In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model  import Ridge,Lasso,RidgeCV, LassoCV, ElasticNet, ElasticNet
from sklearn.model_selection import train_test_split
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from pandas_profiling import ProfileReport
import seaborn as sns
import pickle
```

In [2]:
```
df = pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/diabetes.cs
```

In [3]:
```
df
```

Out[3]:

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

768 rows × 9 columns

In [4]:
```
ProfileReport(df)
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=22.0), H
TML(value='')))

HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0, max=
1.0), HTML(value='')))

HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0), HTML(val
ue='')))
```

Overview

Dataset statistics

Number of variables	9
Number of observations	768
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	54.1 KiB
Average record size in memory	72.2 B

Variable types

Numeric	8
Categorical	1

Warnings

`Pregnancies` is highly correlated with `Age`	High correlation
`Age` is highly correlated with `Pregnancies`	High correlation
`Pregnancies` is highly correlated with `Age`	High correlation

Out[4]:

In [5]:
```
df['BMI'] = df['BMI'].replace(0 , df['BMI'].mean())
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
df['BloodPressure'] = df['BloodPressure'].replace(0,df['BloodPressure'].mean())
```

In [8]:
```
df['Insulin'] = df['Insulin'].replace(0,df['Insulin'].mean())
```

In [9]:
```
df['Glucose'] = df['Glucose'].replace(0,df['Glucose'].mean())
```

In [10]:
```
df['SkinThickness'] = df['SkinThickness'].replace(0,df['SkinThickness'].mean())
```

In [11]:
```
ProfileReport(df)
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=22.0), H
TML(value='')))

HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0, max=
1.0), HTML(value='')))

HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0), HTML(val
ue='')))
```

Overview

Dataset statistics

Number of variables	9
Number of observations	768
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	54.1 KiB
Average record size in memory	72.2 B

Variable types

Numeric	8
Categorical	1

Warnings

`Pregnancies` is highly correlated with `Age`	High correlation
`SkinThickness` is highly correlated with `BMI`	High correlation
`BMI` is highly correlated with `SkinThickness`	High correlation

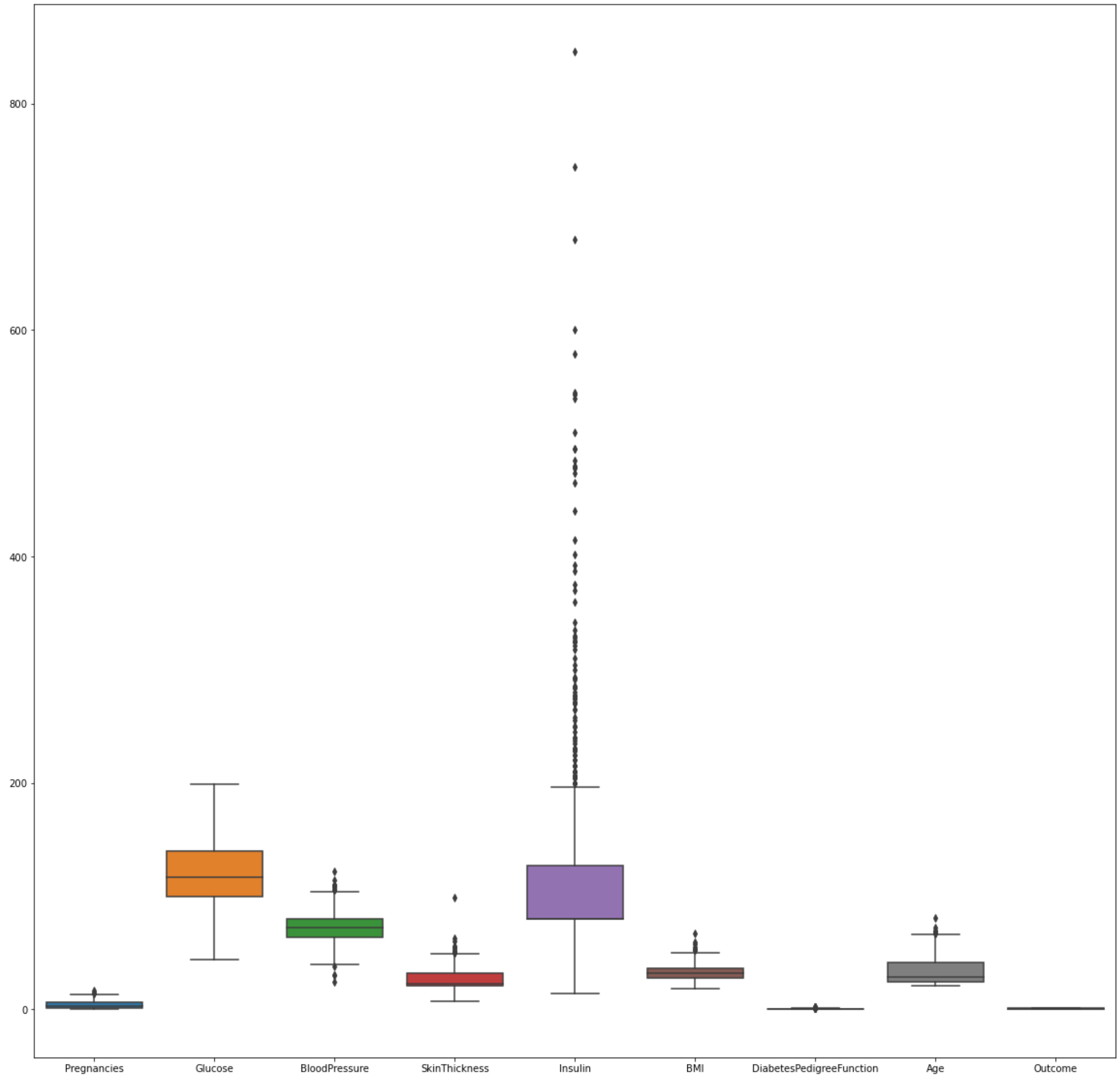
Out[11]:

In [14]:

```
fig ,ax  = plt.subplots(figsize = (20,20))
sns.boxplot(data = df , ax = ax)
```

Out[14]:

```
<AxesSubplot:>
```



In [21]:
```
q = df['Insulin'].quantile(.70)
df_new = df[df['Insulin'] < q]
```

In [22]:
```
df_new
```

Out[22]:

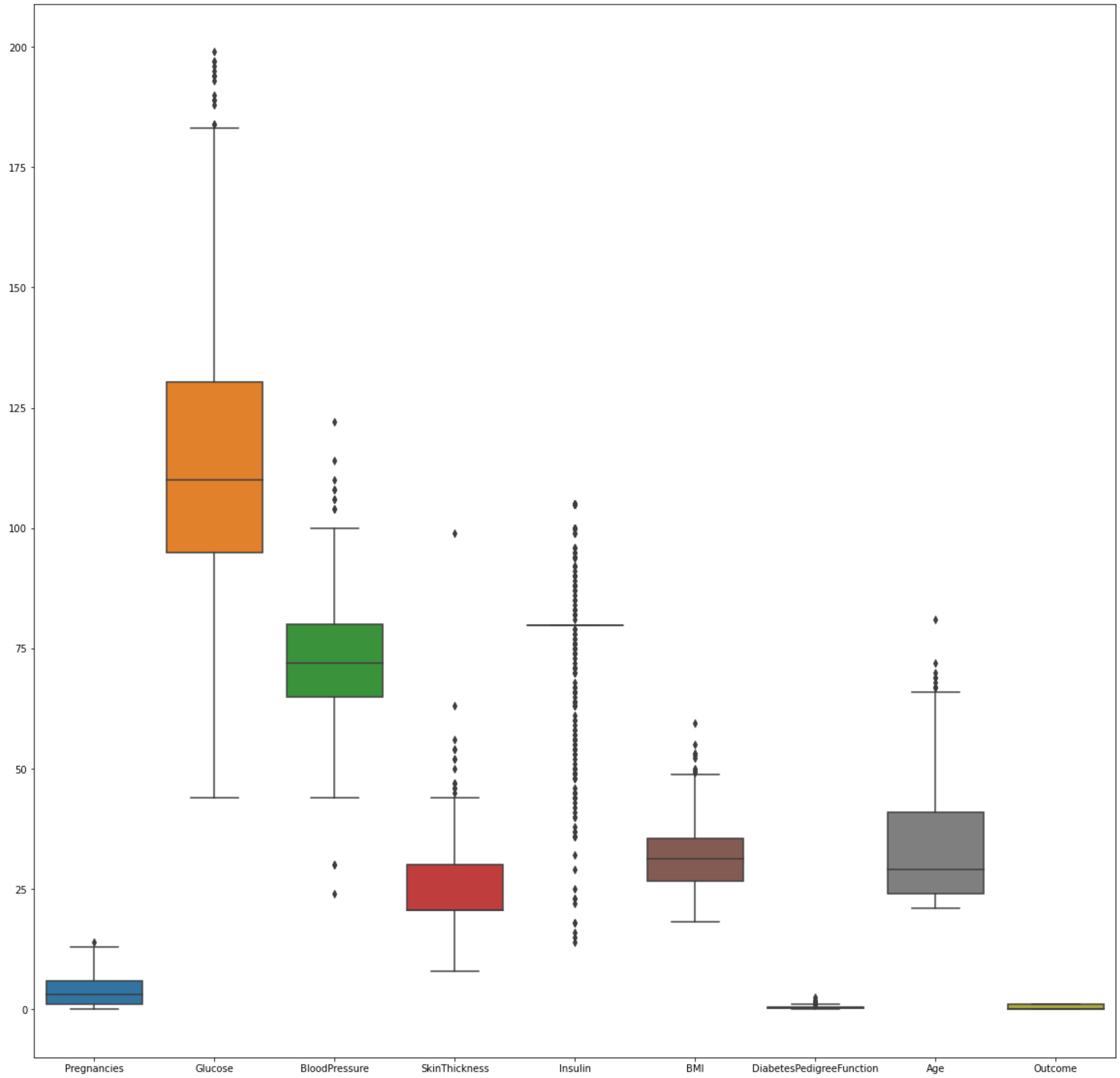
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201
...	...	...	...	...	...	...	...
761	9	170.0	74.0	31.000000	79.799479	44.0	0.403
762	9	89.0	62.0	20.536458	79.799479	22.5	0.142
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315

536 rows × 9 columns

In [23]:

```
fig ,ax  = plt.subplots(figsize = (20,20))
sns.boxplot(data = df_new , ax = ax)
```

Out[23]: <AxesSubplot:>



In [48]:
```
q = df['Pregnancies'].quantile(.98)
df_new = df[df['Pregnancies'] < q]

q = df_new['BMI'].quantile(.99)
df_new = df_new[df_new['BMI']< q]

q = df_new['SkinThickness'].quantile(.99)
df_new = df_new[df_new['SkinThickness']< q]

q = df_new['Insulin'].quantile(.95)
df_new = df_new[df_new['Insulin']< q]

q = df_new['DiabetesPedigreeFunction'].quantile(.99)
df_new = df_new[df_new['DiabetesPedigreeFunction']< q]


q = df_new['Age'].quantile(.99)
df_new = df_new[df_new['Age']< q]
```

In [ ]:
```
def outlier_removal(self,data):
        def outlier_limits(col):
```

```
        Q3, Q1 = np.nanpercentile(col, [75,25])
        IQR= Q3-Q1
        UL= Q3+1.5*IQR
        LL= Q1-1.5*IQR
        return UL, LL

    for column in data.columns:
        if data[column].dtype != 'int64':
            UL, LL= outlier_limits(data[column])
            data[column]= np.where((data[column] > UL) | (data[column] < LL), np.r

    return data
```

In [49]:
```
df_new
```

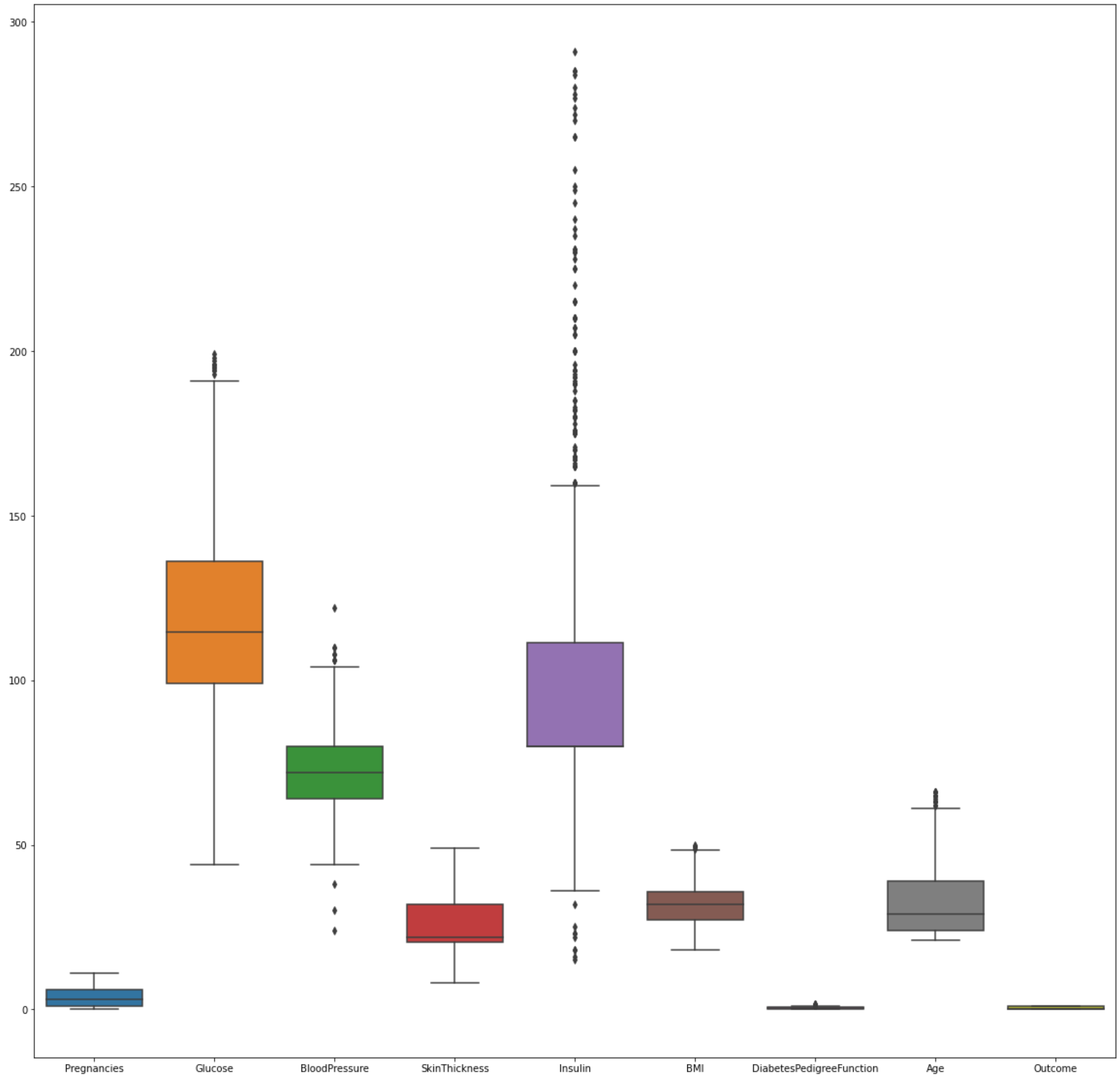
Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201
...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315

674 rows × 9 columns

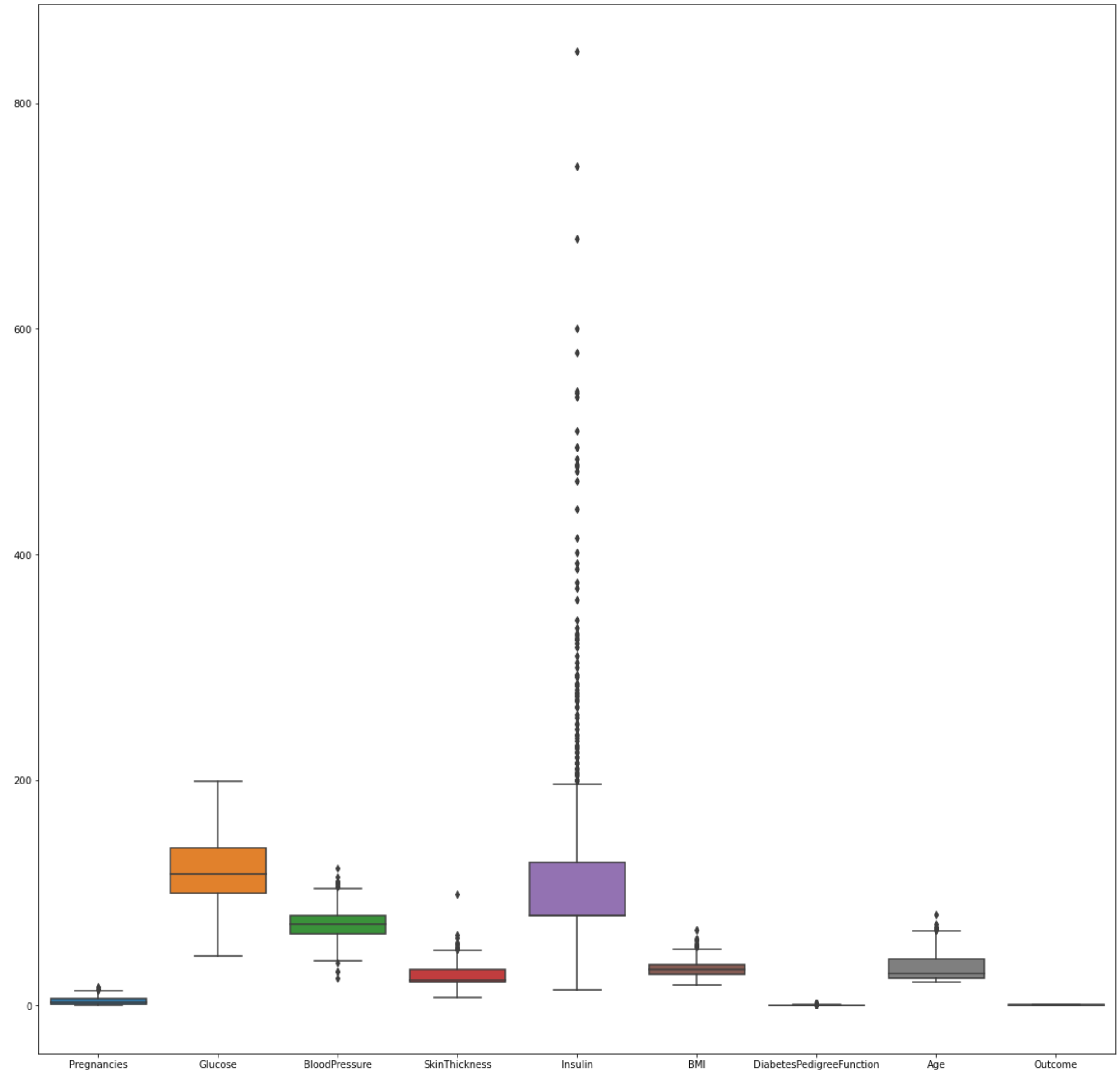
In [50]:
```
fig ,ax  = plt.subplots(figsize = (20,20))
sns.boxplot(data = df_new , ax = ax)
```

Out[50]: <AxesSubplot:>



In [44]:
```
fig ,ax  = plt.subplots(figsize = (20,20))
sns.boxplot(data = df , ax = ax)
```

Out[44]: <AxesSubplot:>



In [51]:
```
ProfileReport(df_new)
```

```
HBox(children=(HTML(value='Summarize dataset'), FloatProgress(value=0.0, max=23.0), H
TML(value='')))

HBox(children=(HTML(value='Generate report structure'), FloatProgress(value=0.0, max=
1.0), HTML(value='')))

HBox(children=(HTML(value='Render HTML'), FloatProgress(value=0.0, max=1.0), HTML(val
ue='')))
```

Overview

Dataset statistics

Number of variables	10
Number of observations	674
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	52.8 KiB
Average record size in memory	80.2 B

Variable types

Numeric	9
Categorical	1

Warnings

`Pregnancies` is highly correlated with `Age`	**High correlation**
`SkinThickness` is highly correlated with `BMI`	**High correlation**
`BMI` is highly correlated with `SkinThickness`	**High correlation**

Out[51]:

In [52]:
```
df_new
```

Out[52]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201
...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315

674 rows × 9 columns

In [54]:

```
y = df_new['Outcome']
y
```

Out[54]:

```
0      1
1      0
2      1
3      0
5      0
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 674, dtype: int64
```

In [55]:

```
X = df_new.drop(columns=['Outcome'])
```

In [56]:

```
 X
```

Out[56]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167
5	5	116.0	74.0	20.536458	79.799479	25.6	0.201
...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315

674 rows × 8 columns

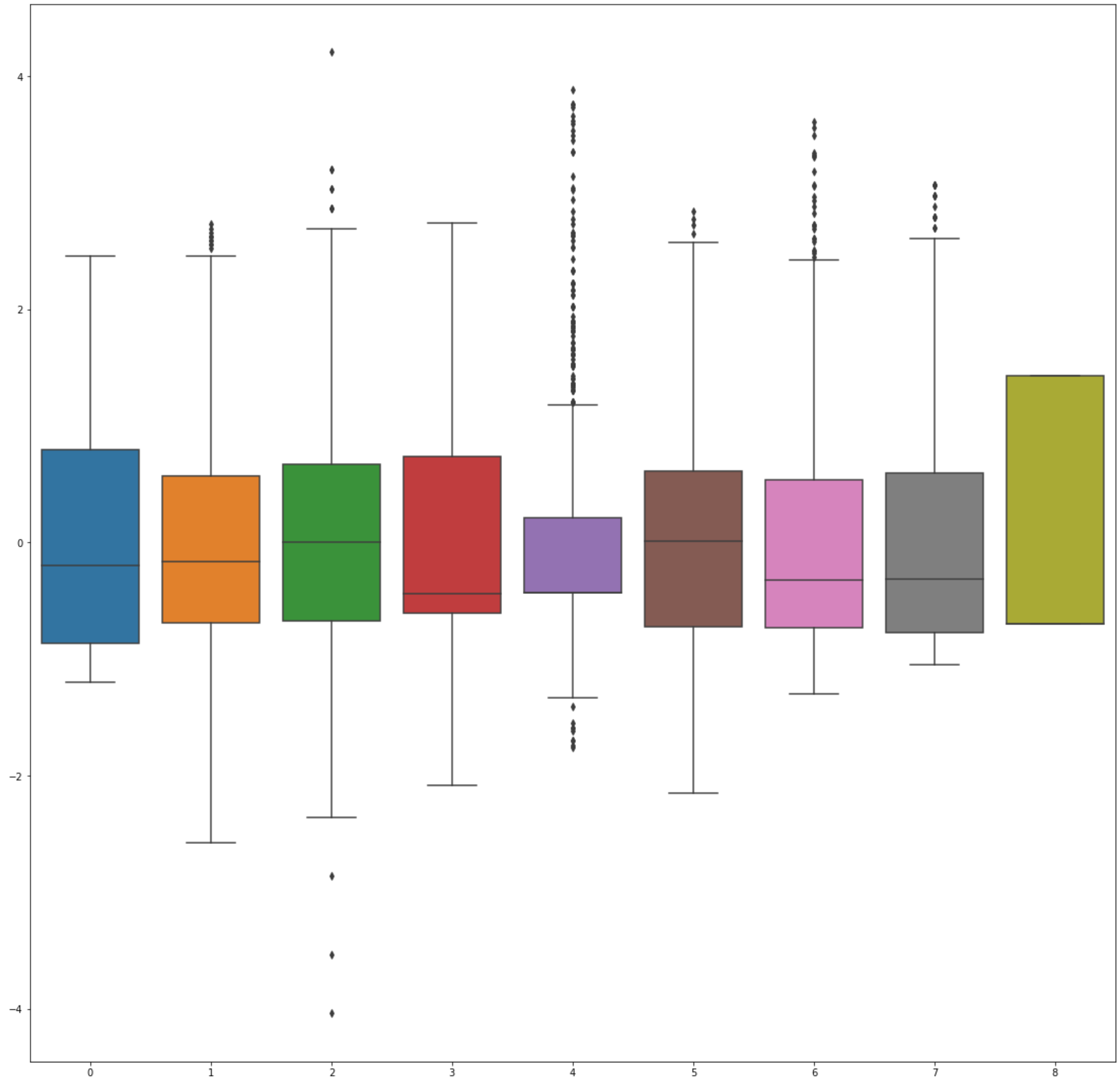
In [61]:

```
scalar = StandardScaler()
ProfileReport(pd.DataFrame(scalar.fit_transform(X)))
X_scaled = scalar.fit_transform(X)
```

In [60]:

```
df_new_scalar = pd.DataFrame(scalar.fit_transform(df_new))
fig ,ax  = plt.subplots(figsize = (20,20))
sns.boxplot(data = df_new_scalar , ax = ax)
```

Out[60]: <AxesSubplot:>



In [62]:
```
X_scaled
```

Out[62]:
```
array([[ 7.96753910e-01,  9.83984062e-01,  4.52611463e-04, ...,
         2.65819648e-01,  6.30484542e-01,  1.60141519e+00],
       [-8.64793539e-01, -1.16977621e+00, -5.04474494e-01, ...,
        -8.31445036e-01, -3.38078670e-01, -1.32706484e-01],
       [ 1.46137289e+00,  2.18051755e+00, -6.72783529e-01, ...,
        -1.34872696e+00,  7.88402456e-01, -4.14369227e-02],
       ...,
       [ 4.64444420e-01,  6.09439465e-02,  4.52611463e-04, ...,
        -8.94145875e-01, -7.10063091e-01, -2.23976046e-01],
       [-8.64793539e-01,  2.31877301e-01, -1.00940160e+00, ...,
        -2.82812694e-01, -3.45097244e-01,  1.32760650e+00],
       [-8.64793539e-01, -8.96282840e-01, -1.67856424e-01, ...,
        -2.35787064e-01, -4.64413001e-01, -8.62862978e-01]])
```

In [63]:
```
y
```

Out[63]:
```
0      1
1      0
2      1
3      0
5      0
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 674, dtype: int64
```

In [64]:
```
def vif_score(x):
    scaler = StandardScaler()
    arr = scaler.fit_transform(x)
    return pd.DataFrame([[x.columns[i], variance_inflation_factor(arr,i)] for i in rar
```

In [66]:
```
vif_score(X)
```

Out[66]:

	FEATURE	VIF_SCORE
0	Pregnancies	1.449056
1	Glucose	1.304263
2	BloodPressure	1.262686
3	SkinThickness	1.470049
4	Insulin	1.271017
5	BMI	1.513160
6	DiabetesPedigreeFunction	1.042300
7	Age	1.662728

In [68]:
```
x_train, x_test, y_train, y_test = train_test_split(X_scaled , y , test_size = .20 , r
```

In [69]:
```
x_train
```

Out[69]:
```
array([[-0.86479354,  0.19769063, -1.85094678, ...,  0.21879402,
         1.80609569, -0.40651517],
       [ 2.45830136,  1.22329076,  0.33707068, ...,  0.21879402,
         3.1782269 ,  1.69268475],
       [ 0.13213493,  0.91561072,  1.09446134, ..., -0.47091521,
        -0.90658316, -0.49778473],
       ...,
       [-0.86479354, -0.28092276,  1.17861586, ..., -0.28281269,
        -1.06801036, -0.86286298],
       [ 1.46137289,  0.02675728,  0.50537972, ..., -1.08224839,
        -0.13454002,  2.87918905],
       [-0.86479354, -0.75953616, -0.33616546, ..., -0.73739378,
         2.27283086, -0.95413254]])
```

In [70]:
```
x_test
```

Out[70]:
```
array([[ 1.32134931e-01, -1.44176079e-01,  4.52611463e-04, ...,
        -4.70915211e-01, -2.50346495e-01,  1.23633694e+00],
       [-1.19710303e+00,  6.76304024e-01, -8.41092564e-01, ...,
        -1.53682948e+00, -8.43415997e-01, -1.04540210e+00],
       [ 2.45830136e+00,  8.13050708e-01,  1.85185200e+00, ...,
         7.36075942e-01, -6.78479508e-01,  1.69268475e+00],
       ...,
       [ 7.96753910e-01, -5.54416131e-01, -5.04474494e-01, ...,
        -1.19197486e+00, -6.96025943e-01, -3.15245608e-01],
       [-5.32484049e-01, -1.20396288e+00, -2.43135269e-01, ...,
         1.38528976e-02, -5.03015158e-01, -1.04540210e+00],
       [-8.64793539e-01, -1.64838960e+00, -2.01925581e+00, ...,
        -1.80330804e+00, -4.36338705e-01, -9.54132539e-01]])
```

In [77]:
```
x_test[0]
```

Out[77]:
```
array([ 1.32134931e-01, -1.44176079e-01,  4.52611463e-04, -6.09921498e-01,
       -4.34192020e-01, -4.70915211e-01, -2.50346495e-01,  1.23633694e+00])
```

In [106…
```
logr_liblinear = LogisticRegression(verbose=1,solver='liblinear')
```

In [107…
```
logr_liblinear.fit(x_train,y_train )
```

```
[LibLinear]
```

Out[107]:
```
LogisticRegression(solver='liblinear', verbose=1)
```

In [100…
```
logr.predict_proba([x_test[1]])
```

Out[100]:
```
array([[0.91450958, 0.08549042]])
```

In [97]:
```
logr.predict([x_test[1]])
```

Out[97]:
```
array([0], dtype=int64)
```

In [92]:
```
logr.predict_log_proba([x_test[1]])
```

Out[92]:
```
array([[-0.08742167, -2.48040456]])
```

In [86]:
```
type(y_test)
```

Out[86]:
```
pandas.core.series.Series
```

In [91]:
```
y_test.iloc[1]
```

Out[91]:
```
0
```

In [88]:
```
y_test
```

Out[88]:
```
406    1
511    0
24     1
751    0
689    1
      ..
3      0
469    0
587    0
60     0
97     0
Name: Outcome, Length: 135, dtype: int64
```

In [101…
```
logr = LogisticRegression(verbose=1)
```

In [102…
```
logr.fit(x_train,y_train)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s finished
```

Out[102]:
```
LogisticRegression(verbose=1)
```

In [103…
```
logr_liblinear
```

Out[103]:
```
LogisticRegression(solver='liblinear', verbose=1)
```

In [104…
```
logr
```

Out[104]:
```
LogisticRegression(verbose=1)
```

In [109…
```
y_pred_liblinear = logr_liblinear.predict(x_test)
y_pred_liblinear
```

Out[109]:
```
array([0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0], dtype=int64)
```

In [111…
```
y_pred_default = logr.predict(x_test)
```

In [112…
```
y_pred_default
```

Out[112]:
```
array([0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0], dtype=int64)
```

In [113…
```
confusion_matrix(y_test,y_pred_liblinear)
```

Out[113]:
```
array([[77, 10],
       [24, 24]], dtype=int64)
```

In [114…
```
confusion_matrix(y_test,y_pred_default)
```

Out[114]:
```
array([[77, 10],
       [24, 24]], dtype=int64)
```

In [119…
```
def model_eval(y_true,y_pred):
    tn, fp, fn, tp = confusion_matrix(y_test,y_pred).ravel()
    accuracy=(tp+tn)/(tp+tn+fp+fn)
    precision=tp/(tp+fp)
    recall=tp/(tp+fn)
    specificity=tn/(fp+tn)
    F1_Score = 2*(recall * precision) / (recall + precision)
    result={"Accuracy":accuracy,"Precision":precision,"Recall":recall,'Specficity':spe
    return result
model_eval(y_test,y_pred_liblinear)
```

Out[119]:
```
{'Accuracy': 0.7481481481481481,
 'Precision': 0.7058823529411765,
 'Recall': 0.5,
 'Specficity': 0.8850574712643678,
 'F1': 0.5853658536585366}
```

In [120…
```
model_eval(y_test,y_pred_default)
```

Out[120]:
```
{'Accuracy': 0.7481481481481481,
 'Precision': 0.7058823529411765,
 'Recall': 0.5,
 'Specficity': 0.8850574712643678,
 'F1': 0.5853658536585366}
```

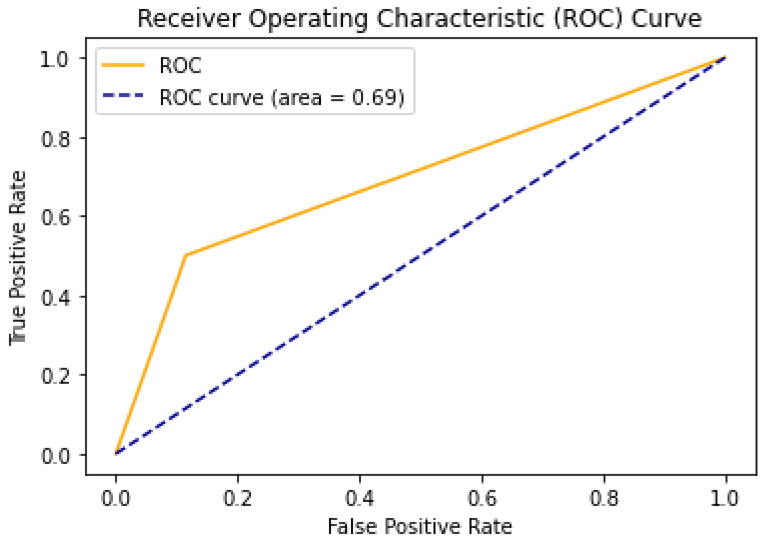
In [127…
```
auc = roc_auc_score(y_test,y_pred_liblinear)
```

In [128…
```
roc_auc_score(y_test,y_pred_default)
```

Out[128]:
```
0.692528735632184
```

In [129…
```
fpr, tpr, thresholds  = roc_curve(y_test,y_pred_liblinear)
```

In [130…
```
plt.plot(fpr, tpr, color='orange', label='ROC')
plt.plot([0, 1], [0, 1], color='darkblue', linestyle='--',label='ROC curve (area = %0.
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
```



In [ ]:
```
#logist regression task

https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+system+based+on+Multisens

Task Logistic Regression
~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. WAP to read folder name and make a label in the csv with folder name
2. Remove unneccesarry info in Automated way
3. No other algorithm must be used other than Logistic Regression
4. Try to utilize multiple solvers and make multiple models
5. Provide the best models
6. EDA and all must be done accordingly
Note: No manual approaches will be appreciated
```