In [1]:
```
# Libs Imports
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
import sys
import warnings
from pyod.models.knn import KNN
from sklearn.preprocessing import Normalizer
from imblearn import under_sampling, over_sampling
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier

import tensorflow
from tensorflow import keras
import keras
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.models import load_model
from tensorflow.keras.layers import Dense
```

In [2]:
```
# Command to suppress warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

Exploratory analysis

In [3]:
```
# Reading the file
df1 = pd.read_csv('diabetes.csv')
```

In [4]:
```
df1.head()
```

Out[4]:

	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

In [5]:
```
# Null Values Detection
df1.isnull().sum()
```

Out[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [6]:
```
# Missing Values Detection
df1.isna().sum()
```

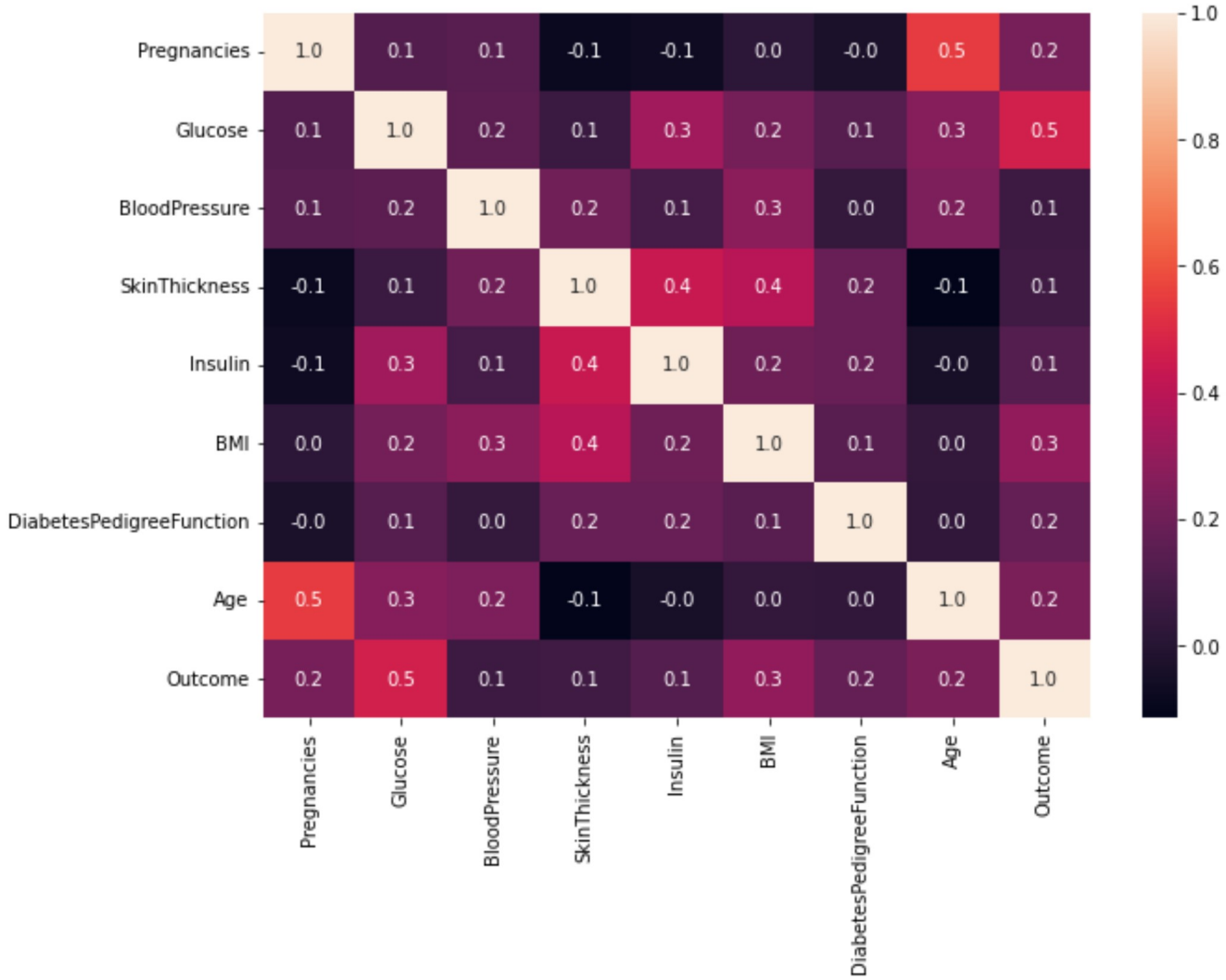
Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```
# Duplicate Values Detection
df1.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
#correlation - Heat map

matriz_correlacao = df1.corr( )
plt.figure(figsize = (10, 7))
sns.heatmap(matriz_correlacao, annot = True, fmt='.1f')
```

Out[8]:
```
<Axes: >
```



In [9]:
```
# Separating the variable
df2 = df1.iloc[:, 0:8]
```

In [10]:
```
df2.head(2)
```

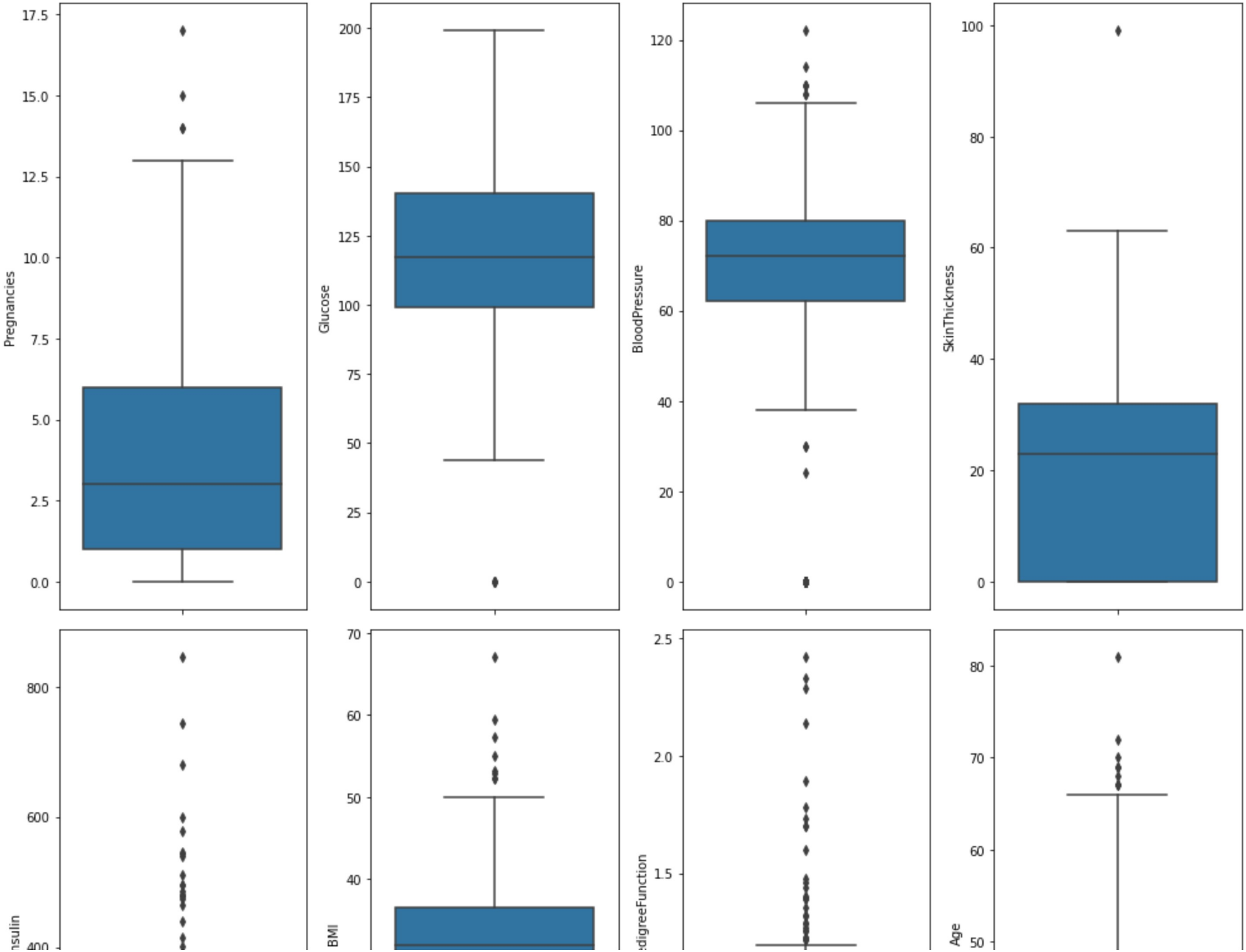
Out[10]:

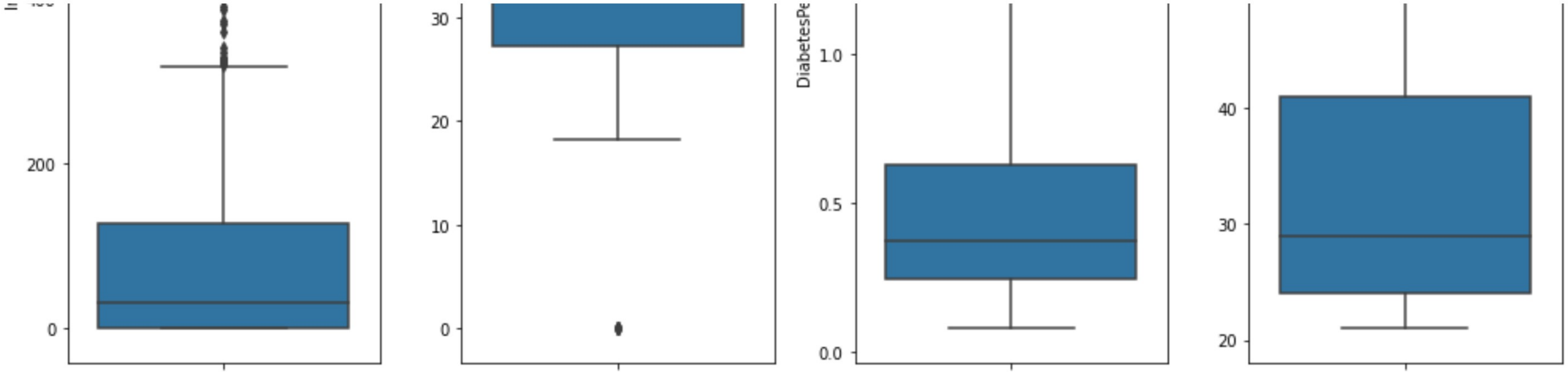
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31

In [11]:

```
#Loading variables to plot

Variables = []
for i in df2.columns[0:8].tolist():
  #      if df2.dtypes[i] == 'int64' or df2.dtypes[i] == 'float64':
            print(i, ':' , df2.dtypes[i])
            Variables.append(i)



plt.rcParams["figure.figsize"] = [14.00, 14.00]
plt.rcParams["figure.autolayout"] = True

plt.rcParams['font.size'] = 10
plt.rcParams['axes.titlesize'] = 20

f, axes = plt.subplots(2, 4) #4 linhas e 3 colunas

linha = 0
coluna = 0
for i in Variables:
    sns.boxplot(data = df2, y=i, ax=axes[linha][coluna])
    coluna += 1
    if coluna == 4:
        linha += 1
        coluna = 0

plt.show()
```

```
Pregnancies : int64
Glucose : int64
BloodPressure : int64
SkinThickness : int64
Insulin : int64
BMI : float64
DiabetesPedigreeFunction : float64
Age : int64
```


Pregnancies
Glucose
BloodPressure
SkinThickness
BMI
Age



In [12]:
```
# building filter - I used df1 because we need to exclude the lines including the target variable.

filter_Pregnacies = df1['Pregnancies'] < 13
filter_glucose = df1['Glucose'] > 0
filter_BP = df1['BloodPressure'] > 0
filter_BMI = df1['BMI'] > 0
filter_SkinThickness = df1['SkinThickness'] < 99
```

In [13]:
```
# Aply filter
df3 = df1[filter_Pregnacies]
df3 = df3[filter_glucose]
df3 = df3[filter_BP]
df3 = df3[filter_BMI]
df3 = df3[filter_SkinThickness]
```

In [14]:
```
df3.shape
```

Out[14]: (710, 9)

In [15]:
```
df3['SkinThickness'].sort_values(ascending=False)
```

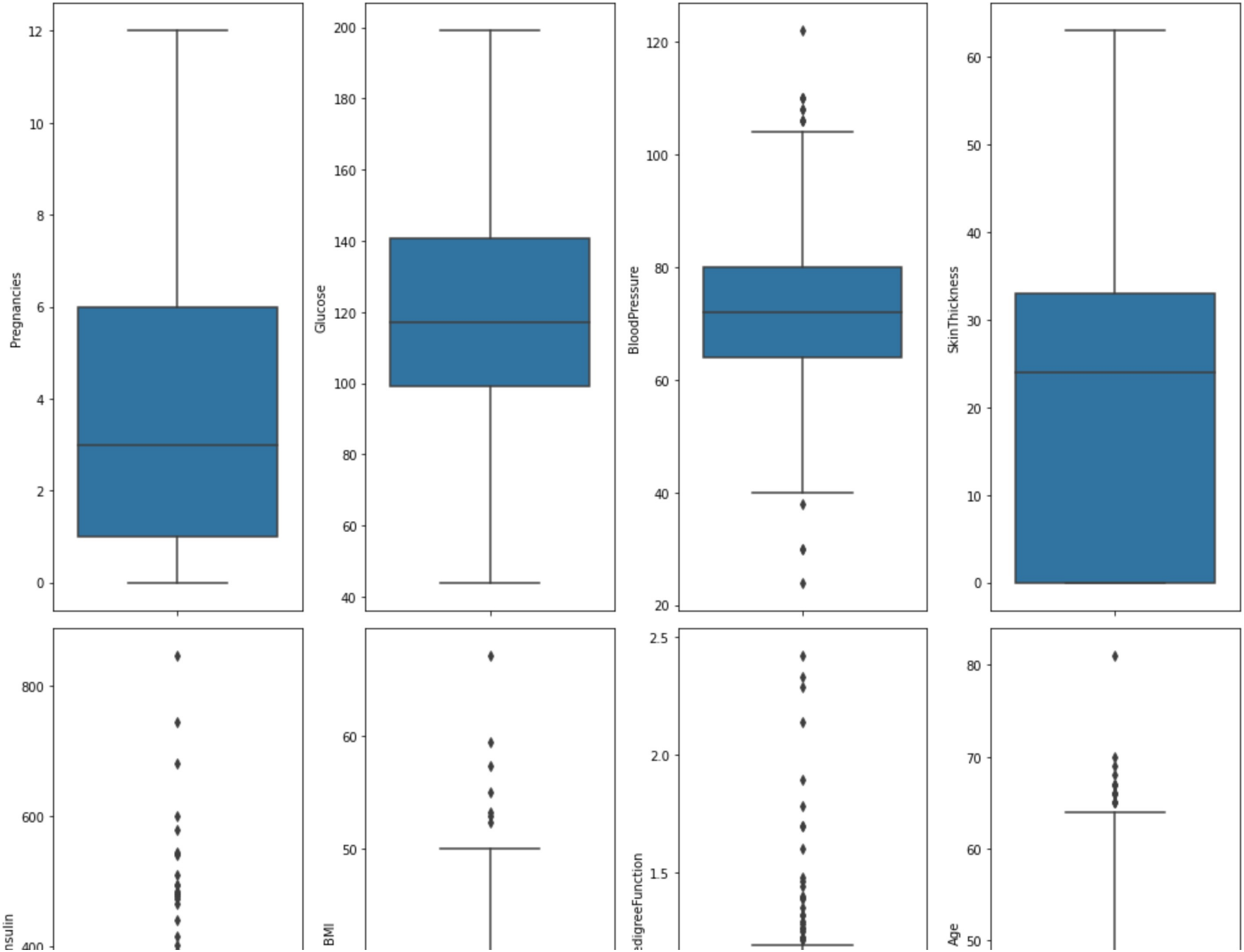
Out[15]:
```
445    63
57     60
120    56
211    54
275    52
       ..
559     0
560     0
264     0
564     0
164     0
Name: SkinThickness, Length: 710, dtype: int64
```

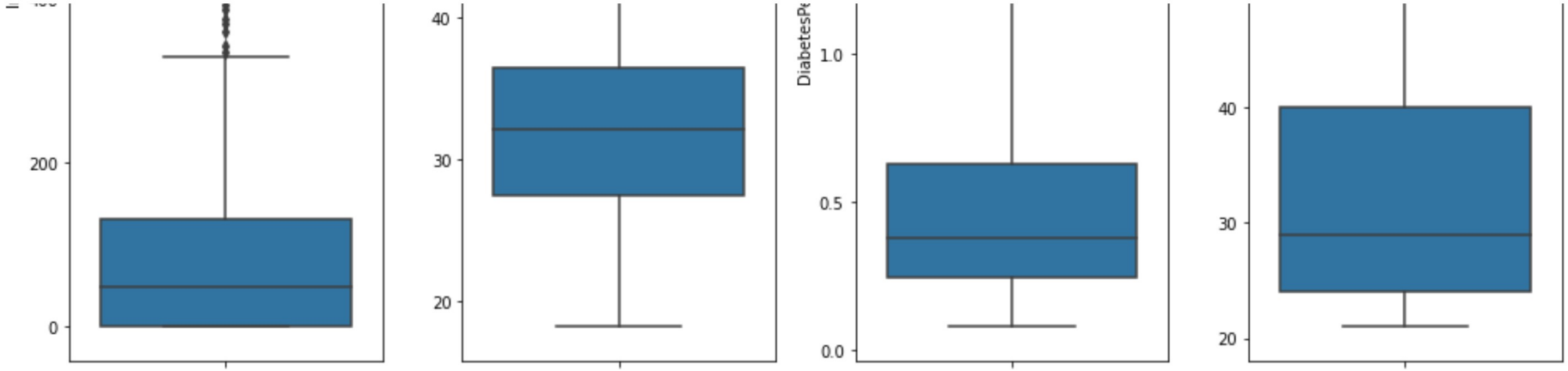
Next Steps:

let's put the predictor variables on the same scale. To do this, we will separate it again into predictor variables and target variables. After placing the predictor variables on the same scale, we will analyze the need to balance the target variable.

In [16]:
```
#Loading variables to plot

Variables = []
for i in df3.columns[0:8].tolist():
  #      if df2.dtypes[i] == 'int64' or df2.dtypes[i] == 'float64':
            print(i, ':' , df3.dtypes[i])
            Variables.append(i)



plt.rcParams["figure.figsize"] = [14.00, 14.00]
plt.rcParams["figure.autolayout"] = True

plt.rcParams['font.size'] = 10
plt.rcParams['axes.titlesize'] = 20

f, axes = plt.subplots(2, 4) #4 linhas e 3 colunas

linha = 0
coluna = 0
for i in Variables:
    sns.boxplot(data = df3, y=i, ax=axes[linha][coluna])
    coluna += 1
    if coluna == 4:
        linha += 1
        coluna = 0

plt.show()
```

```
Pregnancies : int64
Glucose : int64
BloodPressure : int64
SkinThickness : int64
Insulin : int64
BMI : float64
DiabetesPedigreeFunction : float64
Age : int64
```


Pregnancies
Glucose
BloodPressure
SkinThickness
nsulin
BMI
digreeFunction
Age



In [17]:
```
# Separating the variable
df3_predictor = df3.iloc[:, 0:8]
```

In [18]:
```
df3_predictor.shape
```

Out[18]: (710, 8)

In [19]:
```
df3_predictor.head(3)
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

Normalized (put in the same scale)

In [20]:
```
df3_normalized = df3_predictor
```

In [21]:
```
df3_normalized = Normalizer().fit_transform(df3_normalized.values)
```

In [22]:
```
df3_normalized = pd.DataFrame(df3_normalized)
```

In [23]:
```
df3_normalized.head()
```

Out[23]:

	0	1	2	3	4	5	6	7
0	0.033552	0.827625	0.402628	0.195722	0.000000	0.187893	0.003506	0.279603
1	0.008424	0.716040	0.555984	0.244296	0.000000	0.224079	0.002957	0.261144
2	0.040398	0.924097	0.323181	0.000000	0.000000	0.117658	0.003393	0.161591
3	0.006612	0.588467	0.436392	0.152076	0.621527	0.185797	0.001104	0.138852
4	0.000000	0.596386	0.174127	0.152361	0.731335	0.187622	0.009960	0.143655

In [ ]:
```
```

In [24]:
```
df3_normalized.columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunct
```

In [25]:
```
df3_normalized.head()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	0.033552	0.827625	0.402628	0.195722	0.000000	0.187893	0.003506	0.279603
1	0.008424	0.716040	0.555984	0.244296	0.000000	0.224079	0.002957	0.261144
2	0.040398	0.924097	0.323181	0.000000	0.000000	0.117658	0.003393	0.161591
3	0.006612	0.588467	0.436392	0.152076	0.621527	0.185797	0.001104	0.138852
4	0.000000	0.596386	0.174127	0.152361	0.731335	0.187622	0.009960	0.143655

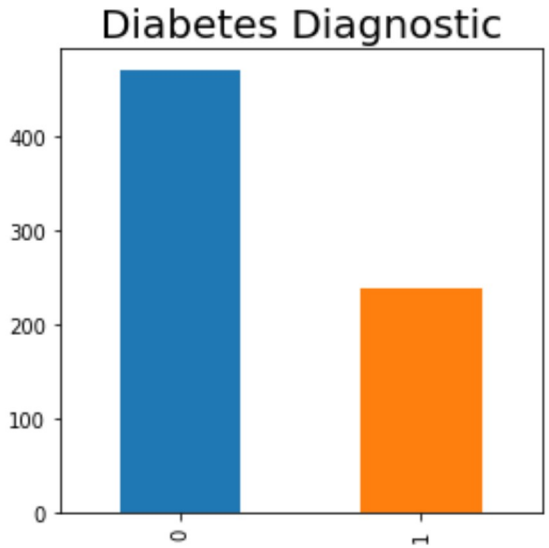
Balanced target variable

In [26]:
```
#Now We goin to verify if the target variable is balanced
df3.groupby(['Outcome']).size()
```

Out[26]:
```
Outcome
0    470
1    240
dtype: int64
```

In [27]:
```
# Building a graphic to show the diference between diagnostics
plt.rcParams['figure.figsize'] = [4, 4]
plt.rcParams['figure.autolayout'] = True
df3.Outcome.value_counts().plot(kind='bar', title="Diabetes Diagnostic", color = ['#1F77B4','#FF7F0E'])
```

Out[27]:
```
<Axes: title={'center': 'Diabetes Diagnostic'}>
```

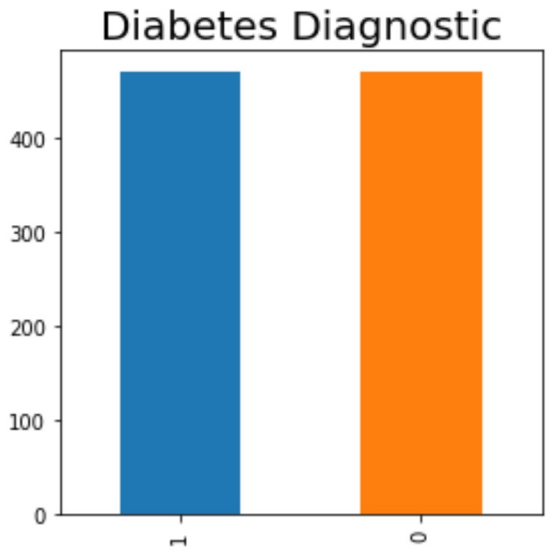


In [28]:
```
target = df3['Outcome']
```

In [29]:
```
# It's necssary Smote the target variable

# creates the smote balancer
balanceador = SMOTE(random_state=100)

# Apply balancer
predictor_res, target_res = balanceador.fit_resample(df3_normalized, target)
```

In [30]:
```
# Building a graphic to show the diference between diagnostics
plt.rcParams['figure.figsize'] = [4, 4]
plt.rcParams['figure.autolayout'] = True
target_res.value_counts().plot(kind='bar', title="Diabetes Diagnostic", color = ['#1F77B4','#FF7F0E'])
```

Out[30]: <Axes: title={'center': 'Diabetes Diagnostic'}>



In [31]:
```
df3_normalized.shape
```

Out[31]: (710, 8)

In [32]:
```
predictor_res.shape
```

Out[32]: (940, 8)

In [33]:
```
target.shape
```

Out[33]: (710,)

In [34]:
```
target_res.shape
```

Out[34]: (940,)

Separate datas in train and test

In [35]:
```
X_train, X_test, Y_train, Y_test = train_test_split(predictor_res, target_res, test_size = 0.3, random_state = 42)
```

Creating the classifier with Random Forest and testing

In [36]:
```
# Creating the classifier with Random Forest
clf = RandomForestClassifier(n_estimators  = 300,
                             max_features='log2',
                             )

# Model construction
clf = clf.fit(X_train, Y_train)
```

In [37]:
```
# Checking model accuracy with test data
scores = clf.score(X_test,Y_test)
```

In [47]:
```
print('Accuracy score:', (scores).round(4)*100)
```

```
Accuracy score: 75.18
```

In [ ]:
```
# ------------------------------------------------------------------------------------------#
```

Comparing Machine Learning and Deep Learning - Neural Networks

In [48]:
```
X_train.shape[1]
```

Out[48]:
```
8
```

In [49]:
```
X_train.head(2)
```

Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
296	0.014559	0.469543	0.218392	0.043678	0.840809	0.100096	0.001918	0.112836
417	0.000000	0.370746	0.277041	0.130372	0.855568	0.162558	0.001552	0.101853

Creating the neural network architecture

In [50]:
```
model_nn = Sequential()
model_nn.add(Dense(units=128, activation='relu', input_dim = X_train.shape[1]))
model_nn.add(Dense(units=64, activation='relu')),
model_nn.add(Dense(units=1, activation='linear'))

# training the neural network:
model_nn.compile(loss = 'mse', optimizer = 'adam' , metrics=['mae', 'accuracy'])

results_nn = model_nn.fit(X_train, Y_train,
                          epochs= 110,
                          batch_size= 32,

                            validation_data= (X_test, Y_test)
                         )
```

```
Epoch 1/110
21/21 [==============================] - 1s 12ms/step - loss: 0.3128 - mae: 0.5013 - accuracy: 0.4985 - val_loss: 0.2624 - val_
mae: 0.4977 - val_accuracy: 0.4858
Epoch 2/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2422 - mae: 0.4863 - accuracy: 0.5745 - val_loss: 0.2393 - val_m
ae: 0.4832 - val_accuracy: 0.6028
Epoch 3/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2375 - mae: 0.4811 - accuracy: 0.6049 - val_loss: 0.2327 - val_m
ae: 0.4766 - val_accuracy: 0.6631
Epoch 4/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2317 - mae: 0.4726 - accuracy: 0.6368 - val_loss: 0.2297 - val_m
ae: 0.4703 - val_accuracy: 0.6348
Epoch 5/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2256 - mae: 0.4650 - accuracy: 0.6489 - val_loss: 0.2223 - val_m
ae: 0.4620 - val_accuracy: 0.6667
Epoch 6/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2235 - mae: 0.4586 - accuracy: 0.6398 - val_loss: 0.2178 - val_m
ae: 0.4531 - val_accuracy: 0.6809
Epoch 7/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2190 - mae: 0.4510 - accuracy: 0.6581 - val_loss: 0.2147 - val_m
ae: 0.4480 - val_accuracy: 0.6773
Epoch 8/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2180 - mae: 0.4476 - accuracy: 0.6535 - val_loss: 0.2127 - val_m
ae: 0.4452 - val_accuracy: 0.6702
Epoch 9/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2150 - mae: 0.4418 - accuracy: 0.6611 - val_loss: 0.2139 - val_m
ae: 0.4426 - val_accuracy: 0.6489
Epoch 10/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2137 - mae: 0.4364 - accuracy: 0.6748 - val_loss: 0.2079 - val_m
ae: 0.4362 - val_accuracy: 0.7057
Epoch 11/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2105 - mae: 0.4327 - accuracy: 0.6793 - val_loss: 0.2209 - val_m
ae: 0.4376 - val_accuracy: 0.6241
Epoch 12/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2131 - mae: 0.4291 - accuracy: 0.6641 - val_loss: 0.2105 - val_m
ae: 0.4331 - val_accuracy: 0.6312
Epoch 13/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2123 - mae: 0.4295 - accuracy: 0.6687 - val_loss: 0.2049 - val_m
ae: 0.4292 - val_accuracy: 0.6879
Epoch 14/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2087 - mae: 0.4290 - accuracy: 0.6793 - val_loss: 0.2029 - val_m
```

```
ae: 0.4264 - val_accuracy: 0.6809
Epoch 15/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2086 - mae: 0.4247 - accuracy: 0.6763 - val_loss: 0.2047 - val_m
ae: 0.4258 - val_accuracy: 0.6596
Epoch 16/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2115 - mae: 0.4240 - accuracy: 0.6611 - val_loss: 0.2176 - val_m
ae: 0.4206 - val_accuracy: 0.6454
Epoch 17/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2153 - mae: 0.4227 - accuracy: 0.6809 - val_loss: 0.2127 - val_m
ae: 0.4224 - val_accuracy: 0.6702
Epoch 18/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2183 - mae: 0.4266 - accuracy: 0.6581 - val_loss: 0.2015 - val_m
ae: 0.4262 - val_accuracy: 0.6950
Epoch 19/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2092 - mae: 0.4241 - accuracy: 0.6687 - val_loss: 0.2109 - val_m
ae: 0.4262 - val_accuracy: 0.6631
Epoch 20/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2059 - mae: 0.4140 - accuracy: 0.6839 - val_loss: 0.2028 - val_m
ae: 0.4201 - val_accuracy: 0.7128
Epoch 21/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2062 - mae: 0.4171 - accuracy: 0.6809 - val_loss: 0.2013 - val_m
ae: 0.4180 - val_accuracy: 0.7057
Epoch 22/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2042 - mae: 0.4166 - accuracy: 0.6854 - val_loss: 0.2014 - val_m
ae: 0.4183 - val_accuracy: 0.6738
Epoch 23/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2040 - mae: 0.4133 - accuracy: 0.6854 - val_loss: 0.1998 - val_m
ae: 0.4161 - val_accuracy: 0.6915
Epoch 24/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2051 - mae: 0.4157 - accuracy: 0.7006 - val_loss: 0.2129 - val_m
ae: 0.4184 - val_accuracy: 0.6489
Epoch 25/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2051 - mae: 0.4089 - accuracy: 0.6809 - val_loss: 0.2035 - val_m
ae: 0.4161 - val_accuracy: 0.6738
Epoch 26/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2018 - mae: 0.4111 - accuracy: 0.7006 - val_loss: 0.1983 - val_m
ae: 0.4149 - val_accuracy: 0.7411
Epoch 27/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2018 - mae: 0.4097 - accuracy: 0.7036 - val_loss: 0.1994 - val_m
ae: 0.4129 - val_accuracy: 0.6844
Epoch 28/110
```

```
21/21 [==============================] - 0s 3ms/step - loss: 0.2018 - mae: 0.4085 - accuracy: 0.6976 - val_loss: 0.1977 - val_m
ae: 0.4132 - val_accuracy: 0.7021
Epoch 29/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2048 - mae: 0.4128 - accuracy: 0.6991 - val_loss: 0.2003 - val_m
ae: 0.4134 - val_accuracy: 0.6915
Epoch 30/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2008 - mae: 0.4077 - accuracy: 0.7052 - val_loss: 0.1985 - val_m
ae: 0.4118 - val_accuracy: 0.6915
Epoch 31/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2010 - mae: 0.4069 - accuracy: 0.6976 - val_loss: 0.1990 - val_m
ae: 0.4105 - val_accuracy: 0.7092
Epoch 32/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2023 - mae: 0.4064 - accuracy: 0.6824 - val_loss: 0.1972 - val_m
ae: 0.4106 - val_accuracy: 0.7163
Epoch 33/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2032 - mae: 0.4082 - accuracy: 0.6915 - val_loss: 0.1971 - val_m
ae: 0.4096 - val_accuracy: 0.7270
Epoch 34/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2017 - mae: 0.4056 - accuracy: 0.7082 - val_loss: 0.2092 - val_m
ae: 0.4125 - val_accuracy: 0.6667
Epoch 35/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2006 - mae: 0.4015 - accuracy: 0.6854 - val_loss: 0.2098 - val_m
ae: 0.4126 - val_accuracy: 0.6596
Epoch 36/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2008 - mae: 0.4057 - accuracy: 0.6991 - val_loss: 0.1958 - val_m
ae: 0.4078 - val_accuracy: 0.7092
Epoch 37/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1986 - mae: 0.4018 - accuracy: 0.7052 - val_loss: 0.1985 - val_m
ae: 0.4081 - val_accuracy: 0.7199
Epoch 38/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2009 - mae: 0.4042 - accuracy: 0.6991 - val_loss: 0.1956 - val_m
ae: 0.4082 - val_accuracy: 0.7234
Epoch 39/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1979 - mae: 0.4001 - accuracy: 0.7036 - val_loss: 0.1961 - val_m
ae: 0.4039 - val_accuracy: 0.7234
Epoch 40/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2097 - mae: 0.4106 - accuracy: 0.6854 - val_loss: 0.2015 - val_m
ae: 0.4100 - val_accuracy: 0.7128
Epoch 41/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2013 - mae: 0.4042 - accuracy: 0.6960 - val_loss: 0.1964 - val_m
ae: 0.4097 - val_accuracy: 0.7128
```

```
Epoch 42/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1995 - mae: 0.4046 - accuracy: 0.6991 - val_loss: 0.1974 - val_m
ae: 0.4102 - val_accuracy: 0.6986
Epoch 43/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1985 - mae: 0.4023 - accuracy: 0.7097 - val_loss: 0.1965 - val_m
ae: 0.4058 - val_accuracy: 0.6950
Epoch 44/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1964 - mae: 0.3986 - accuracy: 0.7021 - val_loss: 0.1976 - val_m
ae: 0.4047 - val_accuracy: 0.6915
Epoch 45/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1983 - mae: 0.3995 - accuracy: 0.7036 - val_loss: 0.1953 - val_m
ae: 0.4039 - val_accuracy: 0.7199
Epoch 46/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1981 - mae: 0.3981 - accuracy: 0.6930 - val_loss: 0.2005 - val_m
ae: 0.4048 - val_accuracy: 0.6809
Epoch 47/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1971 - mae: 0.3967 - accuracy: 0.7021 - val_loss: 0.2029 - val_m
ae: 0.4073 - val_accuracy: 0.7128
Epoch 48/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2030 - mae: 0.4042 - accuracy: 0.6976 - val_loss: 0.1966 - val_m
ae: 0.4054 - val_accuracy: 0.7305
Epoch 49/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2006 - mae: 0.4025 - accuracy: 0.7021 - val_loss: 0.1955 - val_m
ae: 0.4057 - val_accuracy: 0.7234
Epoch 50/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1979 - mae: 0.4020 - accuracy: 0.6960 - val_loss: 0.2019 - val_m
ae: 0.4070 - val_accuracy: 0.6773
Epoch 51/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1977 - mae: 0.3963 - accuracy: 0.6960 - val_loss: 0.1936 - val_m
ae: 0.4019 - val_accuracy: 0.7305
Epoch 52/110
21/21 [==============================] - 0s 4ms/step - loss: 0.2013 - mae: 0.3976 - accuracy: 0.6991 - val_loss: 0.1942 - val_m
ae: 0.4045 - val_accuracy: 0.7163
Epoch 53/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1984 - mae: 0.4020 - accuracy: 0.7036 - val_loss: 0.1996 - val_m
ae: 0.4051 - val_accuracy: 0.7057
Epoch 54/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1963 - mae: 0.3961 - accuracy: 0.6991 - val_loss: 0.2025 - val_m
ae: 0.4049 - val_accuracy: 0.6879
Epoch 55/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1959 - mae: 0.3987 - accuracy: 0.7036 - val_loss: 0.2072 - val_m
```

```
ae: 0.4058 - val_accuracy: 0.6809
Epoch 56/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1958 - mae: 0.3955 - accuracy: 0.7052 - val_loss: 0.1953 - val_m
ae: 0.4014 - val_accuracy: 0.7021
Epoch 57/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1932 - mae: 0.3946 - accuracy: 0.7173 - val_loss: 0.1976 - val_m
ae: 0.3991 - val_accuracy: 0.6915
Epoch 58/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1941 - mae: 0.3886 - accuracy: 0.7173 - val_loss: 0.1930 - val_m
ae: 0.3998 - val_accuracy: 0.7199
Epoch 59/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1943 - mae: 0.3948 - accuracy: 0.7128 - val_loss: 0.1955 - val_m
ae: 0.3992 - val_accuracy: 0.7092
Epoch 60/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1934 - mae: 0.3901 - accuracy: 0.7082 - val_loss: 0.1970 - val_m
ae: 0.3975 - val_accuracy: 0.6986
Epoch 61/110
21/21 [==============================] - 0s 3ms/step - loss: 0.2020 - mae: 0.3948 - accuracy: 0.6884 - val_loss: 0.1985 - val_m
ae: 0.4008 - val_accuracy: 0.6950
Epoch 62/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1971 - mae: 0.3938 - accuracy: 0.6991 - val_loss: 0.1929 - val_m
ae: 0.4013 - val_accuracy: 0.7128
Epoch 63/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1928 - mae: 0.3927 - accuracy: 0.7112 - val_loss: 0.1932 - val_m
ae: 0.3969 - val_accuracy: 0.7199
Epoch 64/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1919 - mae: 0.3901 - accuracy: 0.7158 - val_loss: 0.1921 - val_m
ae: 0.3964 - val_accuracy: 0.7270
Epoch 65/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1921 - mae: 0.3900 - accuracy: 0.7188 - val_loss: 0.1953 - val_m
ae: 0.3971 - val_accuracy: 0.7057
Epoch 66/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1905 - mae: 0.3879 - accuracy: 0.7082 - val_loss: 0.1936 - val_m
ae: 0.3962 - val_accuracy: 0.7376
Epoch 67/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1916 - mae: 0.3844 - accuracy: 0.7143 - val_loss: 0.1898 - val_m
ae: 0.3923 - val_accuracy: 0.7305
Epoch 68/110
21/21 [==============================] - 0s 5ms/step - loss: 0.1904 - mae: 0.3873 - accuracy: 0.7158 - val_loss: 0.1952 - val_m
ae: 0.3944 - val_accuracy: 0.7057
Epoch 69/110
```

```
21/21 [==============================] - 0s 3ms/step - loss: 0.1909 - mae: 0.3874 - accuracy: 0.7143 - val_loss: 0.1980 - val_m
ae: 0.3941 - val_accuracy: 0.6915
Epoch 70/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1902 - mae: 0.3857 - accuracy: 0.7082 - val_loss: 0.1895 - val_m
ae: 0.3915 - val_accuracy: 0.7340
Epoch 71/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1913 - mae: 0.3866 - accuracy: 0.7067 - val_loss: 0.1986 - val_m
ae: 0.3945 - val_accuracy: 0.6950
Epoch 72/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1943 - mae: 0.3880 - accuracy: 0.7082 - val_loss: 0.2093 - val_m
ae: 0.3958 - val_accuracy: 0.6631
Epoch 73/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1909 - mae: 0.3858 - accuracy: 0.7143 - val_loss: 0.1909 - val_m
ae: 0.3935 - val_accuracy: 0.7270
Epoch 74/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1885 - mae: 0.3822 - accuracy: 0.7158 - val_loss: 0.1902 - val_m
ae: 0.3911 - val_accuracy: 0.7270
Epoch 75/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1920 - mae: 0.3899 - accuracy: 0.7158 - val_loss: 0.1920 - val_m
ae: 0.3907 - val_accuracy: 0.7199
Epoch 76/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1901 - mae: 0.3791 - accuracy: 0.7067 - val_loss: 0.1937 - val_m
ae: 0.3904 - val_accuracy: 0.6986
Epoch 77/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1911 - mae: 0.3864 - accuracy: 0.7143 - val_loss: 0.1962 - val_m
ae: 0.3900 - val_accuracy: 0.7021
Epoch 78/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1899 - mae: 0.3850 - accuracy: 0.7249 - val_loss: 0.1891 - val_m
ae: 0.3919 - val_accuracy: 0.7340
Epoch 79/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1897 - mae: 0.3860 - accuracy: 0.7097 - val_loss: 0.1908 - val_m
ae: 0.3916 - val_accuracy: 0.7340
Epoch 80/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1869 - mae: 0.3856 - accuracy: 0.7188 - val_loss: 0.1884 - val_m
ae: 0.3880 - val_accuracy: 0.7270
Epoch 81/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1906 - mae: 0.3809 - accuracy: 0.7264 - val_loss: 0.1905 - val_m
ae: 0.3911 - val_accuracy: 0.7340
Epoch 82/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1922 - mae: 0.3885 - accuracy: 0.7052 - val_loss: 0.1895 - val_m
ae: 0.3887 - val_accuracy: 0.7234
```

```
Epoch 83/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1870 - mae: 0.3803 - accuracy: 0.7173 - val_loss: 0.1974 - val_m
ae: 0.3882 - val_accuracy: 0.6986
Epoch 84/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1874 - mae: 0.3809 - accuracy: 0.7280 - val_loss: 0.1867 - val_m
ae: 0.3857 - val_accuracy: 0.7553
Epoch 85/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1846 - mae: 0.3776 - accuracy: 0.7249 - val_loss: 0.2043 - val_m
ae: 0.3875 - val_accuracy: 0.6702
Epoch 86/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1869 - mae: 0.3779 - accuracy: 0.7219 - val_loss: 0.1943 - val_m
ae: 0.3848 - val_accuracy: 0.7163
Epoch 87/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1878 - mae: 0.3826 - accuracy: 0.7249 - val_loss: 0.1909 - val_m
ae: 0.3850 - val_accuracy: 0.7092
Epoch 88/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1860 - mae: 0.3752 - accuracy: 0.7310 - val_loss: 0.1895 - val_m
ae: 0.3880 - val_accuracy: 0.7376
Epoch 89/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1874 - mae: 0.3768 - accuracy: 0.7280 - val_loss: 0.1939 - val_m
ae: 0.3855 - val_accuracy: 0.6950
Epoch 90/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1873 - mae: 0.3833 - accuracy: 0.7219 - val_loss: 0.1868 - val_m
ae: 0.3872 - val_accuracy: 0.7340
Epoch 91/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1854 - mae: 0.3777 - accuracy: 0.7280 - val_loss: 0.1853 - val_m
ae: 0.3813 - val_accuracy: 0.7482
Epoch 92/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1854 - mae: 0.3754 - accuracy: 0.7280 - val_loss: 0.1881 - val_m
ae: 0.3853 - val_accuracy: 0.7340
Epoch 93/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1876 - mae: 0.3780 - accuracy: 0.7052 - val_loss: 0.1850 - val_m
ae: 0.3822 - val_accuracy: 0.7376
Epoch 94/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1842 - mae: 0.3771 - accuracy: 0.7219 - val_loss: 0.1901 - val_m
ae: 0.3865 - val_accuracy: 0.7411
Epoch 95/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1817 - mae: 0.3695 - accuracy: 0.7416 - val_loss: 0.1877 - val_m
ae: 0.3786 - val_accuracy: 0.7128
Epoch 96/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1813 - mae: 0.3753 - accuracy: 0.7371 - val_loss: 0.1851 - val_m
```

```
ae: 0.3817 - val_accuracy: 0.7411
Epoch 97/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1861 - mae: 0.3756 - accuracy: 0.7264 - val_loss: 0.1851 - val_m
ae: 0.3787 - val_accuracy: 0.7376
Epoch 98/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1808 - mae: 0.3728 - accuracy: 0.7356 - val_loss: 0.1845 - val_m
ae: 0.3795 - val_accuracy: 0.7376
Epoch 99/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1828 - mae: 0.3772 - accuracy: 0.7280 - val_loss: 0.1823 - val_m
ae: 0.3740 - val_accuracy: 0.7482
Epoch 100/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1806 - mae: 0.3672 - accuracy: 0.7401 - val_loss: 0.1934 - val_m
ae: 0.3772 - val_accuracy: 0.7092
Epoch 101/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1804 - mae: 0.3723 - accuracy: 0.7219 - val_loss: 0.1884 - val_m
ae: 0.3787 - val_accuracy: 0.7340
Epoch 102/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1829 - mae: 0.3711 - accuracy: 0.7264 - val_loss: 0.1894 - val_m
ae: 0.3792 - val_accuracy: 0.7199
Epoch 103/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1800 - mae: 0.3701 - accuracy: 0.7371 - val_loss: 0.1857 - val_m
ae: 0.3778 - val_accuracy: 0.7376
Epoch 104/110
21/21 [==============================] - 0s 4ms/step - loss: 0.1793 - mae: 0.3686 - accuracy: 0.7356 - val_loss: 0.1815 - val_m
ae: 0.3721 - val_accuracy: 0.7447
Epoch 105/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1785 - mae: 0.3644 - accuracy: 0.7371 - val_loss: 0.1823 - val_m
ae: 0.3764 - val_accuracy: 0.7518
Epoch 106/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1790 - mae: 0.3691 - accuracy: 0.7477 - val_loss: 0.1829 - val_m
ae: 0.3774 - val_accuracy: 0.7411
Epoch 107/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1775 - mae: 0.3666 - accuracy: 0.7340 - val_loss: 0.1811 - val_m
ae: 0.3722 - val_accuracy: 0.7411
Epoch 108/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1784 - mae: 0.3626 - accuracy: 0.7295 - val_loss: 0.1848 - val_m
ae: 0.3772 - val_accuracy: 0.7447
Epoch 109/110
21/21 [==============================] - 0s 3ms/step - loss: 0.1782 - mae: 0.3690 - accuracy: 0.7553 - val_loss: 0.1923 - val_m
ae: 0.3784 - val_accuracy: 0.7199
Epoch 110/110
```

```
21/21 [==============================] - 0s 3ms/step - loss: 0.1846 - mae: 0.3717 - accuracy: 0.7310 - val_loss: 0.1831 - val_m
ae: 0.3744 - val_accuracy: 0.7411
```

In [72]:
```
loss_test,mae_test, accuracy_test = model_nn.evaluate(X_test, Y_test)

print('Loss test:', loss_test)
print('Accuracy score:', accuracy_test)
```

```
9/9 [==============================] - 0s 2ms/step - loss: 0.1831 - mae: 0.3744 - accuracy: 0.7411
Loss test: 0.18311181664466858
Accuracy score: 0.741134762763977
```

In [57]:
```
# plotar training history graph
plt.figure(figsize = (10, 7))
plt.plot(results_nn.history['loss'])
plt.plot(results_nn.history['val_loss'])
plt.title("Training History")
plt.ylabel('Loss Function')
plt.xlabel('Epochs')
plt.legend(['Loss training', 'Loss test'])
plt.show()
```

In [58]:
```
# plotar training history graph
plt.figure(figsize = (10, 7))
plt.plot(results_nn.history['accuracy'])
plt.plot(results_nn.history['val_accuracy'])
plt.title("Training History")
plt.xlabel('Épochs')
plt.ylabel('Accuracy')
plt.legend(['training', 'test'])
plt.show()
```

Machine Learning(Randon Forest) vs Deep Learning (Neural Networking)

In [96]:
```
print('Machine Learning (Random Forest) =', (scores).round(4)*100)

print("Deep Learning (Neural NetWorking) = %.2f%%" %((accuracy_test)*100))
```

```
Machine Learning (Random Forest) = 75.18
Deep Learning (Neural NetWorking) = 74.11%
```

In [ ]: