DIABETES PREDICTION

It will predict which people are likely to develop diabetes.The objective of this dataset is to diagnostically predict whether or not the patient has diabetes based on a certain diagnostic measurement that is included in the data set.

ATTRIBUTE INFORMATION:

Preg - Number of time pregnant
Plas - Plasma glucose concentration
Pres - Blood pressure
Skin - Skin fold thickness (triceps skin fold thickness)
test - Two hour srum insuline test
mass - Body mass index
pedi - Diabetes pedegree function
age - Age of patient
class- Class variable (1 is tested positive and 0 is tested negative in diabetes)

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
data=pd.read_csv("pima-indians-diabetes.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Preg	Plas	Pres	skin	test	mass	pedi	age	class
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]:
```
data.shape
```

Out[4]:
```
(768, 9)
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Preg    768 non-null    int64  
 1   Plas    768 non-null    int64  
 2   Pres    768 non-null    int64  
 3   skin    768 non-null    int64  
 4   test    768 non-null    int64  
 5   mass    768 non-null    float64
 6   pedi    768 non-null    float64
 7   age     768 non-null    int64  
 8   class   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

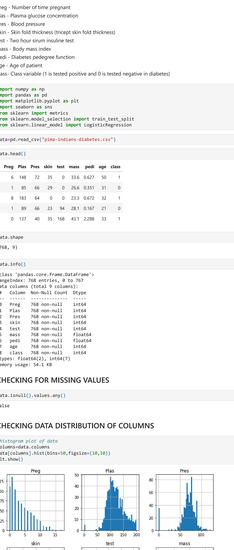
CHECKING FOR MISSING VALUES

In [6]:
```
data.isnull().values.any()
```

Out[6]:
```
False
```

CHECKING DATA DISTRIBUTION OF COLUMNS

In [7]:
```
#histogram plot of data
columns=data.columns
data[columns].hist(figsize=(10,10))
plt.show()
```



CHECKING MEASURE OF ASSOCIATION

In [8]:
```
# correlation
data.corr()
```

Out[8]:

	Preg	Plas	Pres	skin	test	mass	pedi	age	class
Preg	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Plas	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
Pres	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
test	0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	0.042163	0.130548
mass	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
pedi	-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
class	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [9]:
```
sns.pairplot(data)
plt.show()
```

CHECKING TRUE - FALSE DIABETES CASE

In [10]:
```
n_true=len(data.loc[data['class']==True])
n_false=len(data.loc[data['class']==False])
print("Number of true cases: {0} ({1}%)".format(n_true,(n_true/n_false)*100))
print("The number of false cases:{0} ({1}%".format(n_false,(n_false/n_false)*100))
```

```
The number of true cases:268 34.8958333333333%
The number of false cases:500 65.10416666666665%
```

In [11]:
```
x=data.drop('class',axis=1)
y=data['class']
```

In [12]:
```
x.head()
```

Out[12]:

	Preg	Plas	Pres	skin	test	mass	pedi	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [13]:
```
y.head()
```

Out[13]:
```
0    1
1    0
2    1
3    0
4    1
Name: class, dtype: int64
```

Train Test Split

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
```

In [15]:
```
x_train.head()
```

Out[15]:

	Preg	Plas	Pres	skin	test	mass	pedi	age
88	15	136	70	32	110	37.1	0.153	43
467	0	97	64	36	100	36.8	0.600	25
550	1	116	70	28	0	27.4	0.204	21
147	2	106	64	35	119	30.5	1.400	34
481	0	123	88	37	0	35.2	0.197	29

In [16]:
```
print("original diabetes true cases:{0} ({1}%".format(len(data.loc[data['class']==1]),len(data.loc[data['class']==1])/len(data.index)*100))
print("original diabetes false cases:{0} ({1}%".format(len(data.loc[data['class']==0]),len(data.loc[data['class']==0])/len(data.index)*100))
print("Training diabetes true cases:{0} ({1}%".format(len(y_train[y_train[:]==1]),(len(y_train[y_train[:]==1])/(len(y_train))*100)))
print("Training diabetes false cases:{0} ({1}%".format(len(y_train[y_train[:]==0]),(len(y_train[y_train[:]==0])/(len(y_train))*100)))
print("Test diabetes true cases:{0} ({1}%".format(len(y_test[y_test[:]==1]),(len(y_test[y_test[:]==1])/(len(y_test))*100)))
print("Test diabetes false cases:{0} ({1}%".format(len(y_test[y_test[:]==0]),(len(y_test[y_test[:]==0])/(len(y_test))*100)))
```

```
original diabetes true cases:268 34.8958333333333%
original diabetes false cases:500 65.10416666666665%
Training diabetes true cases:183 34.07821229050279%
Training diabetes false cases:354 65.92178770949721%
Test diabetes true cases:85 36.79653679653679%
Test diabetes false cases:146 63.203463203463195%
```

In [17]:
```
x_train.head(10)
```

Out[17]:

	Preg	Plas	Pres	skin	test	mass	pedi	age
88	15	136	70	32	110	37.1	0.153	43
467	0	97	64	36	100	36.8	0.600	25
550	1	116	70	28	0	27.4	0.204	21
147	2	106	64	35	119	30.5	1.400	34
481	0	123	88	37	0	35.2	0.197	29
412	1	143	84	23	310	42.4	1.076	22
248	9	124	70	33	402	35.4	0.282	34
642	6	147	80	0	0	29.5	0.178	50
519	6	129	90	7	326	19.6	0.582	60
730	3	130	78	23	79	28.4	0.323	34

FILLING ZERO VALUES WITH THE MEAN OF THE COLUMN

In [18]:
```
from sklearn.impute import SimpleImputer
rep_0=SimpleImputer(missing_values=0,strategy="mean")
cols=x_train.columns
x_train=pd.DataFrame(rep_0.fit_transform(x_train))
```

In [19]:
```
x_test=pd.DataFrame(rep_0.fit_transform(x_test))
```

In [20]:
```
x_train.columns=cols
x_test.columns=cols
```

In [21]:
```
x_train.head()
```

Out[21]:

	Preg	Plas	Pres	skin	test	mass	pedi	age
0	15.000000	136.0	70.0	32.0	110.000000	37.1	0.153	43.0
1	4.396514	97.0	64.0	36.0	100.000000	36.8	0.600	25.0
2	1.000000	116.0	70.0	28.0	158.243346	27.4	0.204	21.0
3	2.000000	106.0	64.0	35.0	119.000000	30.5	1.400	34.0
4	4.396514	123.0	88.0	37.0	158.243346	35.2	0.197	29.0

APPLYING LOGISTIC REGRESSION

In [22]:
```
model=LogisticRegression(solver='liblinear')
model.fit(x_train,y_train)
```

Out[22]:
```
LogisticRegression(solver='liblinear')
```

In [23]:
```
y_predict=model.predict(x_test)
```

In [24]:
```
model_score=model.score(x_test,y_test)
print(model_score)
```

```
0.7792207792207793
```

Confusion matrix

In [25]:
```
cm=metrics.confusion_matrix(y_test,y_predict)
```

In [26]:
```
print(cm)
```

```
[[132  14]
 [ 37  48]]
```

In [ ]:
```
True Positive (TP) - The probability of them having diabetes was predicted correctly,i.e 132. <br>
True Negative (TN) - The probability that they don't have diabetes was predicted correctly, i.e 48. <br>
False Positive (FP) - The probability of them having diabetes was predicted incorrectly, i.e 14. <br>
False Negative (FN) - The probability that don't have diabetes was predicted incorrectly, i.e 37. <br>
```

In [27]:
```
from sklearn.naive_bayes import GaussianNB
```

In [28]:
```
nb_model=GaussianNB()
```

In [29]:
```
nb_model.fit(x_train,y_train)
```

Out[29]:
```
GaussianNB()
```

In [30]:
```
nb_y_predict=nb_model.predict(x_test)
```

In [31]:
```
nb_model_accuracy=nb_model.score(x_test,y_test)
```

In [32]:
```
print(nb_model_accuracy)
```

```
0.7705627705627706
```

In [33]:
```
cm=metrics.confusion_matrix(y_test,nb_y_predict)
```

In [34]:
```
print(cm)
```

```
[[123  23]
 [ 30  55]]
```

CONCLUSION:

From above metrics we can say that with the help logistic classifier we are predicted higher correctly values and the score of our model is also good calculated by logistic classifier,that means the performance of the model calculated by logistic algorithms is more good than that of naive bayes.

In [ ]:
```

```