "magma")
#sns.countplot(x="Outcome", data=diabetes_df2, palette= "magma")
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x253e6589670>
```



- It is showing imblanced data - One thing we can do we can increase the value dataset

In [19]:
```
diabetes_df2["Outcome"].value_counts()
```

Out[19]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

Cross Validation - Using K model

- We had already done cross validation by using train and test

K-FOLD can be used for other algorithms as well, not exclusively used for these 3 itlsef, similar like train-test split

In [20]:
```
X = diabetes_df2.drop(["Outcome"], axis= 1)
Y = diabetes_df2["Outcome"]
```

In [21]:
```
from sklearn.model_selection import KFold
#n_splits also called as Fold = 10, represented by value K, that is why it is calle
kfold = KFold(n_splits=10, random_state=7, shuffle=True)
```

Adabost Classifier

In [22]:
```
from sklearn.ensemble import AdaBoostClassifier
# n_estimator = No of decison trees, max_iter = Number of Iterations
AB_model = AdaBoostClassifier(n_estimators=30, random_state=7)
```

In [23]:
```
from sklearn.model_selection import cross_val_score
# 1st train your model, and give 10 diffrent train_test combination for diffrent sp
# # cross_val_score predict the o/p using AB_model. Algo AB_model will fit with X
results = cross_val_score(AB_model,X,Y, cv =kfold)
```

In [24]:
```
results.mean()
```

Out[24]:
```
0.7552631578947369
```

Gradient Boosting Classifier

In [25]:
```
from sklearn.ensemble import GradientBoostingClassifier
GB_model=GradientBoostingClassifier(n_estimators = 120, random_state = 7)
```

In [26]:
```
# Prediction
results = cross_val_score(GB_model, X, Y, cv = kfold)
```

In [27]:
```
results.mean()
```

Out[27]:
```
0.7708988380041012
```

- K-fold cross validation-These methods are used so that we will get to see average performance of the algorithm for all the dataset

XGboost - Extreme Gradient Boosting

In [32]:
```
!pip install xgboost
from xgboost import XGBClassifier
XG_model=XGBClassifier()
```

```
Collecting xgboost
  Downloading xgboost-1.7.2-py3-none-win_amd64.whl (89.1 MB)
Requirement already satisfied: numpy in c:\users\s323\anaconda3\lib\site-packages
(from xgboost) (1.21.5)
Requirement already satisfied: scipy in c:\users\s323\anaconda3\lib\site-packages
(from xgboost) (1.7.3)
Installing collected packages: xgboost
Successfully installed xgboost-1.7.2
```

In [34]:
```
results = cross_val_score(XG_model, X, Y, cv = kfold)
```

In [35]:
```
results.mean()
```

Out[35]:
```
0.7356288448393712
```

Make Prediction

In [43]:
```
results_gb = cross_val_score(GB_model, X, Y, cv = kfold)
```

In [44]:
```
results_gb.mean()
```

Out[44]:
```
0.7708988380041012
```

In [45]:
```
from sklearn.model_selection import cross_val_predict
```

In [46]:
```
y_predict = cross_val_predict(GB_model, X, Y, cv = kfold)
```

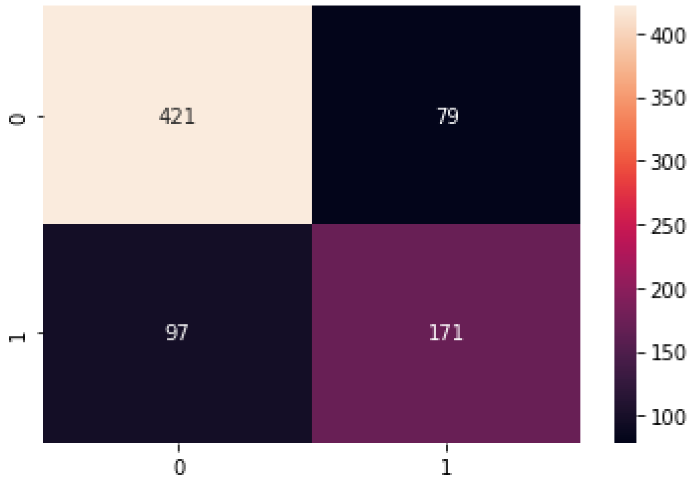
In [47]:
```
from sklearn.metrics import confusion_matrix
```

In [50]:
```
confusion_matrix(Y, y_predict )
```

Out[50]:
```
array([[421,  79],
       [ 97, 171]], dtype=int64)
```

In [53]:
```
sns.heatmap(confusion_matrix(Y, y_predict), annot = True, fmt='0.0f')
```

Out[53]:
```
<AxesSubplot:>
```



```
In [ ]:
```