Breast Cancer Project
YunTzu Yu

Introduction :

Problem:

I was diagnosed with breast cancer at the age of 27 three years ago, so I'm curious what kind of symptom that we get from examinations could be a sign of breast cancer. I would like to use the dataset obtained from this website to predict whether the patient has breast cancer using the features provided and see how accurate the model that could predict correctly based on the attribute.
Here we try to build a model using features as input and labeled diagnosis Benign or Malignant as output. Afterward, we could check the accuracy of the model through a test dataset.

Source:

The dataset is publically available on UCI Machine Learning Repository:

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Dataset Introduction :

The dataset includes 569 observations and 32 features.
Since our research question is what factors could be a signal that people are more likely to get breast cancer,I'll use Diagnosis as the independent variable here.

Variable Information:

1) ID number

2) Diagnosis (M = malignant, B = benign) 3-32)

Ten real-valued features are computed for each cell nucleus:
a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry

j) fractal dimension ("coastline approximation" - 1)

Research Questions :

Given a fine needle aspiration (FNA), we could obtain a small amount of breast tissue to see if the tissue is benign or malignant.
We have the data that consists of a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

So we try to see what characteristics are that a malignant tissue would perform, so we could build the model with these characteristics to make the prediction.

Data Cleaning

Here we'll do some data cleaning
First of all, we'll load our data and see that there are 32 attributes and 569 data points.

```
df = pd.read_csv('wdbc.data', header=None)
df
```

	0	1	2	3	4	5	6	7	8	9	...	22	23	24	25	26	27	28	29	30
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17.33	184.60	2019.0	0.16220	0.66560	0.7119	0.2654	0.4601
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23.41	158.80	1956.0	0.12380	0.18660	0.2416	0.1860	0.2750
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25.53	152.50	1709.0	0.14440	0.42450	0.4504	0.2430	0.3613
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26.50	98.87	567.7	0.20980	0.86630	0.6869	0.2575	0.6638
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16.67	152.20	1575.0	0.13740	0.20500	0.4000	0.1625	0.2364
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30.37	59.16	268.6	0.08996	0.06444	0.0000	0.0000	0.2871

569 rows × 32 columns

We'll do the bivariate analysis for the data to see whether the patient is diagnosed with breast cancer or not, so we need to transform the data type of diagnosis variables from character into numerical one by applying LabelBinarizar function in sklearn package.

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
y1 = lb.fit_transform(y).ravel()
y1
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1,
```

EDA

After cleaning the data, we'll do the exploratory data analysis here. First of all, we'll check the statistical information of the dataset and visualize it to better understand what the properties contained in our dataset .
From the table, we can see the difference of range among each variable is large.Some range of variable is within decimal points and some is in hundred digit so we might need to scale it.

```
x.describe().T
```

	count	mean	std	min	25%	50%	75%	max
mean radius	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
mean texture	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
mean perimeter	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
mean area	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
mean smoothness	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
mean compactness	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
mean concavity	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
mean concave points	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
mean symmetry	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
mean fractal dimension	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
radius error	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
texture error	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
perimeter error	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000

Then we can check the data type and we can see that all dependent variables are all floats. So we would have to do the correlation analysis to filter out what variables are highly correlated .

```
x.dtypes
```

```
mean radius                float64
mean texture               float64
mean perimeter             float64
mean area                  float64
mean smoothness            float64
mean compactness           float64
mean concavity             float64
mean concave points        float64
mean symmetry              float64
mean fractal dimension     float64
radius error               float64
texture error              float64
perimeter error            float64
```

Here we check if there are any null values in the dataset, we can see that there is no null value in this dataset.

```
df.isnull().sum()
```

```
id                         0
diagnosis                  0
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
```

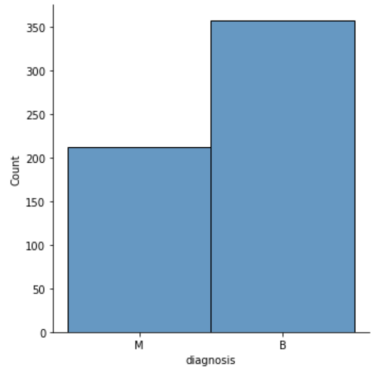
We can check the ratio between positive and negative for breast cancer in the dataset by applying a seaborn package to plot it out.
We can see there are 357 who were diagnosed with breast cancer in our dataset.

```
df["diagnosis"].value_counts()
```

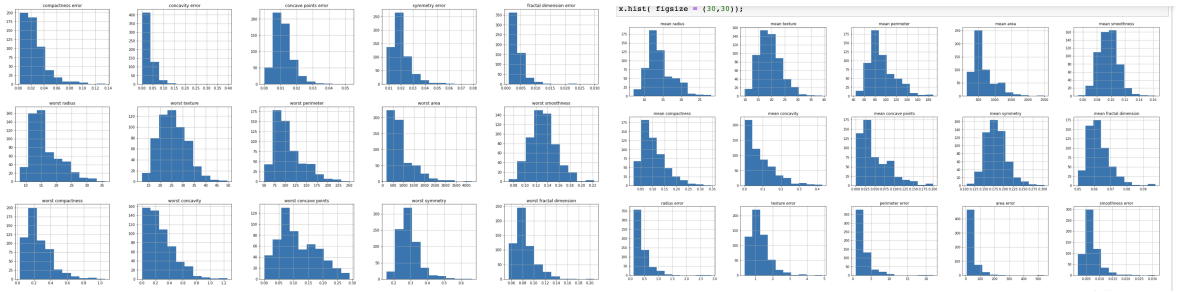
```
B    357
M    212
Name: diagnosis, dtype: int64
```

```
import seaborn as sns
sns.displot(df["diagnosis"])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd30397a340>
```

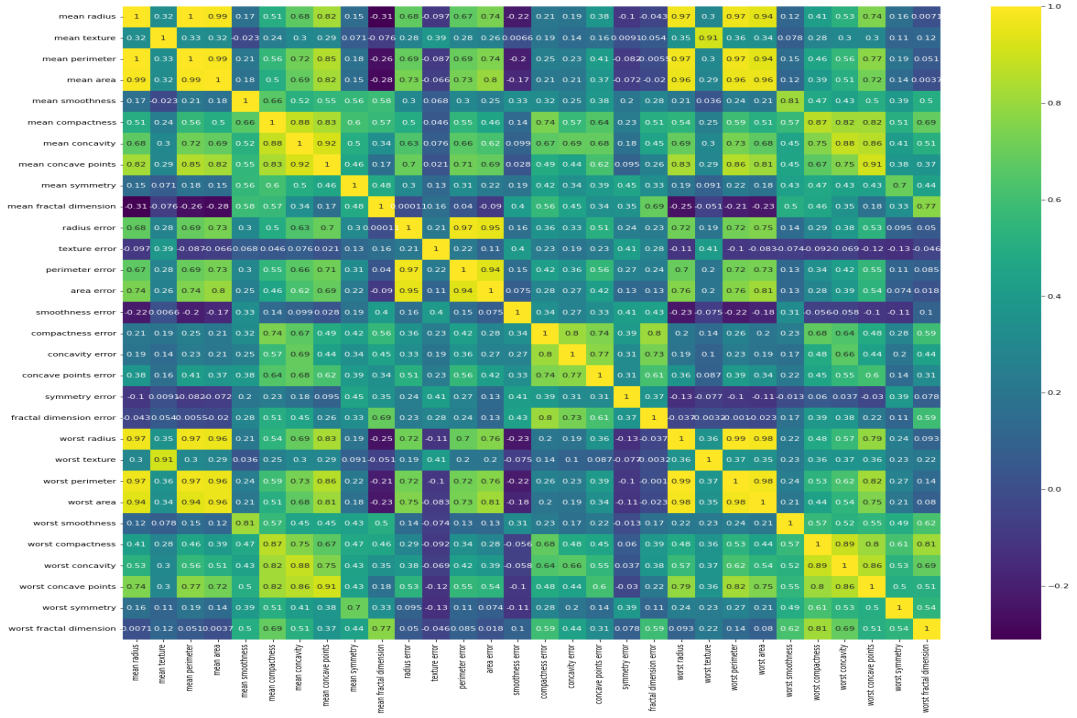


Then we could plot histgram of the dependent variables to see its distribution, here we can see that some variables just as “mean compactness”,“mean concavity”
“radius error” , “perimeter error” ,“area error” ,“concavity error” ,“facial dimension error”,“compactness error”, these variables have an exponential distribution.
Most variables tend to be right-skewed.



Then, we could check the correlation of the variables through heatmap. We can see from the graph that these two variables as following are highly correlated with each other that we might have to keep our eye on them.
There are strong correlation between these two variables:
mean radius - mean perimeter
mean radius-mean area
mean radius-worst radius
mean radius-worst perimeter
mean radius- worst area

mean perimeter-mean area
mean perimeter-mean radius
mean perimeter-worst texture
mean area-worst radius
worst area- mean area
mean concavity-mean concave points
worst concave points- mean concavity



Data Analysis

We'll split the dataset into a training set and testing set, we'll use a training set to train our model and use the testing set to see how accurate the model is.

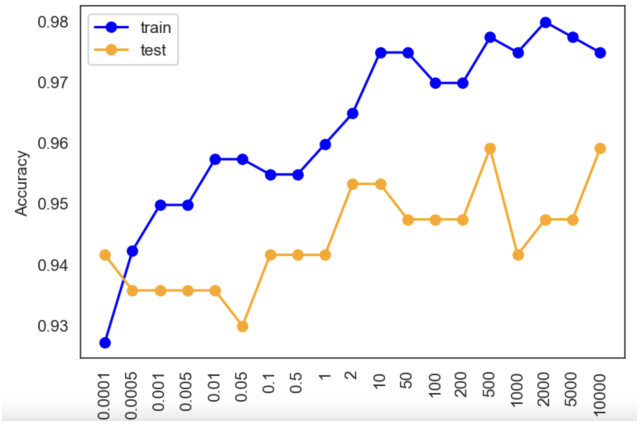
- **Logistic Regression Analysis**

Since our independence is a binary variable, we'll use the logistic regression analysis to initiate modeling. From the logistic regression modeling, we could tell the probability of an event that would take place so that we could see what are the factors that could have more possibility to be a signal of cancer tissues.

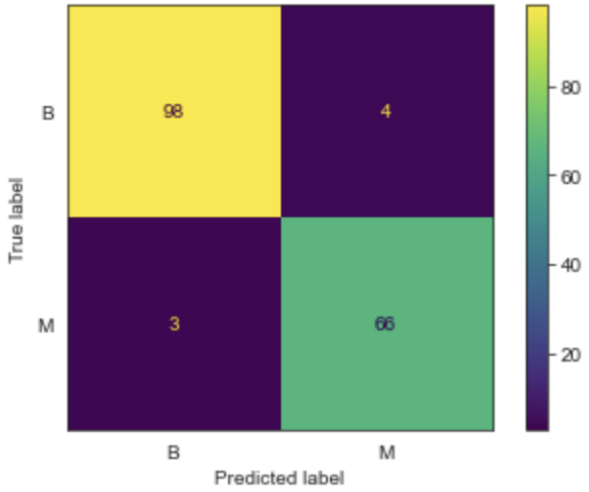
In our logistic model we'll try a sequence of parameters for C: C_list = [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2, 10, 50, 100, 200, 500, 1000, 2000, 5000, 10000]. C is the inverse of the regularization parameter λ
C increases, the penalty decreases.

We'll also apply L2 Ridge penalties into our logistic model. The penalty term is the sum of the squares of the coefficients ; it could help to reduce overfitting by shrinking large coefficients.
The result of accuracy of the model is as follow:

So here , We could choose the parameter c with the highest accuracy score in both training and testing dataset to build our model. I'll apply c equal 500 for the logistic model.



Then we could feed the model with a testing dataset to see the accuracy, specificity, recall and sensitivity to evaluate the performance of the model.



	precision	recall	f1-score	support
B	0.97	0.96	0.97	102
M	0.94	0.96	0.95	69
accuracy			0.96	171
macro avg	0.96	0.96	0.96	171
weighted avg	0.96	0.96	0.96	171

Observation

There are two possible predicted classes: "B" and "M".

Rates as computed from the confusion matrix

- **Accuracy**: Overall, how often is the classifier correct?
  - (TP+TN)/total = (66+98)/171 = 0.959
- **Misclassification Rate**: Overall, how often is it wrong?
  - (FP+FN)/total = (3+4)/171 = 0.04

    equivalent to 1 minus Accuracy also known as *"Error Rate"*

- **True Positive Rate:** When it's actually yes, how often does it predict M?
  - TP/actual yes = 66/69 = 0.95

    Also known as "Sensitivity" or *"Recall"*

- **False Positive Rate**: When it's actually M, how often does it predict B?
  - FP/actual no = 4/102 = 0.039
- **Specificity**: When it's actually B, how often does it predict B?
  - TN/actual no = 98/102 = 0.96 equivalent to M minus False Positive Rate

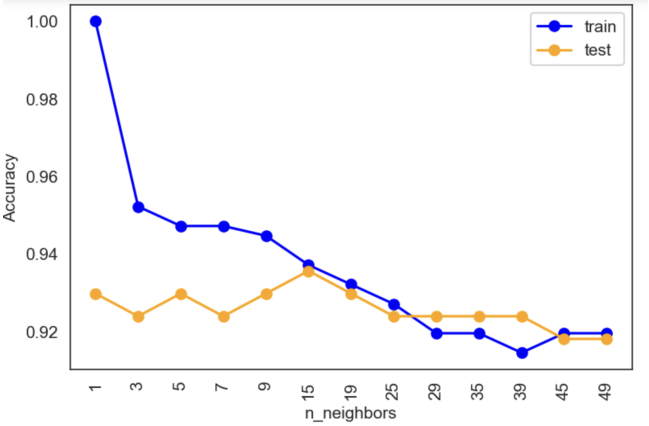
    Also known as true positive rate

- **Precision**: When it predicts M, how often is it correct?
  - TP/predicted yes = 66/69 = 0.98
- **Prevalence**: How often does the yes condition actually occur in our sample?
  - actual yes/total = 69/171 = 0.4

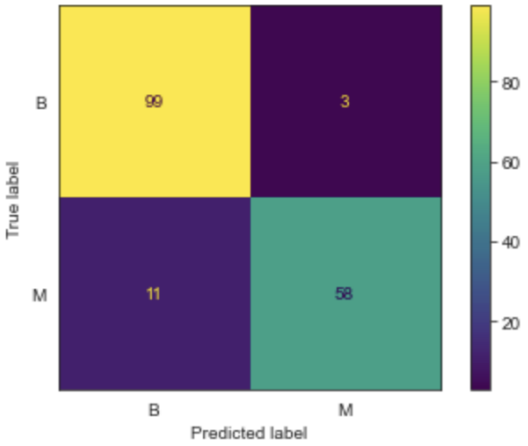
- K-NN Algorithm

Because we have a properly labeled dataset that our independent variable is clearly labeled as malignant and benign, plus that our dataset is pretty small, we could also apply the K-NN algorithm to classify the variables.
K-NN classifies the data point on how its neighbor is classified by getting the smallest Euclidean distance and based on the number of smaller distances we perform our calculation.
We'll build the K-NN model by a sequence of parameters K (n_neighbors): K_list = [1,3,5,7,9,15,19,25,29,35,39,45,49].
The we could plot the accuracy for training dataset and testing data set,
We could choose the parameter n_neighbors with the highest accuracy score in both training and testing dataset to build our model.

From the plot,we would say n_neighbors= 5 for our K-NN model.



Then we could evaluate our model:



	precision	recall	f1-score	support
B	0.90	0.97	0.93	102
M	0.95	0.84	0.89	69
accuracy			0.92	171
macro avg	0.93	0.91	0.91	171
weighted avg	0.92	0.92	0.92	171

Observation

Rates as computed from the confusion matrix

- **Accuracy**: Overall, how often is the classifier correct?
  - (TP+TN)/total = (58+99)/171 = 0.91
- **Misclassification Rate**: Overall, how often is it wrong?
  - (FP+FN)/total = (11+3)/171 = 0.07

    equivalent to 1 minus Accuracy also known as *"Error Rate"*

- **True Positive Rate:** When it's actually yes, how often does it predict M?
  - TP/actual yes = 56/69 = 0.81

Also known as "Sensitivity" or *"Recall"*

- **False Positive Rate**: When it's actually M, how often does it predict B?
    - FP/actual no = 3/102 = 0.029

  Specificity: When it's actually B, how often does it predict B?

    - TN/actual no = 99/102 = 0.97 equivalent to M minus False Positive Rate

      Also known as true positive rate

- **Precision**: When it predicts M, how often is it correct?
    - TP/predicted yes = 66/69 = 0.98
- **Prevalence**: How often does the yes condition actually occur in our sample?
    - actual yes/total = 69/171 = 0.4