Statistical Analysis of Breast Cancer Data and EDA

```
%%HTML
<script src="./require.js"></script>
```

```
<IPython.core.display.HTML object>
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Importing data
data = pd.read_csv('./Breast Cancer Detection.csv')
del data['Unnamed: 32']
```

Performing Data Wrangling Operations

```
X = data.iloc[:, 2:].values
y = data.iloc[:, 1].values
```

Replacing Null Values if any.

```
data.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
```

```
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

Total number of counts for Malignant (M) and Benign (B)

```
data['diagnosis'].value_counts()

diagnosis
B    357
M    212
Name: count, dtype: int64
```

A quick description of data

```
data.describe()

                 id  radius_mean  texture_mean  perimeter_mean
area_mean
count  5.690000e+02   569.000000    569.000000      569.000000
569.000000  \
mean   3.037183e+07    14.127292     19.289649       91.969033
654.889104
std    1.250206e+08     3.524049      4.301036       24.298981
351.914129
min    8.670000e+03     6.981000      9.710000       43.790000
143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000
420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000
551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000
782.700000
max    9.113205e+08    28.110000     39.280000      188.500000
2501.000000

       smoothness_mean  compactness_mean  concavity_mean  concave
points_mean
count       569.000000        569.000000      569.000000
569.000000  \
mean          0.096360          0.104341        0.088799
0.048919
```

```
std           0.014064         0.052813        0.079720
0.038803
min           0.052630         0.019380        0.000000
0.000000
25%           0.086370         0.064920        0.029560
0.020310
50%           0.095870         0.092630        0.061540
0.033500
75%           0.105300         0.130400        0.130700
0.074000
max           0.163400         0.345400        0.426800
0.201200

       symmetry_mean  ...  radius_worst  texture_worst
perimeter_worst
count     569.000000  ...    569.000000     569.000000
569.000000  \
mean        0.181162  ...     16.269190      25.677223
107.261213
std         0.027414  ...      4.833242       6.146258
33.602542
min         0.106000  ...      7.930000      12.020000
50.410000
25%         0.161900  ...     13.010000      21.080000
84.110000
50%         0.179200  ...     14.970000      25.410000
97.660000
75%         0.195700  ...     18.790000      29.720000
125.400000
max         0.304000  ...     36.040000      49.540000
251.200000

        area_worst  smoothness_worst  compactness_worst
concavity_worst
count   569.000000        569.000000         569.000000
569.000000  \
mean    880.583128          0.132369           0.254265
0.272188
std     569.356993          0.022832           0.157336
0.208624
min     185.200000          0.071170           0.027290
0.000000
25%     515.300000          0.116600           0.147200
0.114500
50%     686.500000          0.131300           0.211900
0.226700
75%    1084.000000          0.146000           0.339100
0.382900
max    4254.000000          0.222600           1.058000
```

```
1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

[8 rows x 31 columns]
```

Categorical Observation to Detect Breast Cancer across Radius, Perimeter, Area, Smoothness, Concavity and Worst Fractal Dimensions.

```python
import plotly.express as px
import plotly.io as pio
pio.renderers.default='notebook'

px.scatter_matrix(data, dimensions=["radius_mean", "perimeter_mean",
"area_mean", "smoothness_mean", "concavity_mean",
"fractal_dimension_worst"], color="diagnosis", title="Malignancy
Analysis for Given Features.")
```

Exploring Distribution of Cases over Fractal Dimensions. (M = Malignant, B = Benign)

```python
fig = px.box(data, x='diagnosis', y= 'fractal_dimension_worst')

fig.show()
```

Using the data to create a Deep Learning Model to predict breast cancer with given features.

```python
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
labelencoder_X_1 = LabelEncoder()
y = labelencoder_X_1.fit_transform(y)

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.1, random_state = 0)

#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Using Keras and Tensorflow package to develop CNN model

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout

# Initialising the ANN
classifier = Sequential()
```

Developing First Input Layer using RELU activation

```
# Adding the input layer and the first hidden layer
classifier.add(Dense(16, activation='relu'))
# Adding dropout to prevent overfitting
classifier.add(Dropout(0.1))
```

Developing Second Input Layer Using RELU Activation

```
# Adding the second hidden layer
classifier.add(Dense(16, activation='relu'))
# Adding dropout to prevent overfitting
classifier.add(Dropout(0.1))
```

Using sigmoid function as gradient descent algorithm for backward propagation.

```
classifier.add(Dense(1, activation='sigmoid'))
```

Optimizing the Classifier with Adam algorithm and reducing loss rate with Binary Crossentropy.

```
classifier.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

Fitting the classifier with training sets.

```
classifier.fit(X_train, y_train, batch_size=100)

6/6 [==============================] - 0s 2ms/step - loss: 0.1987 -
accuracy: 0.9551

<keras.callbacks.History at 0x14804c42740>
```

Running iterations to optimize prediction accuracy

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)

2/2 [==============================] - 0s 2ms/step
```

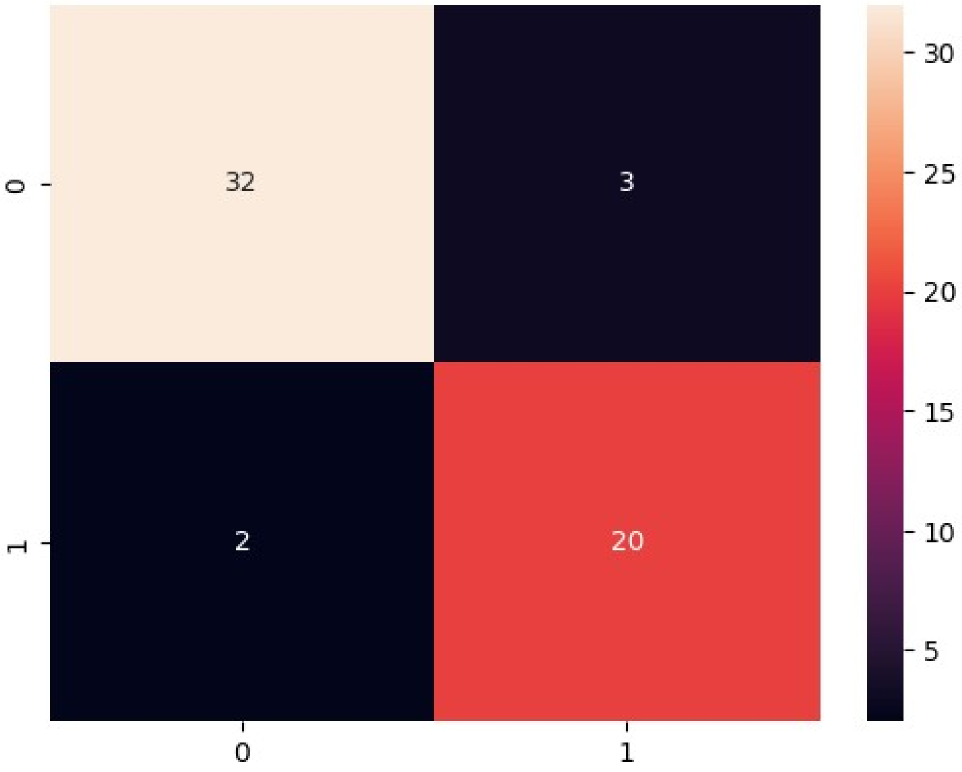
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
print("Our accuracy is {}%".format(((cm[0][0] + cm[1][1])/57)*100))
```

```
Our accuracy is 91.22807017543859%
```

```
sns.heatmap(cm,annot=True)
```

```
<Axes: >
```


32
3
2
20
0
1
0
1
30
25
20
15
10
5