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

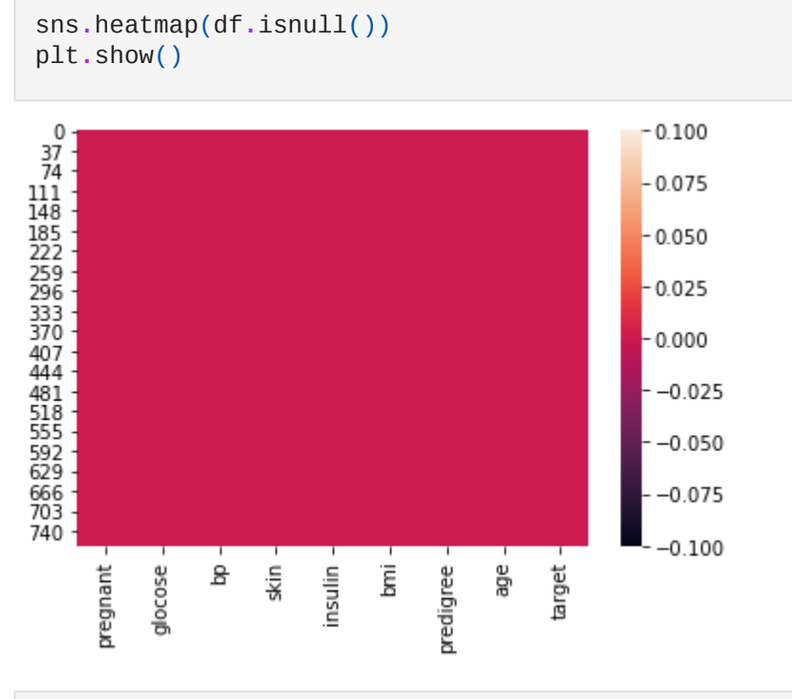
In [2]:
```python
df=pd.read_csv(r'C:\Users\sharma17\Downloads\ml\classification dataset\diabeties.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	pregnant	glocose	bp	skin	insulin	bmi	predigree	age	target
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

In [4]:
```python
sns.heatmap(df.isnull())
plt.show()
```



In [5]:
```python
df.describe()
```

Out[5]:

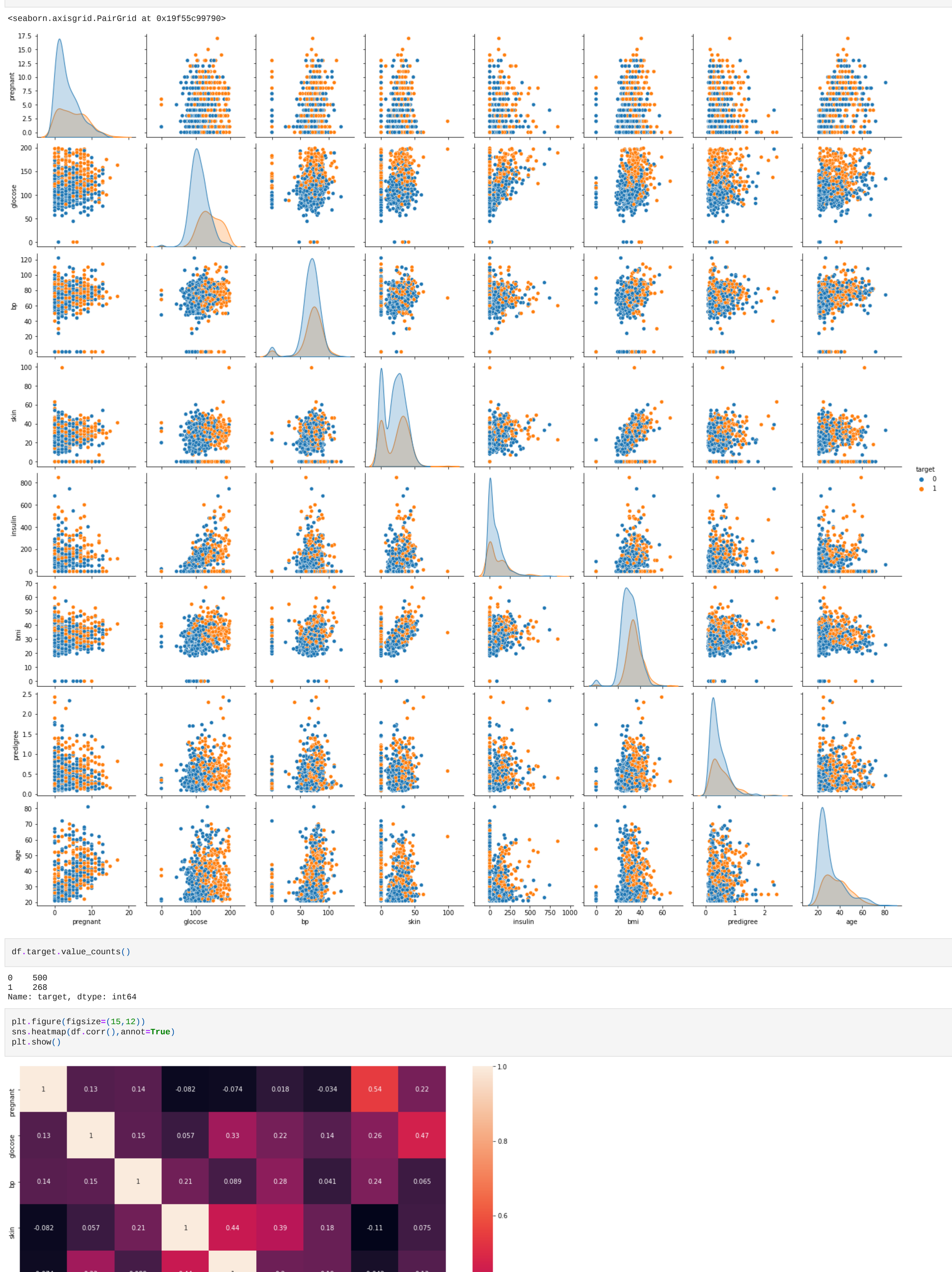
	pregnant	glocose	bp	skin	insulin	bmi	predigree	age	target
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	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	140.250000	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

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pregnant   768 non-null    int64
 1   glocose    768 non-null    int64
 2   bp         768 non-null    int64
 3   skin       768 non-null    int64
 4   insulin    768 non-null    int64
 5   bmi        768 non-null    float64
 6   predigree  768 non-null    float64
 7   age        768 non-null    int64
 8   target     768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [7]:
```python
sns.pairplot(df,hue='target')
```

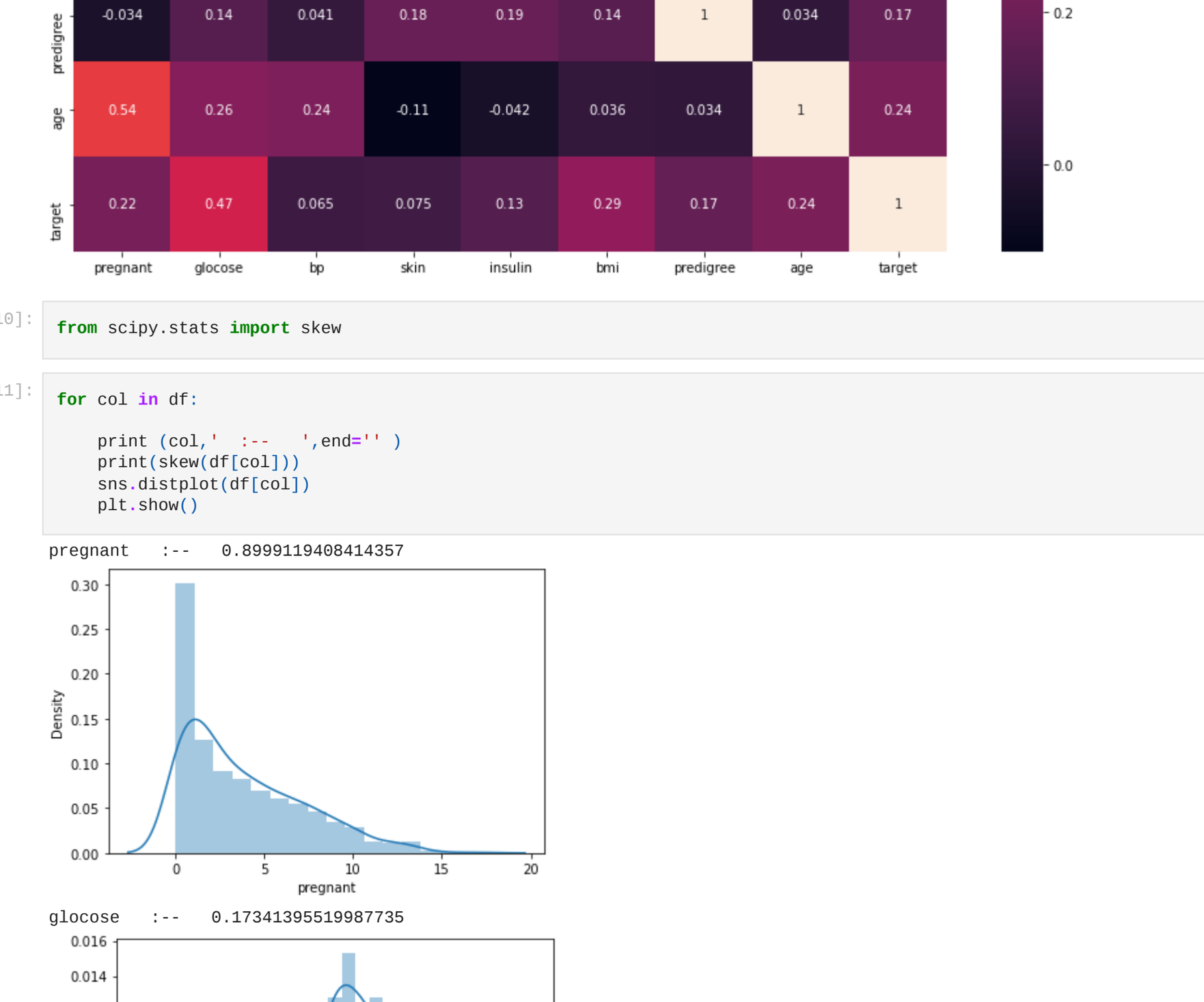
Out[7]: <seaborn.axisgrid.PairGrid at 0x19f55c99790>



In [8]:
```python
df.target.value_counts()
```

Out[8]:
```
0    500
1    268
Name: target, dtype: int64
```

In [9]:
```python
plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),annot=True)
plt.show()
```

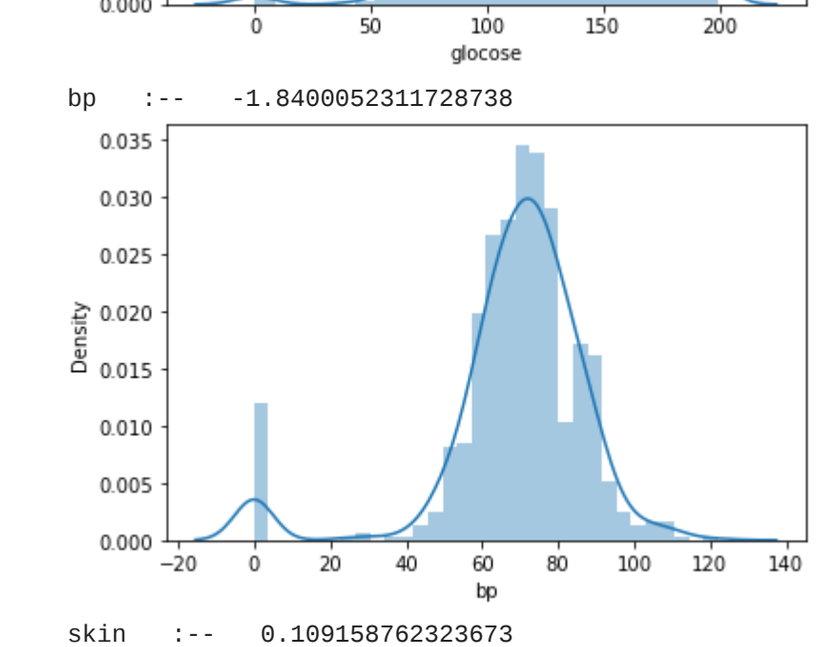


In [10]:
```python
from scipy.stats import skew
```

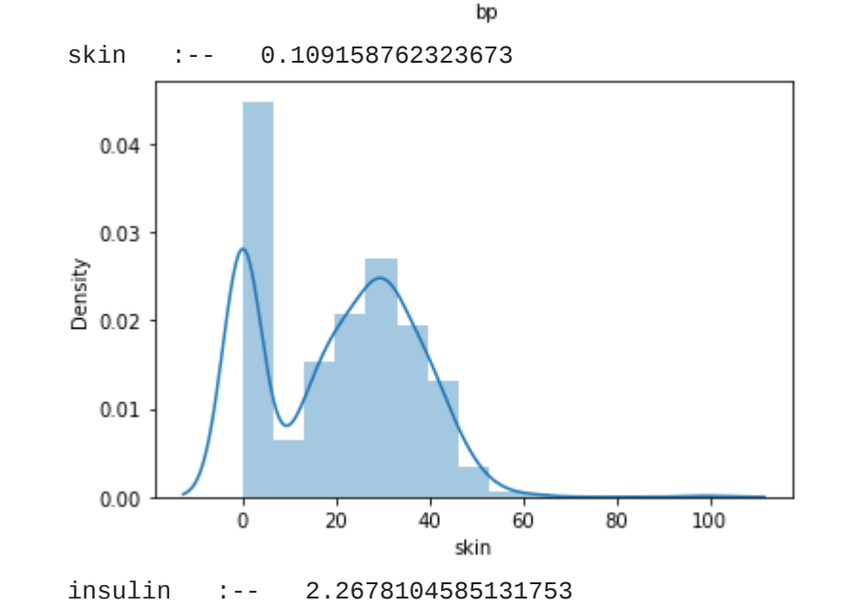
In [11]:
```python
for col in df:

    print (col,'  :--   ',end='' )
    print(skew(df[col]))
    sns.distplot(df[col])
    plt.show()
```

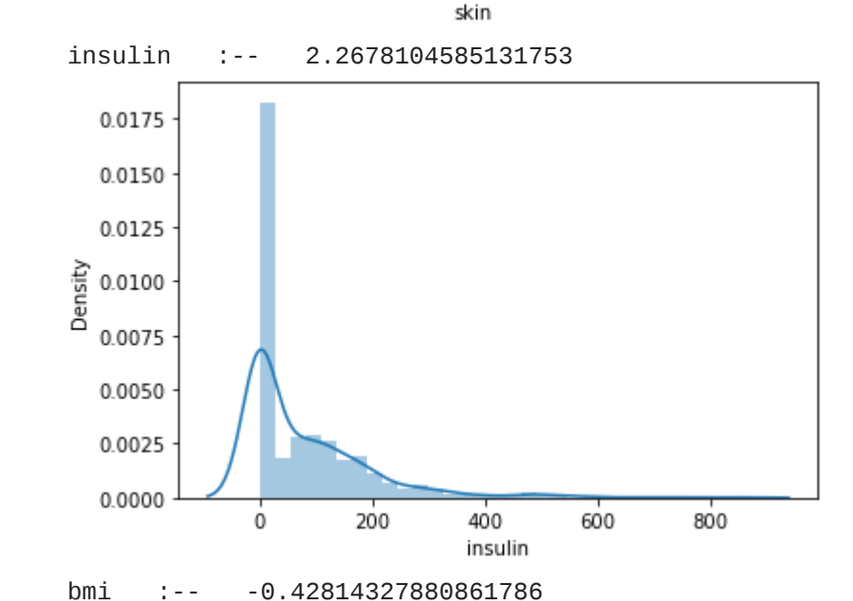
pregnant   :--   0.8999119408414357



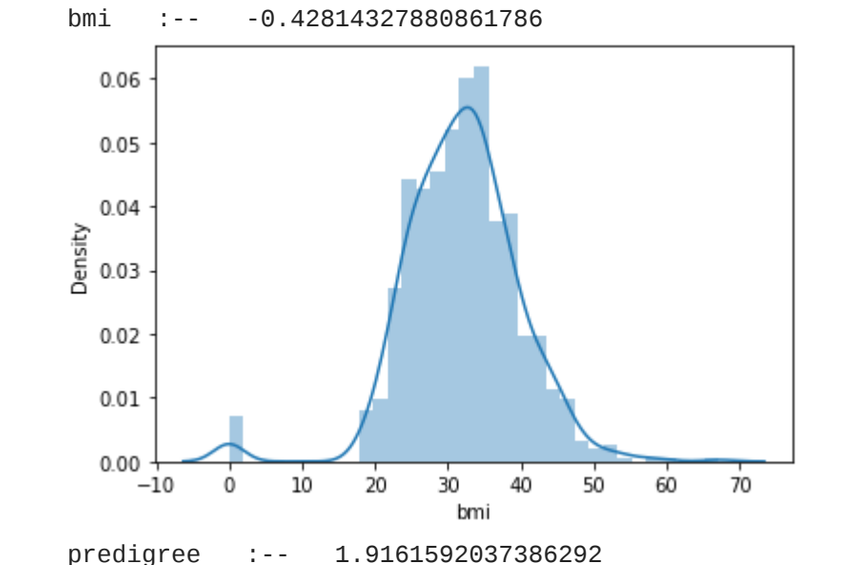
glocose   :--   0.17341395519987735



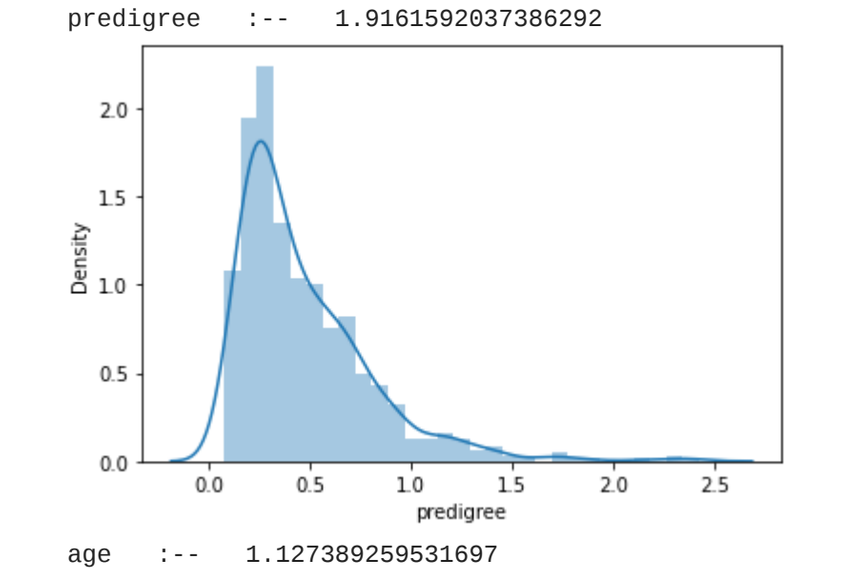
bp   :--   -1.8400052311728738



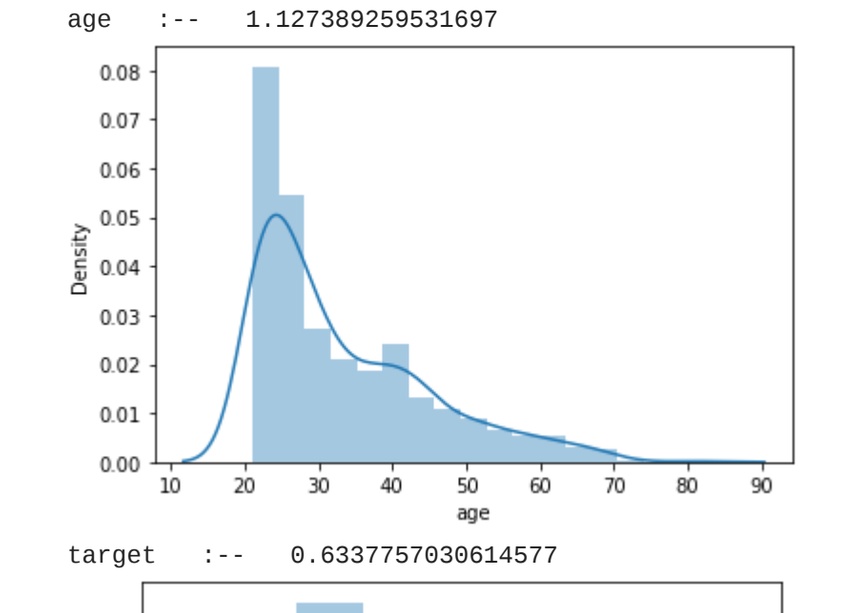
skin   :--   0.109158762323673



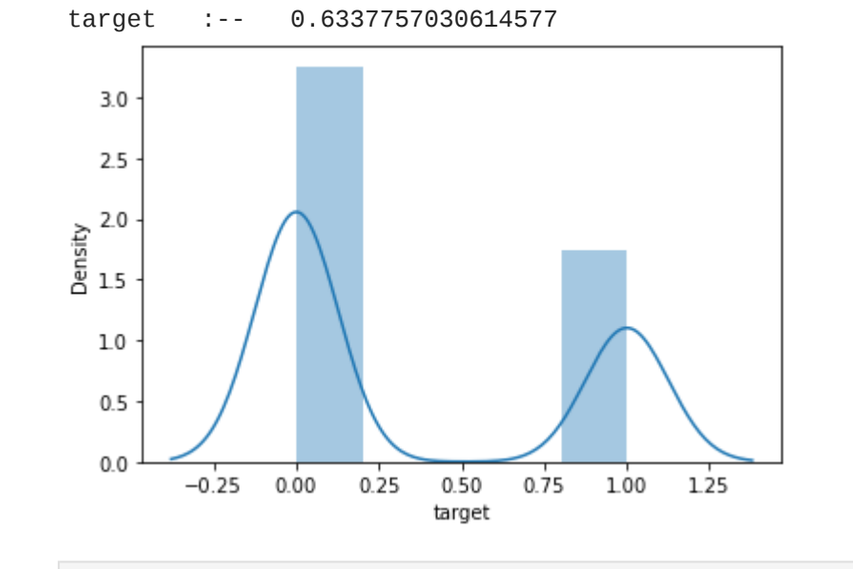
insulin   :--   2.2678104585131753



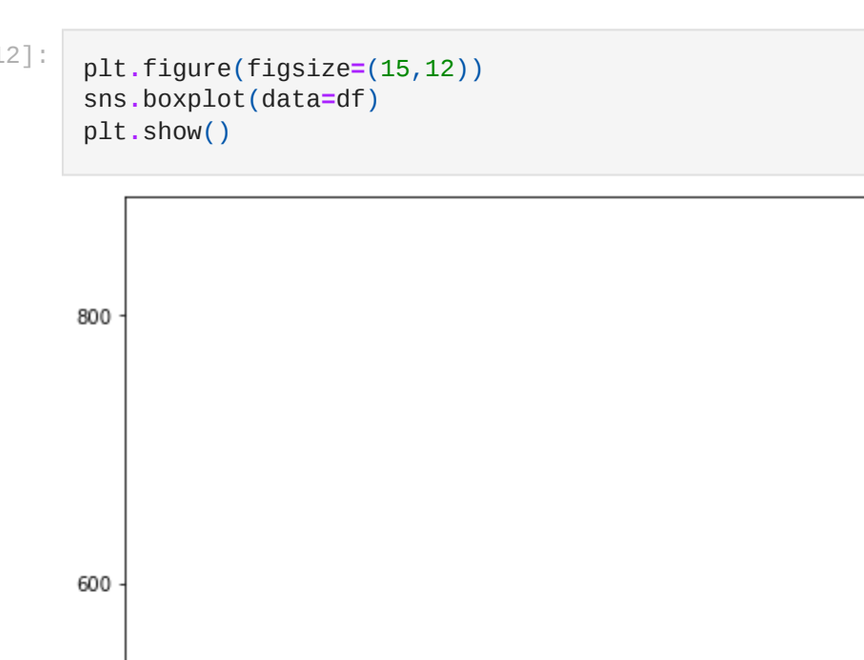
bmi   :--   -0.42814327880861786



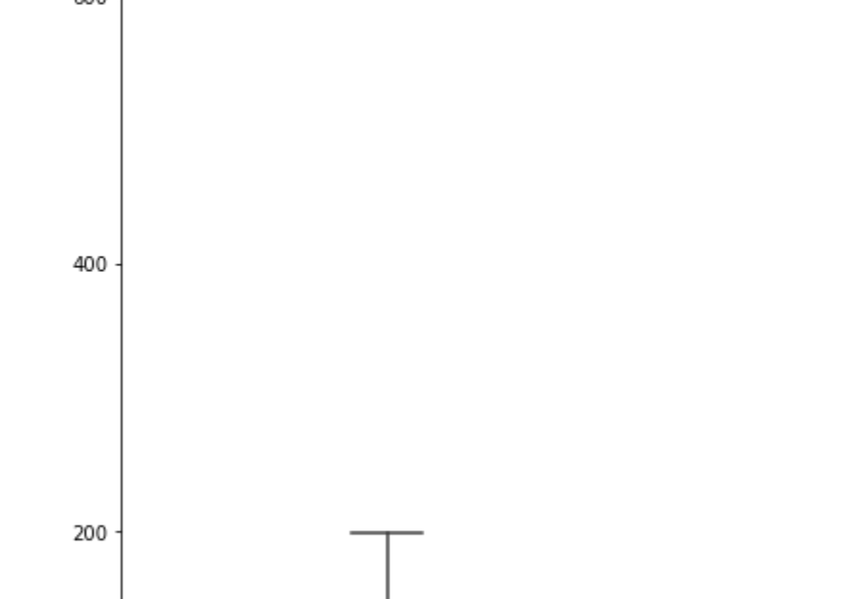
predigree   :--   1.9161592037386292



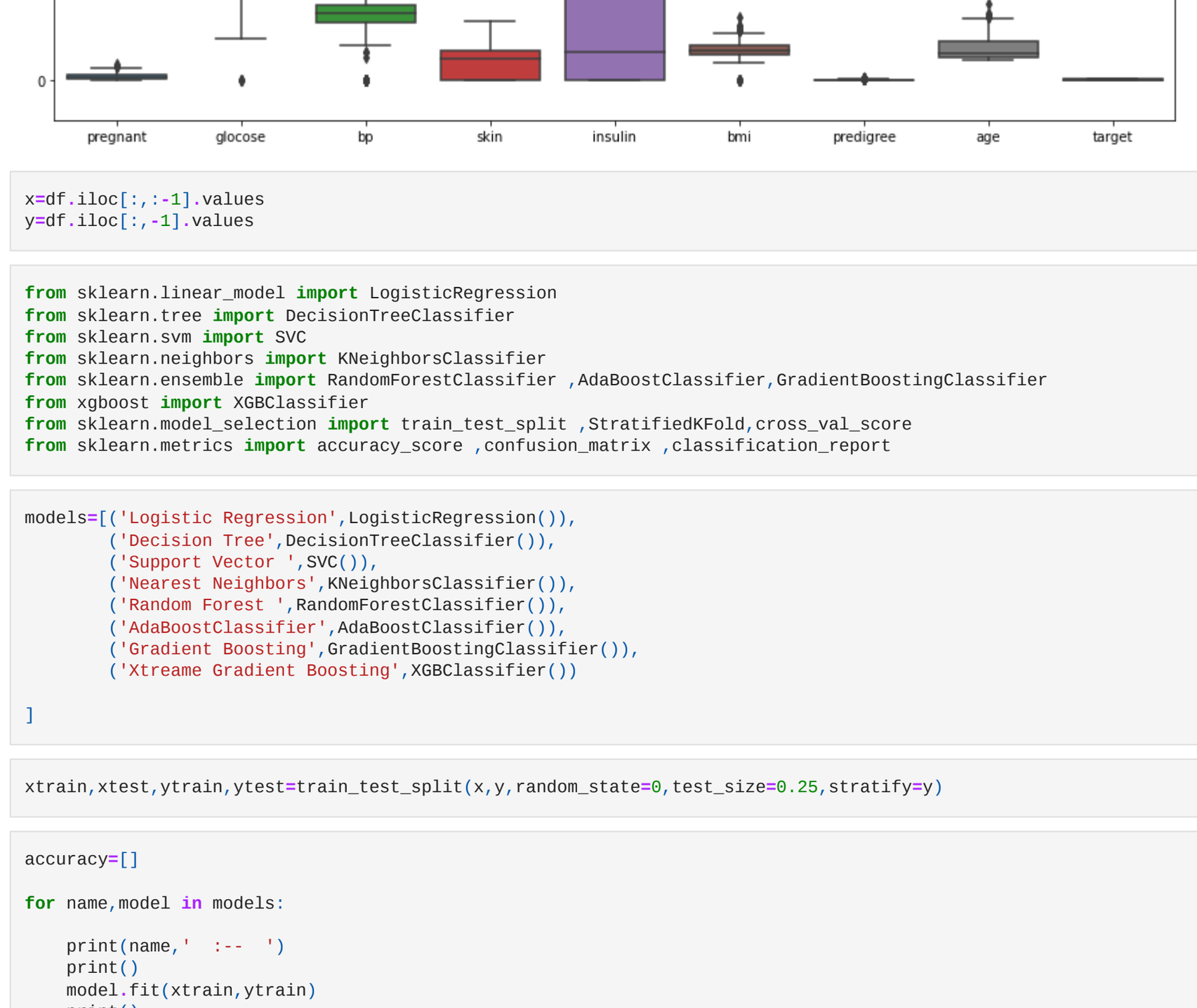
age   :--   1.127389259531697



target   :--   0.6337757030614577



In [12]:
```python
plt.figure(figsize=(15,12))
sns.boxplot(data=df)
plt.show()
```



In [13]:
```python
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [14]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier ,AdaBoostClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split ,StratifiedKFold,cross_val_score
from sklearn.metrics import accuracy_score ,confusion_matrix ,classification_report
```

In [15]:
```python
models=[('Logistic Regression',LogisticRegression()),
        ('Decision Tree',DecisionTreeClassifier()),
        ('Support Vector ',SVC()),
        ('Nearest Neighbors',KNeighborsClassifier()),
        ('Random Forest ',RandomForestClassifier()),
        ('AdaBoostClassifier',AdaBoostClassifier()),
        ('Gradient Boosting',GradientBoostingClassifier()),
        ('Xtreame Gradient Boosting',XGBClassifier())

]
```

In [16]:
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0,test_size=0.25,stratify=y)
```

In [17]:
```python
accuracy=[]

for name,model in models:

    print(name,'  :--  ')
    print()
    model.fit(xtrain,ytrain)
    print()
    ypred=model.predict(xtest)
    print(confusion_matrix(ytest,ypred))
    print()

    print(classification_report(ytest,ypred))
    accuracy.append(accuracy_score(ytest,ypred))
```

```
Logistic Regression   :--


[[110  15]
 [ 27  40]]

              precision    recall  f1-score   support

           0       0.80      0.88      0.84       125
           1       0.73      0.60      0.66        67

    accuracy                           0.78       192
   macro avg       0.77      0.74      0.75       192
weighted avg       0.78      0.78      0.78       192

Decision Tree   :--


[[104  21]
 [ 27  40]]

              precision    recall  f1-score   support

           0       0.79      0.83      0.81       125
           1       0.66      0.60      0.62        67

    accuracy                           0.75       192
   macro avg       0.72      0.71      0.72       192
weighted avg       0.75      0.75      0.75       192

Support Vector    :--


[[113  12]
 [ 33  34]]

              precision    recall  f1-score   support

           0       0.77      0.90      0.83       125
           1       0.74      0.51      0.60        67

    accuracy                           0.77       192
   macro avg       0.76      0.71      0.72       192
weighted avg       0.76      0.77      0.75       192

Nearest Neighbors   :--


[[104  21]
 [ 26  41]]

              precision    recall  f1-score   support

           0       0.80      0.83      0.82       125
           1       0.66      0.61      0.64        67

    accuracy                           0.76       192
   macro avg       0.73      0.72      0.73       192
weighted avg       0.75      0.76      0.75       192

Random Forest    :--


[[110  15]
 [ 28  39]]

              precision    recall  f1-score   support

           0       0.80      0.88      0.84       125
           1       0.72      0.58      0.64        67

    accuracy                           0.78       192
   macro avg       0.76      0.73      0.74       192
weighted avg       0.77      0.78      0.77       192

AdaBoostClassifier   :--


[[108  17]
 [ 28  39]]

              precision    recall  f1-score   support

           0       0.79      0.86      0.83       125
           1       0.70      0.58      0.63        67

    accuracy                           0.77       192
   macro avg       0.75      0.72      0.73       192
weighted avg       0.76      0.77      0.76       192

Gradient Boosting   :--


[[108  17]
 [ 22  45]]

              precision    recall  f1-score   support

           0       0.83      0.86      0.85       125
           1       0.73      0.67      0.70        67

    accuracy                           0.80       192
   macro avg       0.78      0.77      0.77       192
weighted avg       0.79      0.80      0.79       192

Xtreame Gradient Boosting   :--

[21:16:17] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'er
ror' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

[[106  19]
 [ 22  45]]

              precision    recall  f1-score   support

           0       0.83      0.85      0.84       125
           1       0.70      0.67      0.69        67

    accuracy                           0.79       192
   macro avg       0.77      0.76      0.76       192
weighted avg       0.78      0.79      0.79       192
```

In [18]:
```python
print('the mean accuracy of the model  :--  ',np.array(accuracy).mean()*100)
```

```
the mean accuracy of the model  :--   77.21354166666667
```

In [147…
```python
xgb=GradientBoostingClassifier(ccp_alpha=0.008)
xgb.fit(xtrain,ytrain)
ypred=xgb.predict(xtest)
print('trainig score  :-  ',xgb.score(xtrain,ytrain)*100)
print('testing score   :-  ',xgb.score(xtest,ytest)*100)
```

```
trainig score  :-   74.63414634146342
testing score   :-   73.8562091503268
```

In [148…
```python
skf=StratifiedKFold(n_splits=5)
skf.get_n_splits(x,y)
```

Out[148… 5

In [149…
```python
accuracy=[]

for train,test in skf.split(x,y):
    xtrain,xtest=x[train],x[test]
    ytrain,ytest=y[train],y[test]


    xgb=GradientBoostingClassifier(ccp_alpha=0.008)
    xgb.fit(xtrain,ytrain)
    ypred=xgb.predict(xtest)
    accuracy.append(accuracy_score(ytest,ypred))
```

In [150…
```python
print('this is the  minimum accuracy we can get from xtream boosting classifier  :--',np.array(accuracy).mean()*100)
```

```
this is the  minimum accuracy we can get from xtream boosting classifier  :-- 74.349376114082
```

Because the dataset is imbalance we are getting less accuracy

In [151…
```python
df.target.value_counts()
```

Out[151…
```
0    500
1    268
Name: target, dtype: int64
```

In [152…
```python
df.corr()
```

Out[152…

	pregnant	glocose	bp	skin	insulin	bmi	predigree	age	target
pregnant	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
glocose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
bp	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
skin	-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
predigree	-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
target	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [ ]: