Breast Cancer Data Analysis

```
In [2]: #importing the dataset
    df = pd.read_csv('C:/Users/zizhe/Desktop/Leo Zhao/Breast Cancer Prediction/data.csv')
    df.head()
```

nean	concavity_mean	concave points_mean	•••	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst
7760	0.3001	0.14710		17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654
7864	0.0869	0.07017		23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860
5990	0.1974	0.12790		25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430
8390	0.2414	0.10520		26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575
3280	0.1980	0.10430		16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625

Data Preprocessing Part 1

```
In [3]: # dropping unnecessary columns
    df.drop([32','id'],axis=1,inplace=True)
In [4]: df.head()
```

Out[4]:		diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean
	0	М	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710
	1	М	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017
	2	М	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790
	3	М	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520
	4	М	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430

5 rows × 31 columns

In [5]: #checking for the missing values
 df.isnull().sum()

```
diagnosis
                           0
radius_mean
                           0
                           0
texture_mean
perimeter_mean
                           0
area_mean
smoothness_mean
compactness mean
                           0
concavity mean
                           0
concave points mean
symmetry mean
fractal dimension mean
                          0
radius se
                           0
texture se
                          0
perimeter_se
                           0
                           0
area se
smoothness se
compactness se
concavity_se
concave points se
symmetry_se
fractal_dimension_se
radius_worst
                           0
texture_worst
                           0
perimeter worst
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
```

In [6]: #checking the data types of the columns
 df.dtypes

```
diagnosis
                                    object
Out[6]:
        radius_mean
                                   float64
        texture mean
                                   float64
        perimeter_mean
                                   float64
                                   float64
        area_mean
        smoothness_mean
                                   float64
        compactness mean
                                   float64
        concavity mean
                                   float64
        concave points mean
                                   float64
        symmetry mean
                                   float64
        fractal dimension mean
                                   float64
        radius se
                                   float64
        texture se
                                   float64
        perimeter se
                                   float64
        area se
                                   float64
        smoothness se
                                   float64
        compactness se
                                   float64
        concavity_se
                                   float64
        concave points se
                                   float64
        symmetry se
                                   float64
        fractal dimension se
                                   float64
        radius_worst
                                   float64
        texture worst
                                   float64
        perimeter worst
                                   float64
        area_worst
                                   float64
        smoothness_worst
                                   float64
        compactness worst
                                   float64
        concavity_worst
                                   float64
        concave points_worst
                                   float64
        symmetry worst
                                   float64
        fractal dimension worst
                                   float64
        dtype: object
```

In [19]: # checking the data description
 df.describe()

_				
Ou	+	11	a 1	
vu		1 4	2	

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	points_mean	symm
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	

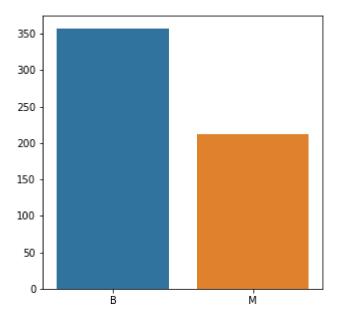
concave

8 rows × 30 columns

Exploratory Data Analysis

```
In [17]: # bar plot for the number of diagnosis
    plt.figure(figsize=(5,5)) #plt.figure(figsize=(5, 5)):
    sns.barplot(x=df['diagnosis'].value_counts(),y=df['diagnosis'].value_counts().values)
```

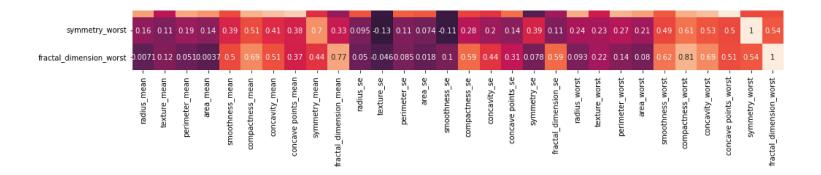
Out[17]: <AxesSubplot:>



```
In [21]: # create a heatmap to check the correlation.
   plt.figure(figsize=(20,20))
   sns.heatmap(df.cor(),annot=True)
```

Out[21]: <AxesSubplot:>

0.32 1 0.99 0.17 0.51 0.68 0.82 0.15 -0.31 texture_mean - 0.32 1 0.33 0.32 0.023 0.24 0.3 0.29 0.071-0.076 0.28 0.39 0.28 0.26 0.0066 0.19 0.14 0.16 0.00910.054 0.35 0.91 0.36 0.34 0.078 0.28 0.3 0.3 0.1 0.12 perimeter mean - 1 0.33 1 0.99 0.21 0.56 0.72 0.85 0.18 0.26 0.69 0.087 0.69 0.74 0.2 0.25 0.23 0.41 0.0820.005 0.97 0.3 0.97 0.94 0.15 0.46 0.56 0.77 0.19 0.051 area mean - 0.99 0.32 0.99 1 0.18 0.5 0.69 0.82 0.15 0.28 0.73 0.066 0.73 0.8 0.17 0.21 0.21 0.37 0.072 0.02 0.96 0.29 0.96 0.96 0.12 0.39 0.51 0.72 0.14 0.003 smoothness mean - 0.17 -0.023 0.21 0.18 1 0.66 0.52 0.55 0.56 0.58 0.3 0.068 0.3 0.25 0.33 0.32 0.25 0.38 0.2 0.28 0.21 0.036 0.24 0.21 <mark>0.81</mark> 0.47 0.43 0.5 0.39 0.5 compactness mean - 0.51 0.24 0.56 0.5 0.66 1 0.88 0.83 0.6 0.57 0.5 0.046 0.55 0.46 0.14 0.74 0.57 0.64 0.23 0.51 0.54 0.25 0.59 0.51 0.57 0.87 0.82 0.82 0.51 0.64 0.3 0.72 0.69 0.52 0.88 1 0.92 0.5 0.34 0.63 0.076 0.66 0.62 0.099 0.67 0.69 0.68 0.18 0.45 0.69 0.3 0.73 0.68 0.45 0.75 0.88 0.86 0.41 0.51 concave points mean - 0.82 0.29 0.85 0.82 0.55 0.83 0.92 1 0.46 0.17 0.7 0.021 0.71 0.69 0.028 0.49 0.44 0.62 0.095 0.26 0.83 0.29 0.86 0.81 0.45 0.67 0.75 0.91 0.38 0.37 symmetry mean - 0.15 0.071 0.18 0.15 0.56 0.6 0.5 0.46 1 0.48 0.3 0.13 0.31 0.22 0.19 0.42 0.34 0.39 0.45 0.33 0.19 0.091 0.22 0.18 0.43 0.47 0.43 0.43 0.7 fractal dimension mean -0.31-0.076-0.26-0.28 0.58 0.57 0.34 0.17 0.48 1 .000110.16 0.04 -0.09 0.4 0.56 0.45 0.34 0.35 0.69 0.25 0.051-0.21 -0.23 0.5 0.46 0.35 0.18 0.33 0.77 68 0.28 0.69 0.73 0.3 0.5 0.63 0.7 0.300001 1 0.21 0.97 0.95 0.16 0.36 0.33 0.51 0.24 0.23 0.72 0.19 0.72 0.75 0.14 0.29 0.38 0.53 0.095 0.05 texture se -0.097 0.39 -0.087-0.0660.068 0.046 0.076 0.021 0.13 0.16 0.21 1 0.22 0.11 0.4 0.23 0.19 0.23 0.41 0.28 -0.11 0.41 -0.1 -0.083-0.074-0.092-0.069-0.12 -0.13 -0.046 0.28 0.69 0.73 0.3 0.55 0.66 0.71 0.31 0.04 0.97 0.22 1 0.94 0.15 0.42 0.36 0.56 0.27 0.24 0.7 0.2 0.72 0.73 0.13 0.34 0.42 0.55 0.11 0.085 area_se - 0.74 0.26 0.74 0.8 0.25 0.46 0.62 0.69 0.22 0.09 0.95 0.11 0.94 1 0.075 0.28 0.27 0.42 0.13 0.13 0.76 0.2 0.76 0.81 0.13 0.28 0.39 0.54 0.074 0.018 smoothness se -0.220.0066 -0.2 -0.17 -0.33 -0.14 -0.099.0.028 -0.19 -0.4 -0.16 -0.4 -0.15 -0.075 -1 -0.34 -0.27 -0.33 -0.41 -0.43 -0.23 -0.075 -0.22 -0.18 -0.31 -0.0560.058 -0.1 -0.11 -0.11 -0.15 compactness se - 0.21 0.19 0.25 0.21 0.32 0.74 0.67 0.49 0.42 0.56 0.36 0.23 0.42 0.28 0.34 1 0.8 0.74 0.39 0.8 0.2 0.14 0.26 0.2 0.23 0.68 0.64 0.48 0.28 0.59 concavity se - 0.19 0.14 0.23 0.21 0.25 0.57 0.69 0.44 0.34 0.45 0.33 0.19 0.36 0.27 0.27 0.8 1 0.77 0.31 0.73 0.19 0.1 0.23 0.19 0.17 0.48 0.66 0.44 0.2 0.44 concave points se - 0.38 0.16 0.41 0.37 0.38 0.64 0.68 0.62 0.39 0.34 0.51 0.23 0.56 0.42 0.33 0.74 0.77 1 0.31 0.61 0.36 0.087 0.39 0.34 0.22 0.45 0.55 0.6 0.14 0.31 symmetry se - 0.1 0.00910.0820.072 0.2 0.23 0.18 0.095 0.45 0.35 0.24 0.41 0.27 0.13 0.41 0.39 0.31 0.31 1 0.37 -0.13 -0.07 -0.1 -0.11 -0.013 0.06 0.037 -0.03 0.39 0.078 fractal dimension se -0.0430.0540.00550.02 0.28 0.51 0.45 0.26 0.33 0.69 0.23 0.28 0.24 0.13 0.43 0.8 0.73 0.61 0.37 1 0.0370.00320.0010.023 0.17 0.39 0.38 0.22 0.11 0.51 radius worst - 0.97 0.35 0.97 0.96 0.21 0.54 0.69 0.83 0.19 0.25 0.72 0.11 0.7 0.76 0.23 0.2 0.19 0.36 0.13 0.037 1 0.36 0.99 0.98 0.22 0.48 0.57 0.79 0.24 0.093 texture worst - 0.3 0.91 0.3 0.29 0.036 0.25 0.3 0.29 0.091 0.051 0.19 0.41 0.2 0.2 0.075 0.14 0.1 0.087 0.0770 0.032 0.36 1 0.37 0.35 0.23 0.36 0.37 0.36 0.23 0.22 perimeter_worst - 0.97 0.36 0.97 0.96 0.24 0.59 0.73 0.86 0.22 0.21 0.72 0.1 0.72 0.76 0.22 0.26 0.23 0.39 0.1 0.001 0.99 0.37 1 0.98 0.24 0.53 0.62 0.82 0.27 0.14 area_worst - 0.94 0.34 0.94 0.96 0.21 0.51 0.68 0.81 0.18 0.23 0.75 0.083 0.73 0.81 0.18 0.2 0.19 0.34 0.11 0.023 0.98 0.35 0.98 1 0.21 0.44 0.54 0.75 0.21 0.08 smoothness worst - 0.12 0.078 0.15 0.12 0.81 0.57 0.45 0.45 0.45 0.43 0.5 0.14 0.074 0.13 0.13 0.31 0.23 0.17 0.22 0.013 0.17 0.22 0.23 0.24 0.21 1 0.57 0.52 0.55 0.49 0.63 compactness worst - 0.41 0.28 0.46 0.39 0.47 0.87 0.75 0.67 0.47 0.46 0.29 0.092 0.34 0.28 0.056 0.68 0.48 0.45 0.06 0.39 0.48 0.36 0.53 0.44 0.57 1 0.89 0.8 0.61 0.81 concavity worst - 0.53 0.3 0.56 0.51 0.43 0.82 0.88 0.75 0.43 0.35 0.38 0.069 0.42 0.39 0.058 0.64 0.66 0.55 0.037 0.38 0.57 0.37 0.62 0.54 0.52 0.89 1 0.86 0.53 0.62 concave points worst - 0.74 0.3 0.77 0.72 0.5 0.82 0.86 0.91



Train Test Split

```
In [22]: #split a DataFrame 'df' into training and testing datasets for machine learning purposes.

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(df.drop(['diagnosis']),df['diagnosis'],test_size,random_state=42)
```

Using Decision Tree Classifier

```
In [23]: from sklearn.tree import DecisionTreeClassifier
    dtree = DecisionTreeClassifier()
    dtree.fit(X_train,y_train)

Out[23]: DecisionTreeClassifier()

In [24]: #predicting the diagnosis
    y_pred = dtree.predict(X_test)
```

Model Evaluation

```
In [25]: # printing samples from predicted and actual values
    print('Predicted values: ',y_pred[:10])
    print('Actual values:y_test[:10])
```

```
Predicted values: ['B' 'M' 'M' 'B' 'B' 'M' 'M' 'M'
         Actual values: 204
         70
         131
                Μ
         431
         540
         567
                Μ
         369
                Μ
         29
         81
                В
         477
         Name: diagnosis, dtype: object
         # model evaluation
In [26]:
         print(dtree.score(X_test,y_test))
         0.9415204678362573
         # 0.94 means that the model is making correct predictions for approximately 94% of the samples in the test data.
```

Using logistic regression

Model Evaluation

```
In [31]: # printing samples from predicted and actual values
         print('Predicted values: ',yhat[:10])
         print('Actual values: ',y_test[:10])
         # If the predicted values closely match the actual values, it's a positive sign that your model is making accurate pred
         Predicted values: ['B' 'M' 'M' 'B' 'B' 'M' 'M' 'M' 'B' 'B']
         Actual values: 204
         70
                Μ
         131
                Μ
         431
                В
         540
                В
         567
                Μ
         369
                Μ
         29
                Μ
         81
         477
         Name: diagnosis, dtype: object
         #The numbers "204" and "477" you see in the output are actually indices or row numbers from your test dataset. In the c
        # model evaluation
In [32]:
         print(logmodel.score(X test,y test))
         #logmodel.score(X test, y test), calculates and prints the accuracy of the logistic regression model (logmodel) on the
         0.9707602339181286
         # Conclusion
         # From both the models we can see that the accuracy is 94% and 97% respectively. But we can see that the recall value f
```