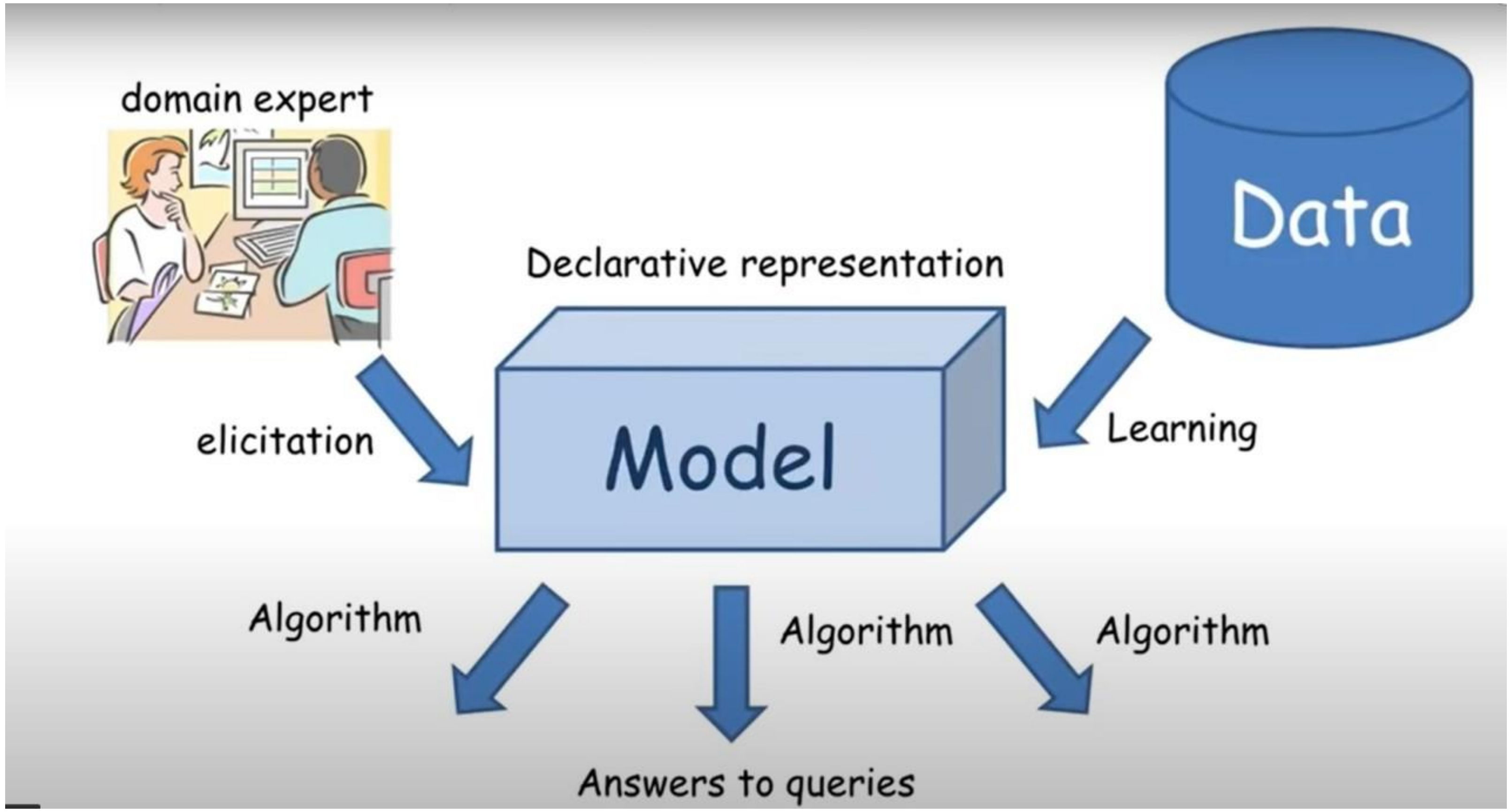
Machine Learning

- 1st Term, 2025/2026
- October 2025

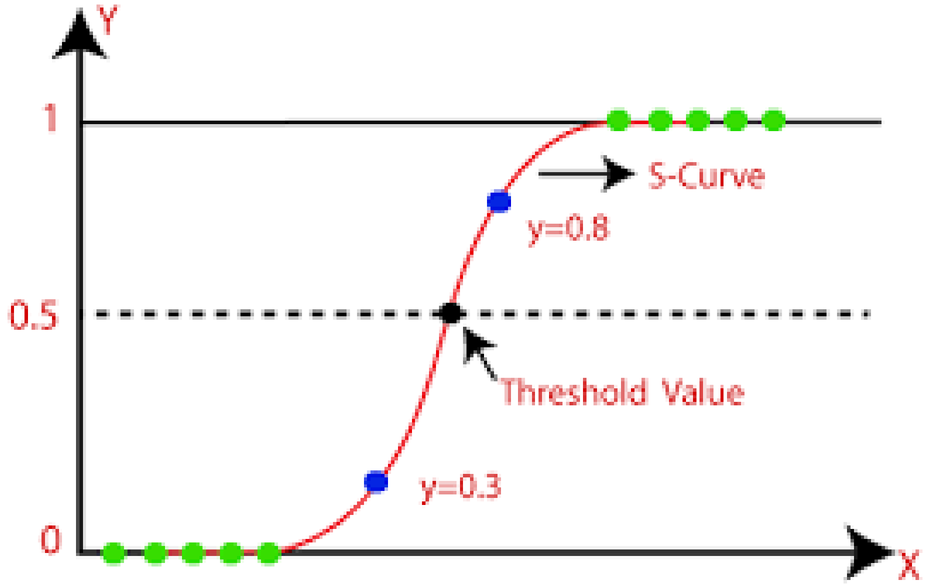
- **Prof. Mohammed A. Al Ghamdi**

Probabilistic Modeling: Review of Statistical Principles

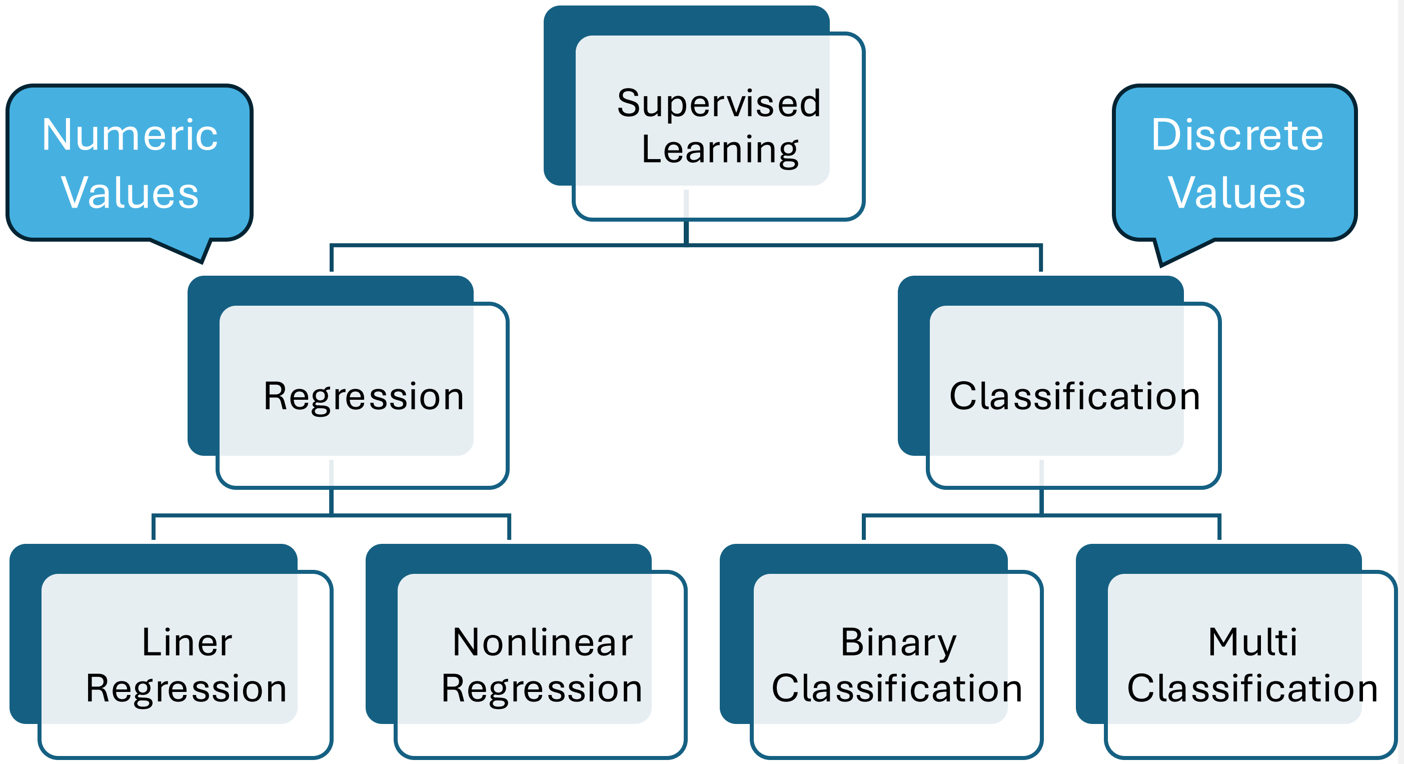


Review of Some Statistical Principles:

- **Logistic Regression.**

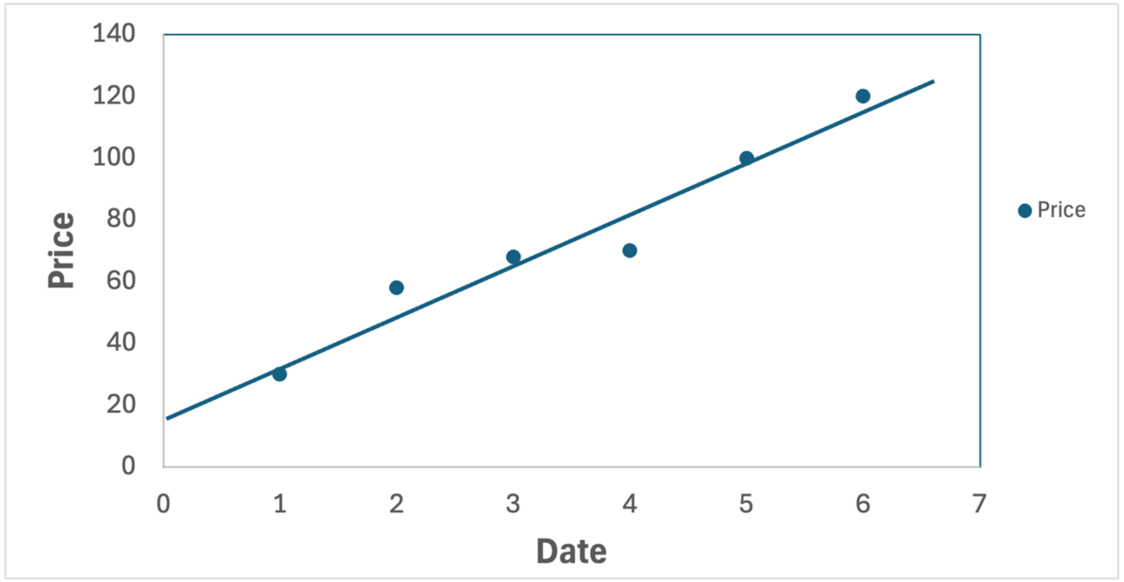


Facts:


Supervised Learning
Numeric Values
Discrete Values
Regression
Classification
Liner Regression
Nonlinear Regression
Binary Classification
Multi Classification

Logistic Regression vs Linear Regression:

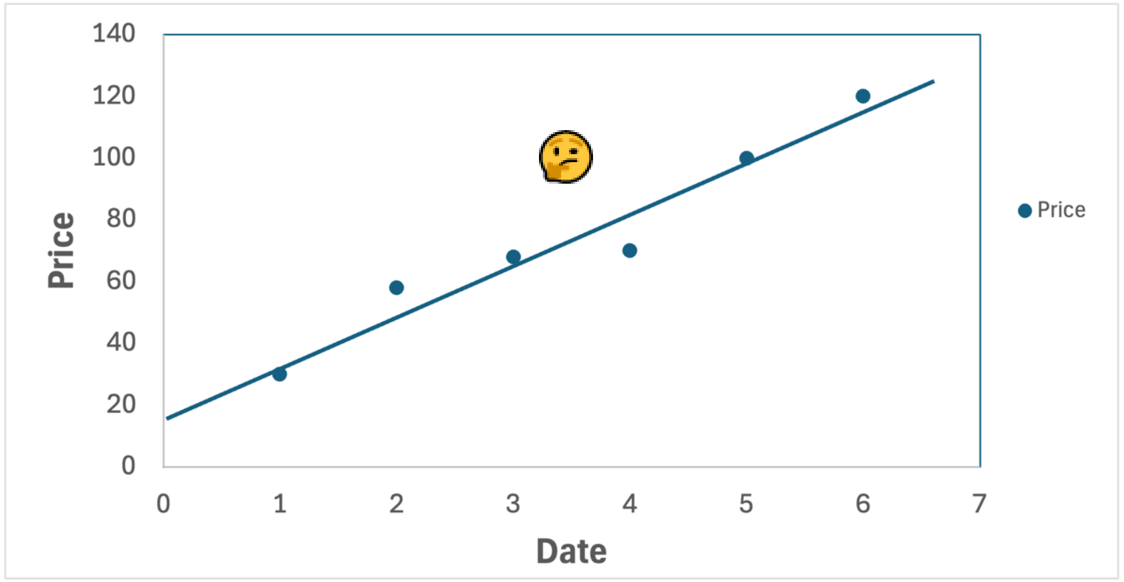
Date	Price
1	30
2	58
3	68
4	70
5	100
6	120



$$h(x) = \vartheta_0 + \vartheta_0 x_1$$

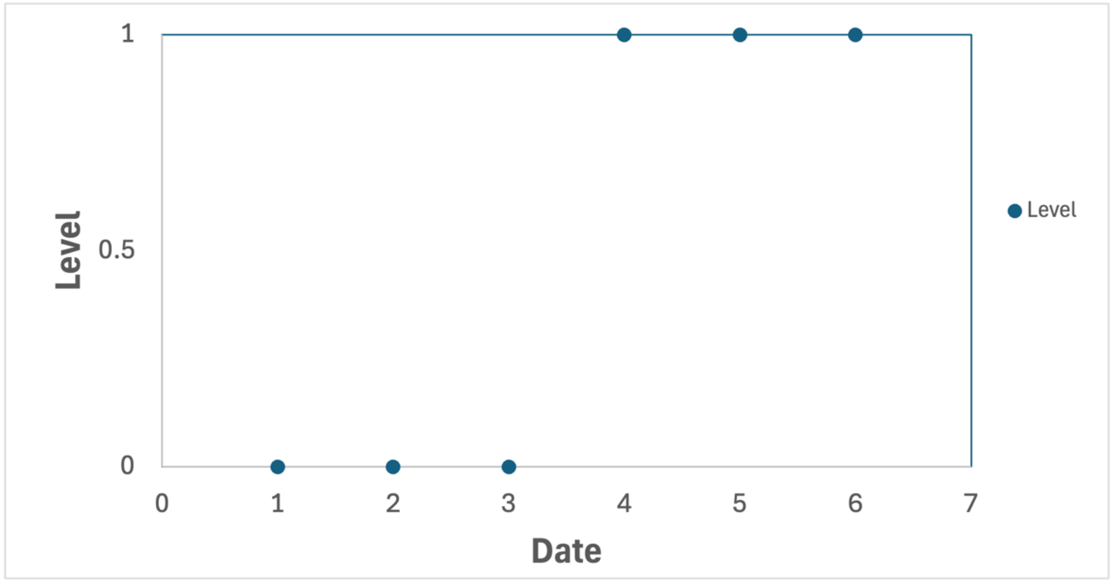
Logistic Regression vs Linear Regression:

Date	Level
1	Low
2	Low
3	Low
4	High
5	High
6	High



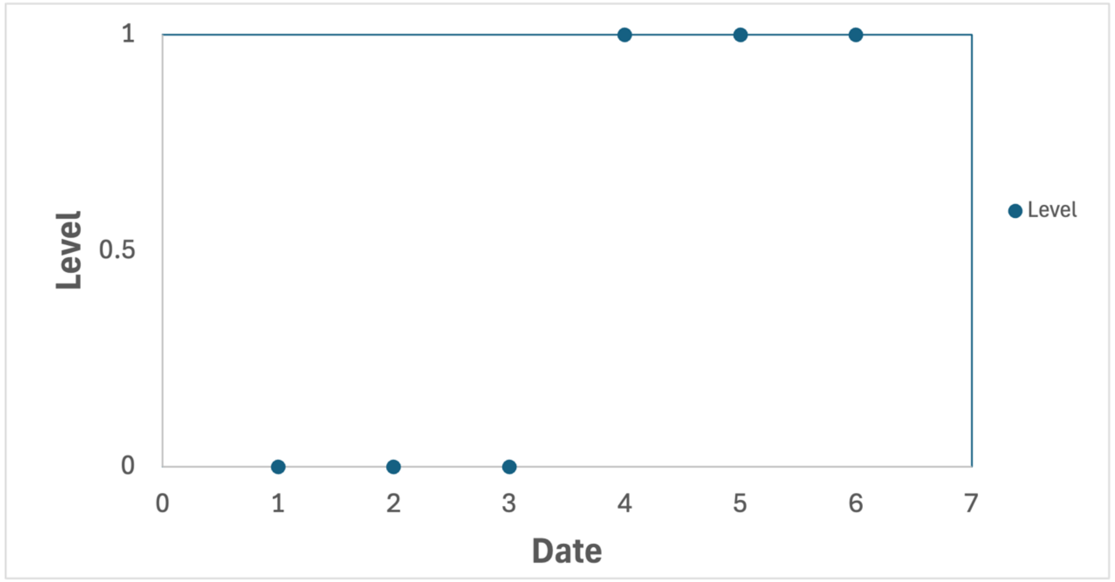
Logistic Regression vs Linear Regression:

Date	Level
1	0
2	0
3	0
4	1
5	1
6	1



Logistic Regression vs Linear Regression:

Date	Level
1	0
2	0
3	0
4	1
5	1
6	1



- We need to determine a Threshold value (0.5).

Where;

If $h(x) \geq 0.5$, Then y=1.

If $h(x) < 0.5$, Then y=0.

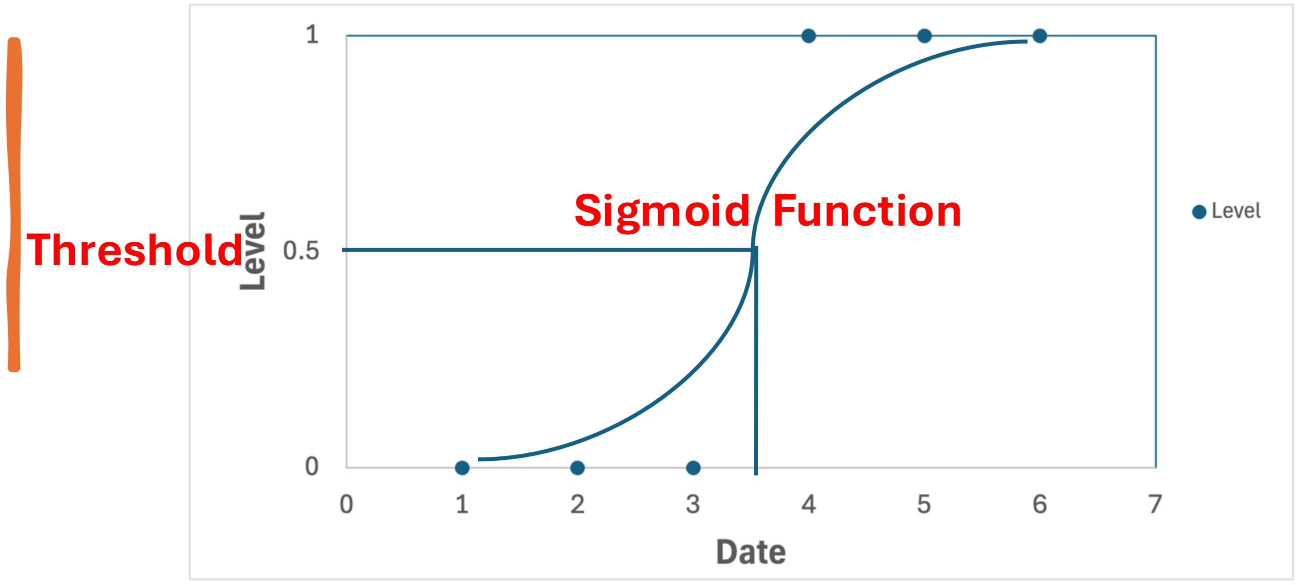
- In Logistic Regression:

$$0 \leq h(x) \leq 1$$

- For this, we use Sigmoid Function:

$$g(z) = \frac{1}{1+e-^{z}}, \text{ where } z = h(x)$$

Logistic Regression vs Linear Regression:



Date	Level
1	0
2	0
3	0
4	1
5	1
6	1

- We need to determine a Threshold value (0.5).

Where;

If $h(x) \geq 0.5$*, Then* $y=1$.

If $h(x) < 0.5$*, Then* $y=0$.

- In Logistic Regression:

$$0 \leq h(x) \leq 1$$

- For this, we use Sigmoid Function:

$$g(z) = \frac{1}{1+e^{-z}}, \text{ where } z = h(x), \ \& \ 0 < g(z) < 1.$$

Example:

$h(x) = -2x + 6$

$z = h(x)$

$$g(z) = \frac{1}{1 + e - ^{z}}$$

Y∈ $\{0,1\}$ → 0: benign

→ 1: malignant

x: tumor size

x	y
3	1
2	1
1	1
5	0
4	0
6	0

x	z	$e - ^{z}$	$1 + e - ^{z}$	g(z)	y
3	0	1	2	0.5	1
2	2	0.13536335	1.13536335	0.8808	1
1	4	0.018323237	1.018323237	0.9820	1
5	-4	54.57551085	55.57551085	0.0179	0
4	-2	7.387524	8.387524	0.1192	0
6	-6	403.1778962	404.1778962	0.00247	0

Example:

Y∈ {0,1} → 0: Fail
→ 1: Pass

x: Hours Study

Hours Study	Status
29	0
15	0
33	1
28	1
39	1

Q: Find out the probability of pass for student who studied 33 hours?

We assume the optimizer for odds of passing course is;

log(odds) = -64 + 2 * Hours

We use Sigmoid function;

$$g(z) = \frac{1}{1 + e - ^{z}}$$

Then;

*log(odds) = z = -64 + 2*Hours*

z = -64 + 66 = +2

So;

$$g(z) = \frac{1}{1+e-2} = \frac{1}{1+2.71828-^{2}} = \frac{1}{1+0.1353} = \frac{1}{1.1353} = 0.88$$

*That is; if the student studied 33 hours, then there is **88%** chance that the student will **pass** the exam.*

Example:

Y$\in \{0,1\}$ → 0: Fail

→ 1: Pass

x: Hours Study

Hours Study	Status
29	0
15	0
33	1
28	1
39	1

Q: How many hours at least student should study to pass the course with probability of 95%?

We use Sigmoid function;

$$g(z) = \frac{1}{1+e-^{z}} = 0.95$$

$$0.95 * (1 + e - ^{z}) = 1$$

$$0.95 + 0.95 * e - ^{z} = 1$$

$$e - ^{z} = \frac{0.05}{0.95} = 0.0526$$

$$e - ^{z} = 0.0526$$

$$ln(e - ^{z}) = ln(0.0526)$$

$$\Rightarrow -z = ln(0.0526) = -2.94$$

$$z = 2.94$$

Then;

$$log(odds) = z = -64 + 2 * Hours$$

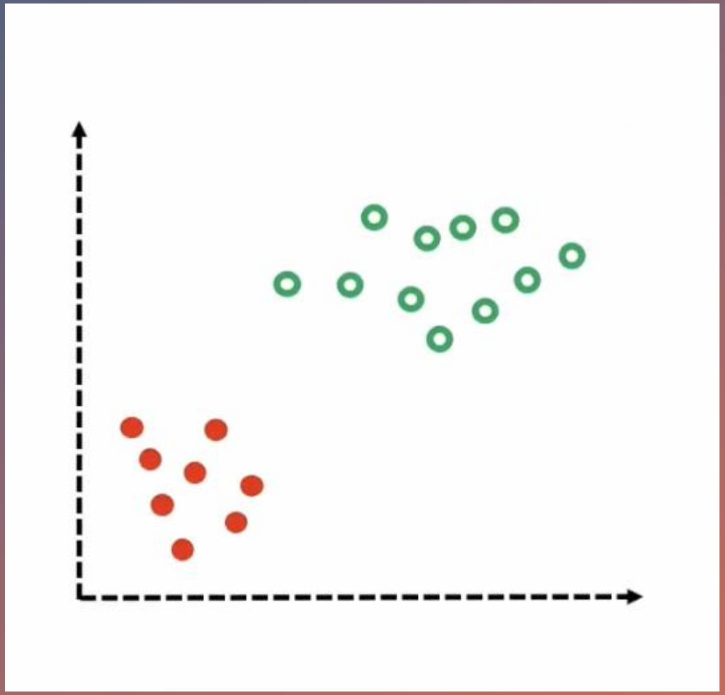
$$2.94 = -64 + 2 * Hours$$

$$\Rightarrow Hours = 33.47$$

*That is; The student should study at least **33.47** hours to pass the exam with more than **95%** probability.*

Facts:

Decision Boundary.

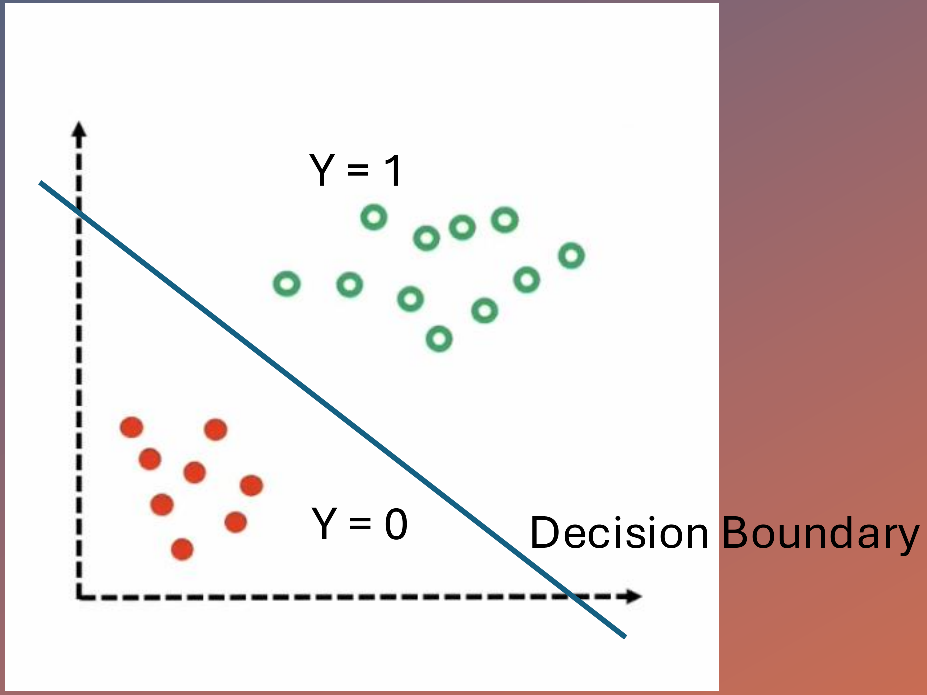


Rule:

- We need to separate the values into two region.
- The green area represents the 1st probability, and the red is for 2nd one.

Facts:

Decision Boundary.



Rule:

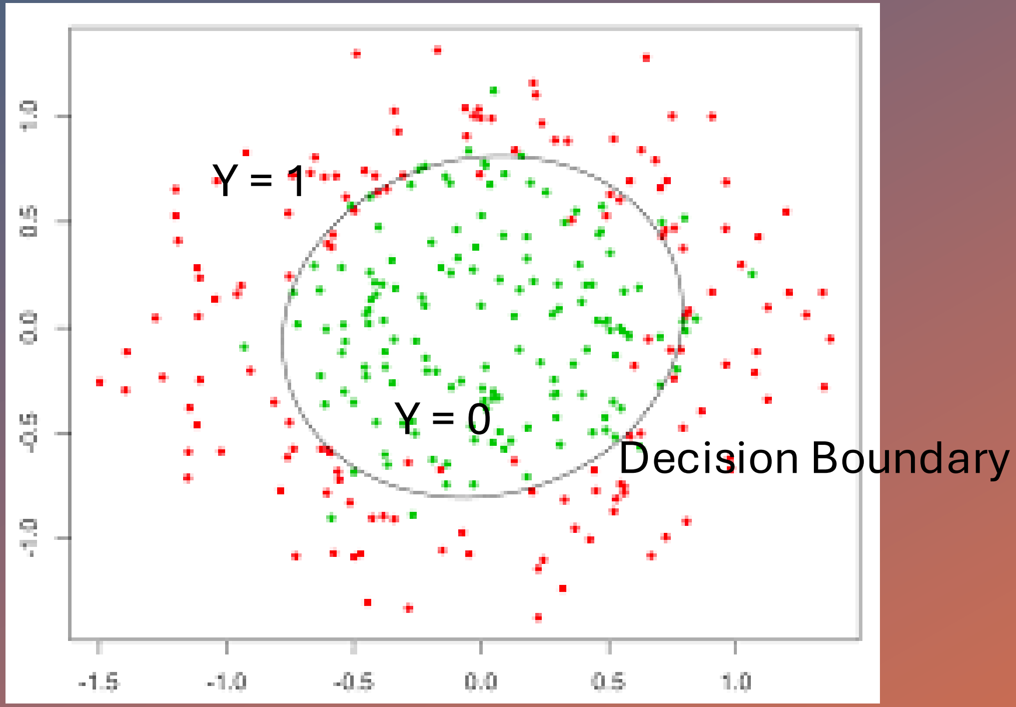
- We need to separate the values into two region.
- The green area represents the 1st probability, and the red is for 2nd one.
- The line that make perfect separation between the data is known as Decision Boundary.
- It represented by the following function;

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

i.e. $h_\theta(x) = -3 + x_1 + x_2$

Facts:

Decision Boundary.



Rule:

- It represented by the following function;

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2$$

$$\text{i.e. } h_\theta(x) = 1 + x_1^2 + x_2^2$$

- We need cost function to measure the success of our prediction;

$$j(\theta) = cost(h(x), y);$$

Where;

$$cost(h(x), y) = \begin{cases} -\log(h(x)), & if\ y = 1 \\ -\log(1 - h(x)), & if\ y = 0 \end{cases}$$

That is;

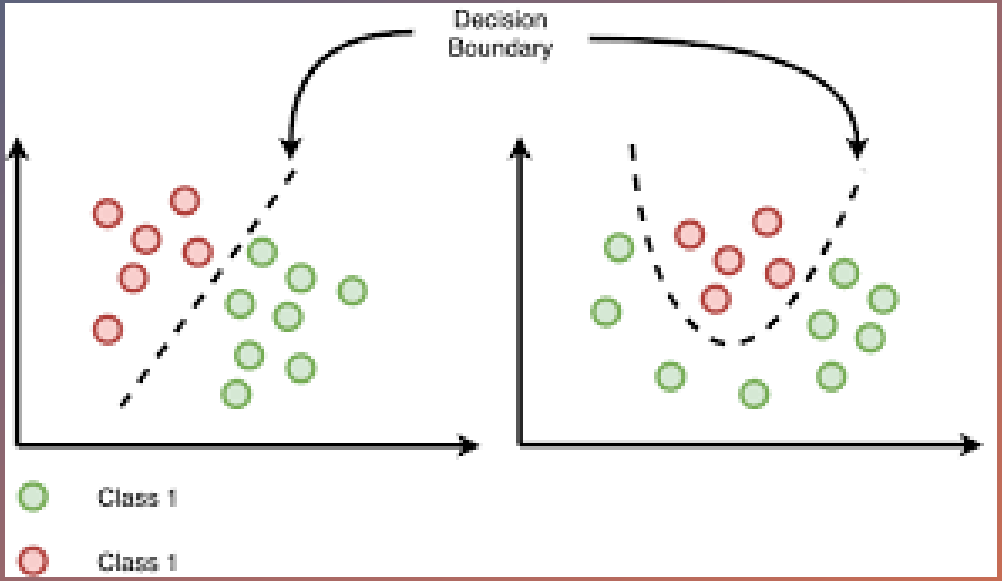
$j(\theta) = y * -log(h(x)) – (1-y) * log(1-h(x))$.

for all training set =>

$$j(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y_i log(h(x_i)) + (1 - yi)\log(1 - h(x_i))\right]$$

Facts:

Gradient Descent.



Now;

- The main goal is to minimize the cost function.

How;

- By using optimizer algorithm. That is "Gradient Descent".

$$\theta_j := \theta_j - y \frac{d}{d\theta_j} j(\theta), \text{ linear regression}$$

- So;

$$\theta_j := \theta_j - y \frac{1}{m} \left[\sum_{i=1}^{m} (g(z)^i - yi) * x_j^i\right], \quad Repeat$$

Prof. Mohammed A. Al Ghamdi, Machine Learning Course.

Multiclass Classification:

Binary Classification:

$\{x_i, y\}^m$ $Y \in \{0,1\}$

Multiclass Classification:

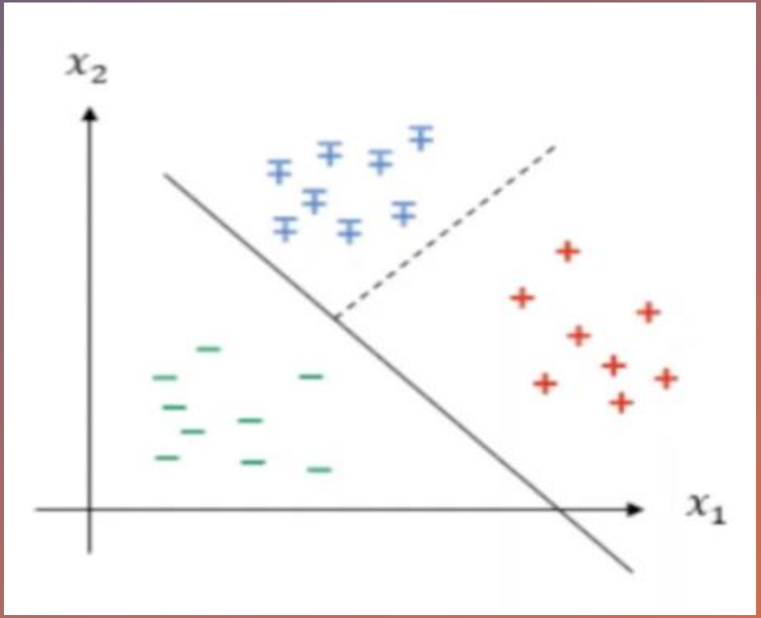
$\{x_i, y\}^m$ $Y \in \{1, 2, \ldots, N\}$

i.e. Covid-19 Dataset:

+ Positive

- Negative

∓ Positive/No Symptoms

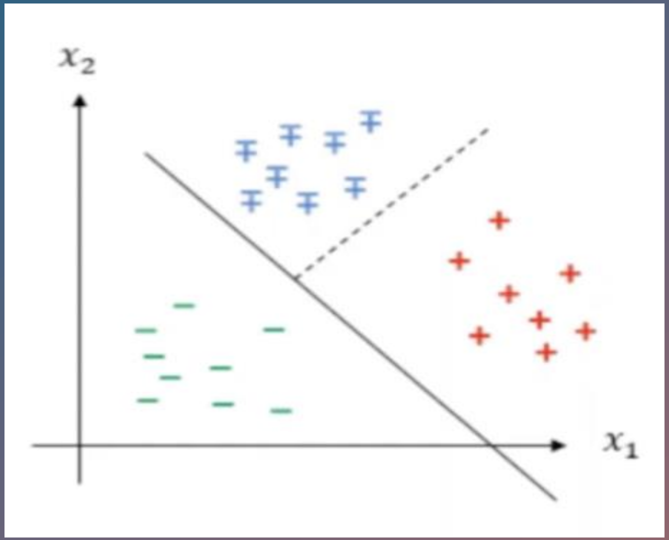


- To deal with Multiclass Classification;
- *We convert it to the Binary Classification;*

How;

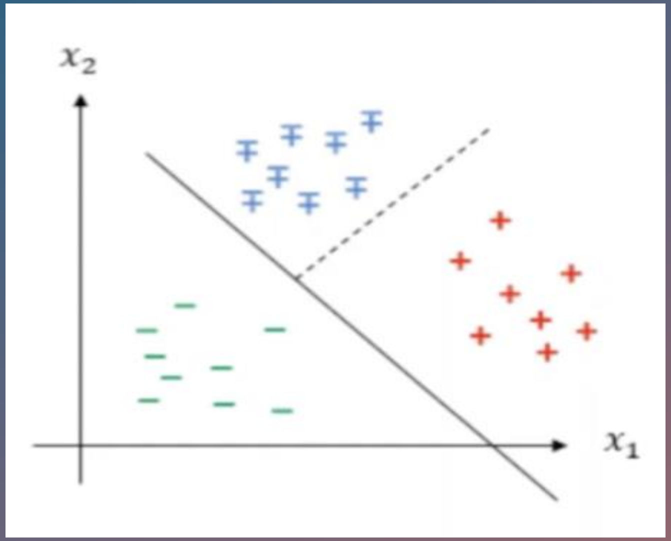
- *By using one of these two techniques;*
  1) *One vs All.*
  2) *One vs One.*

One vs All



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

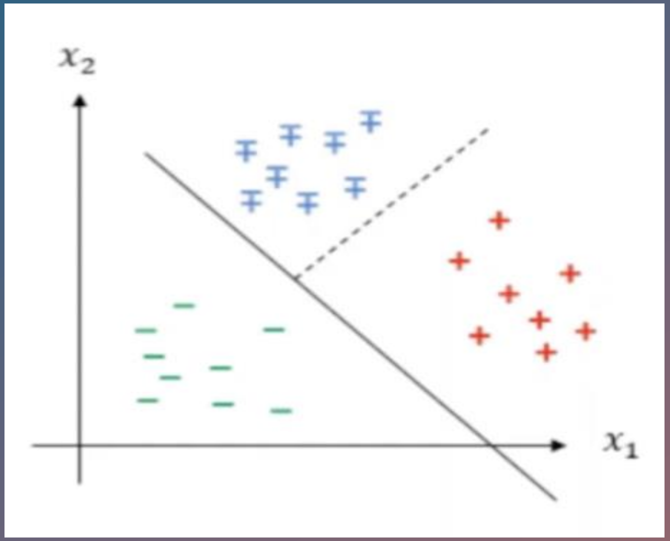
One vs All



Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

One vs All

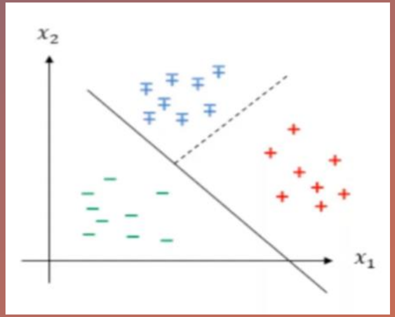


Training Set for +

Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

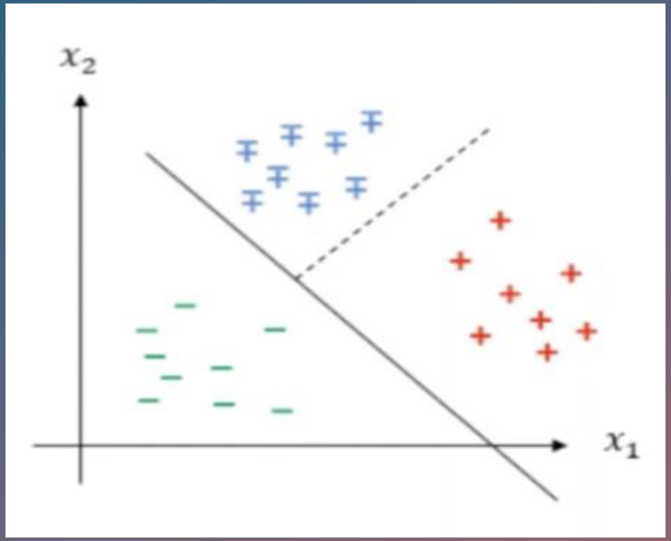
Training Binary Classifier 1		
$\vartheta^t . x_i$	g(z)	GD

Classification model 1



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

One vs All



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

Training Set for +

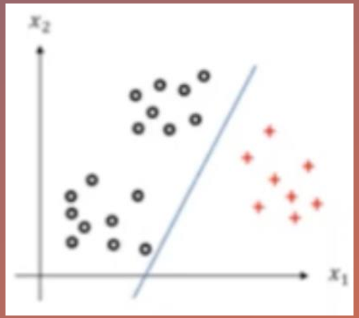
Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

Training Set for -

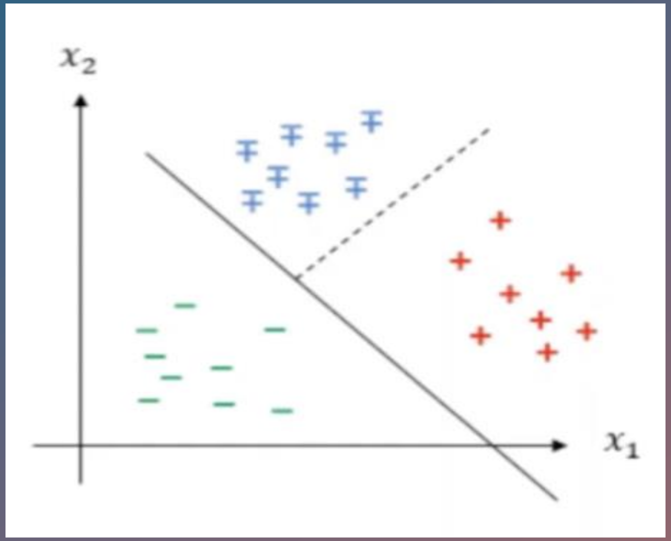
Features		Class
x1	x2	1
x1	x2	0
x1	x2	0
x1	x2	0

Training Binary Classifier 1		
$\vartheta^t . x_i$	g(z)	GD

Classification model 1



One vs All



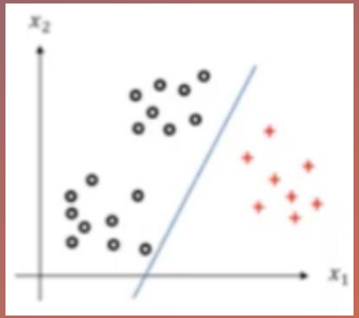
Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

Training Set for +

Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

Training Binary Classifier 1		
$\vartheta^t.x_i$	g(z)	GD

Classification model 1

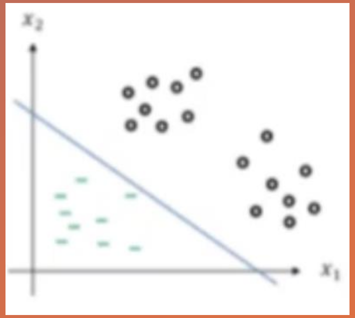


Training Set for -

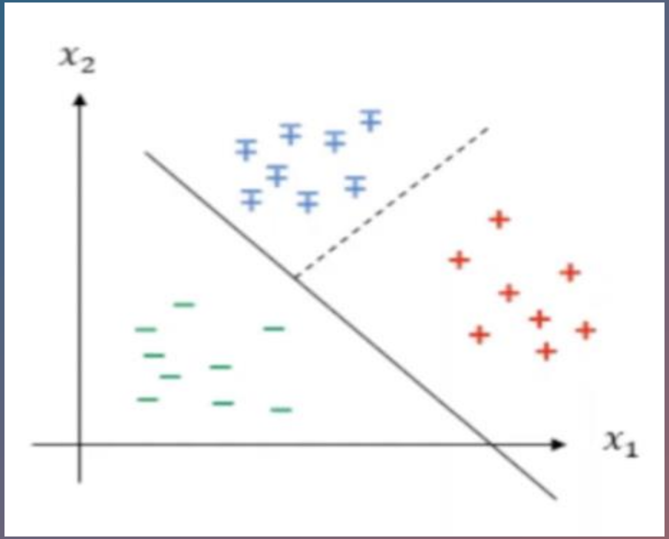
Features		Class
x1	x2	1
x1	x2	0
x1	x2	0
x1	x2	0

Training Binary Classifier 2		
$\vartheta^t.x_i$	g(z)	GD

Classification model 2



One vs All



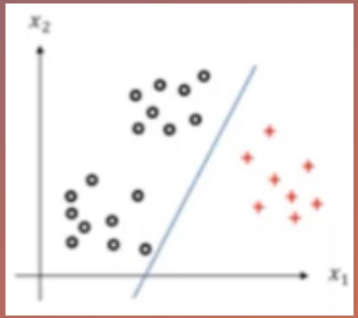
Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

Training Set for +

Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

Training Binary Classifier 1		
$\vartheta^t.x_i$	g(z)	GD

Classification model 1

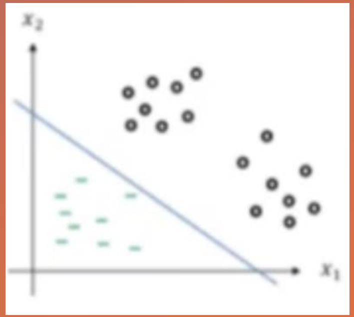


Training Set for -

Features		Class
x1	x2	1
x1	x2	0
x1	x2	0
x1	x2	0

Training Binary Classifier 2		
$\vartheta^t.x_i$	g(z)	GD

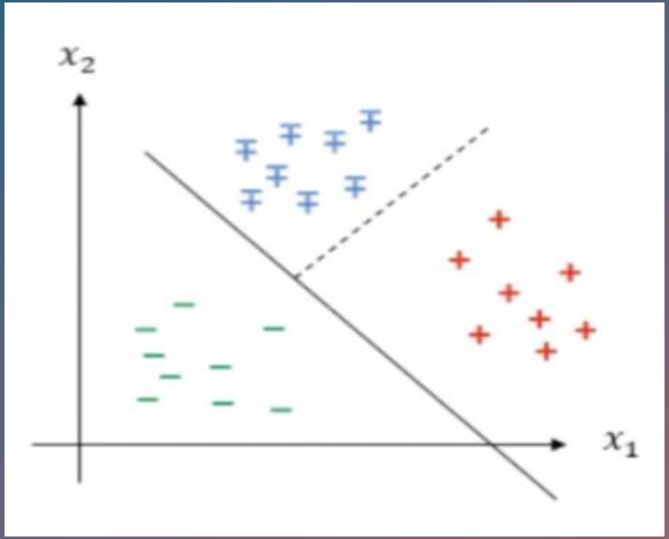
Classification model 2



Training Set for ∓

Features		Class
x1	x2	0
x1	x2	0
x1	x2	0
x1	x2	1

One vs All



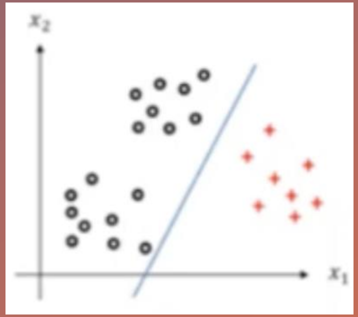
Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

Training Set for +

Features		Class
x1	x2	0
x1	x2	1
x1	x2	1
x1	x2	0

Training Binary Classifier 1		
$\vartheta^t.x_i$	g(z)	GD

Classification model 1

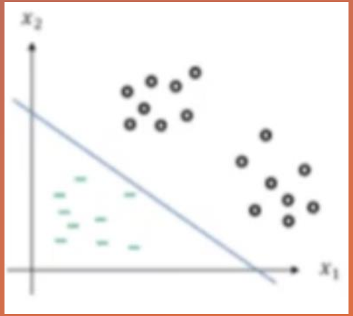


Training Set for -

Features		Class
x1	x2	1
x1	x2	0
x1	x2	0
x1	x2	0

Training Binary Classifier 2		
$\vartheta^t.x_i$	g(z)	GD

Classification model 2

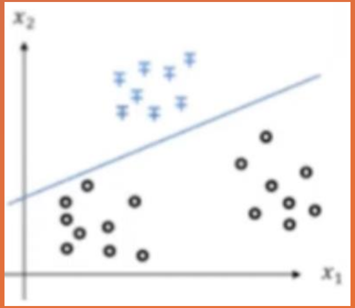


Training Set for ∓

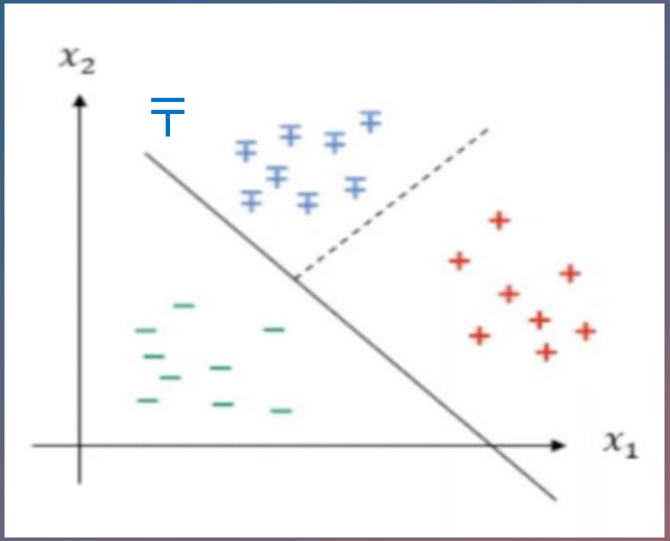
Features		Class
x1	x2	0
x1	x2	0
x1	x2	0
x1	x2	1

Training Binary Classifier 3		
$\vartheta^t.x_i$	g(z)	GD

Classification model 3



One vs All



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

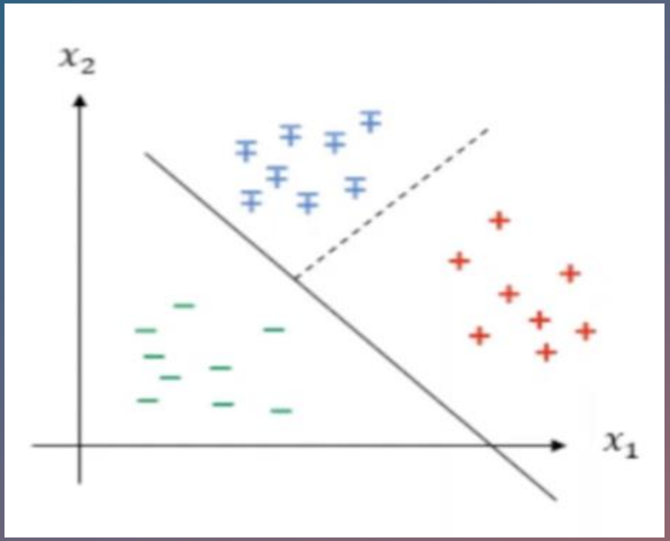
- Now, if we have new data to predict their class;

  x1, and x2
- Firstly, the ***classification model 1, 2, and 3*** will be applied over the given data.
- In each model the given data will be classified separately.
- The output will be presented;

  $$p(y = 1 \mid x_i : \theta)$$
- Then, the predicted y is defined as;

  Predicted y = Max(c1, c2, c3)

One vs One



Features		Class
x1	x2	-
x1	x2	+
x1	x2	+
x1	x2	∓

- In this technique, the classification model for each class is compared with another ONE model (that is; One vs. One) = Binary Classification.
- How many Binary Classification do we need?

$$\frac{N * (N-1)}{2}$$

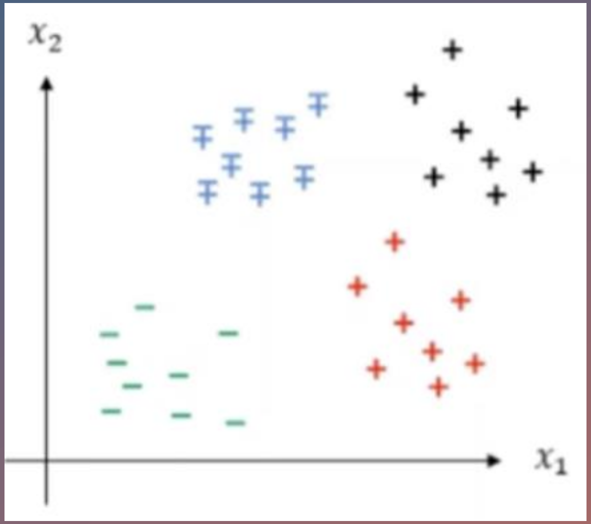
- If we have 4 classes (+, -, ∓). Then, we will do 6 binary classification.
- The output for each classifier model will be presented as;

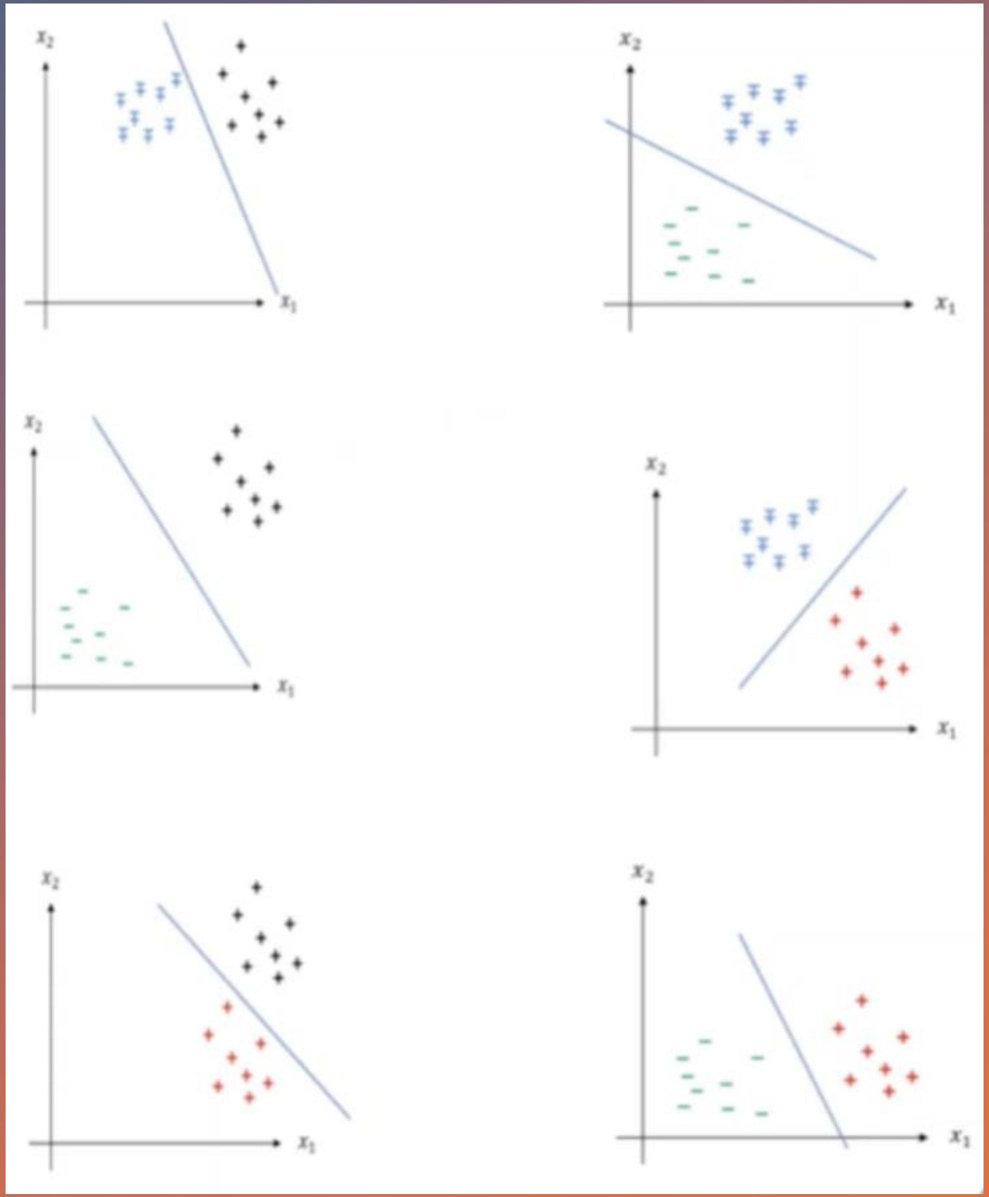
$$p(y = 1 \mid x_i : \theta)$$

- Then, the predicted y is defined as;

$$Predicted\ y = Max(c1, c2, c3, \ldots)$$

One vs One





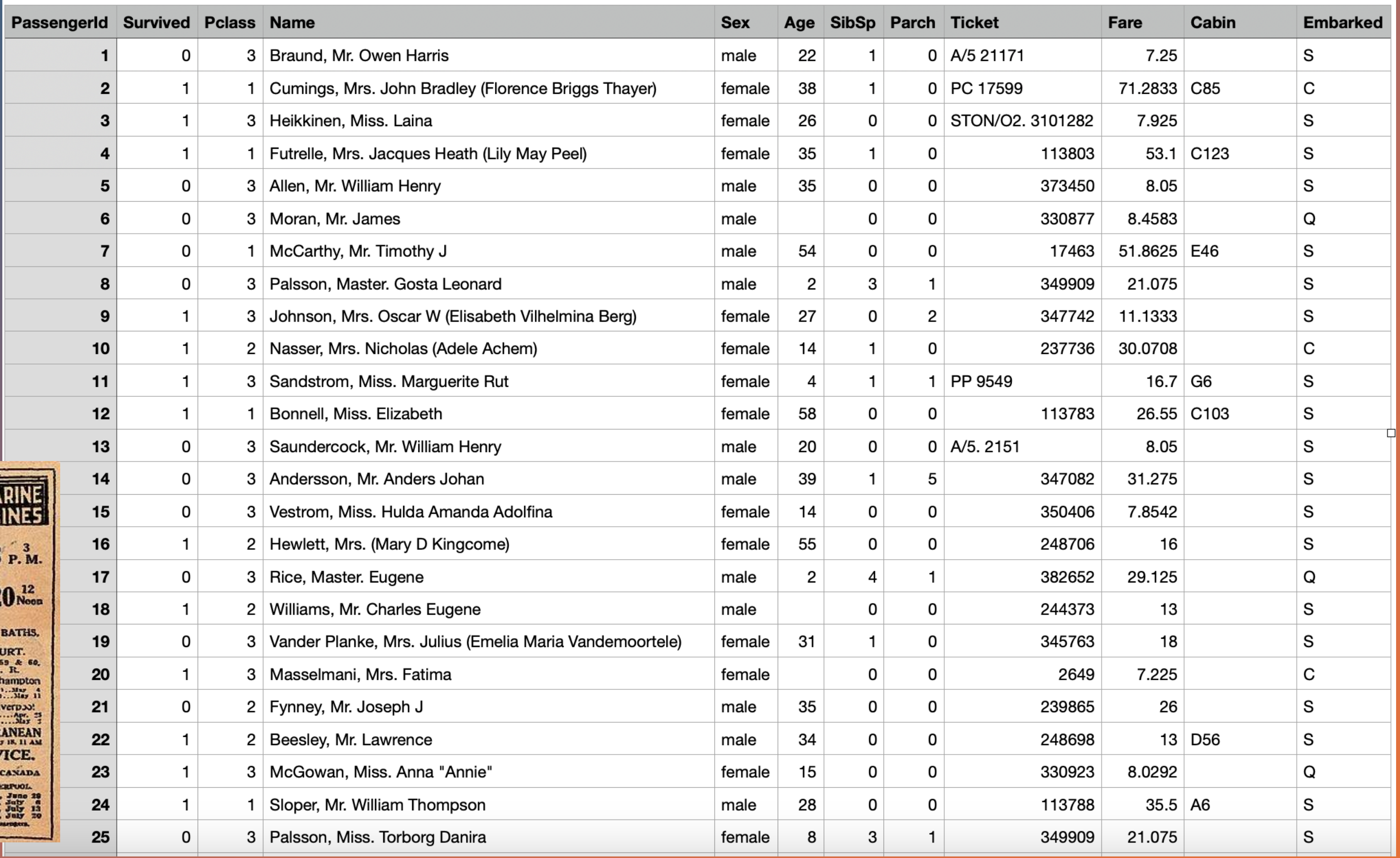
Predicted y = Max(c1, c2, c3, c4, c5, c6)

Advantages and disadvantages of Logistic regression

- Easy to implement and interpret.
- The predicted parameters give inference about the importance of each feature.
- Performs well on low-dimensional data.
- Very efficient when the dataset has features that are linearly separable.
- Outputs well-calibrated probabilities along with classification results.

- Overfits on high dimensional data.
- Non linear problems can't be solved since it has a linear decision surface.
- Assumes linearity between dependent and independent variables.
- Fails to capture complex relationships.
- Only important and relevant features should be used otherwise model's predictive value will degrade.

Example I:



PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr. James	male		0	0	330877	8.4583		Q
7	0	1	McCarthy, Mr. Timothy J	male	54	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2	3	1	349909	21.075		S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	347742	11.1333		S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0	237736	30.0708		C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4	1	1	PP 9549	16.7	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58	0	0	113783	26.55	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20	0	0	A/5. 2151	8.05		S
14	0	3	Andersson, Mr. Anders Johan	male	39	1	5	347082	31.275		S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14	0	0	350406	7.8542		S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55	0	0	248706	16		S
17	0	3	Rice, Master. Eugene	male	2	4	1	382652	29.125		Q
18	1	2	Williams, Mr. Charles Eugene	male		0	0	244373	13		S
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31	1	0	345763	18		S
20	1	3	Masselmani, Mrs. Fatima	female		0	0	2649	7.225		C
21	0	2	Fynney, Mr. Joseph J	male	35	0	0	239865	26		S
22	1	2	Beesley, Mr. Lawrence	male	34	0	0	248698	13	D56	S
23	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0	330923	8.0292		Q
24	1	1	Sloper, Mr. William Thompson	male	28	0	0	113788	35.5	A6	S
25	0	3	Palsson, Miss. Torborg Danira	female	8	3	1	349909	21.075		S

INTERNATIONAL MERCANTILE MARINE LINES

ALL OUR STEAMERS HAVE WIRELESS TELEGRAPH AND SUBMARINE SIGNALS

The Largest Steamers in the World

New 45,000 Tons Each **OLYMPIC** Sail from NEW YORK **April 13** 3 P. M.
MAY 4, 25, JUNE 15, JULY 6.

TITANIC **APRIL 20** 12 Noon
May 11, June 1, June 22, July 13

FRENCH A LA CARTE RESTAURANT, TURKISH & ELECTRIC BATHS, SWIMMING POOL, FOUR ELEVATORS, GYMNASIUM, VERANDAH CAFE, PALM COURT, SQUASH RACQUET COURT.

NEARBY SAILINGS

Baltic........	April 11.	12:00 Noon
Vaderland....	April 13.	10:00 A. M.
Minnewaska..	April 14.	
Cedric........	April 18.	12:00 Noon

AMERICAN Pier 62, N. R. 9:30 A. M.
Plymouth—Cherbourg—Southampton
St. Paul........May 4—St. Louis........May 11

ATLANTIC TRANSPORT
New York—London Direct, Pier 58, N.R.
Minnewaska..........Apr. 14 | Minnehaha..May 11
Minnetonka.May 4. 10:30 AM | Minnew'ka..May 18

RED STAR Pier 61, N. R. 10 A. M.
London, Paris via Dover—Antwerp
Vaderland........Apr. 13 | Kroonland........Apr. 27
Lapland..........Apr 20 | Finland..........May 4

WHITE STAR Piers 59 & 60, N. R.
Plymouth—Cherbourg—Southampton
Olympic(new) Ap.13,3PM | Olympic (new)..May 4
Titanic (new).Ap.20,n'n | Titanic (new)...May 11

New York—Queenstown—Liverpool
Baltic....Apr. 11, noon | Celtic ....Apr. 25
Cedric....Apr. 18, noon | Adriatic ....May 2

FROM BOSTON To The MEDITERRANEAN
CANOPIC, Ap.27, 3 PM. CRETIC, May 18, 11 AM

CANADIAN SERVICE.

LARGEST STEAMERS FROM CANADA

MONTREAL VIA QUEBEC TO LIVERPOOL.

*CANADA	May 4,	June 1,	June 29
MEGANTIC	May 11,	June 8,	July 6
*TEUTONIC	May 18,	June 15,	July 13
LAURENTIC	May 25,	June 22,	July 20

*Only One Class of Cabin (II) Passengers.

PASSENGER DEPARTMENT, 9 BROADWAY, NEW YORK.

Prof. Mohammed A. Al Ghamdi, Machine Learning Course.

Example I:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report,accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv('Titanic.csv')
df = df[['Survived', 'Age', 'Sex', 'Pclass']]
df = pd.get_dummies(df, columns=['Sex', 'Pclass'])
df.dropna(inplace=True)

df['Survived'].value_counts().plot(kind='bar')

#x = df.drop('Survived', axis=1)
x = df.drop('Survived', axis=1)
y = df['Survived']

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, stratify=y, random_state=0)

model = LogisticRegression(random_state=0)
model.fit(x_train.values, y_train.values)
```

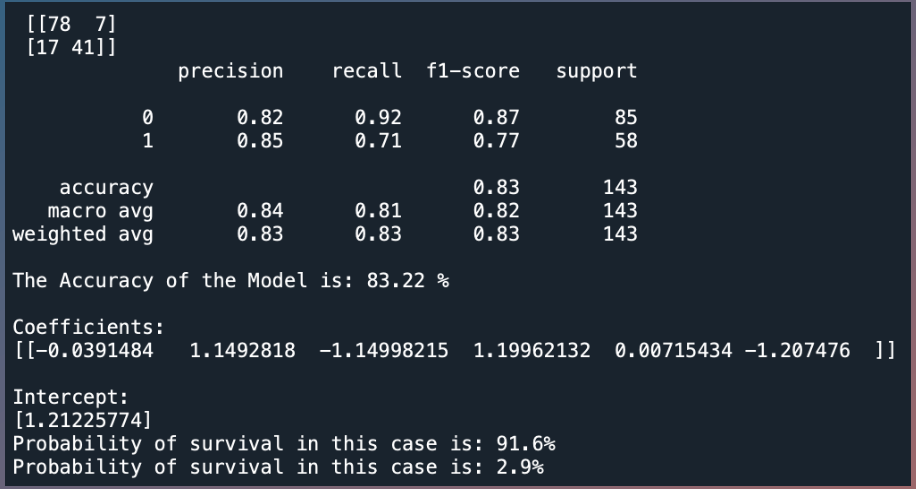
Example I:

```
predict_test = model.predict(x_test.values)
confusion_matrix(y_test, predict_test)

print(confusion_matrix(y_test,predict_test))
print(classification_report(y_test,predict_test))
print("The Accuracy of the Model is:", round(accuracy_score(y_test,predict_test)*100,2),"%")
#print(model.score(x_test.values, y_test))

fig, ax = plt.subplots()

sns.heatmap(pd.DataFrame(confusion_matrix(y_test,predict_test)), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

# Calculate and display the coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

print("\nCoefficients:")
print(coefficients)
print("\nIntercept:")
print(intercept)
```

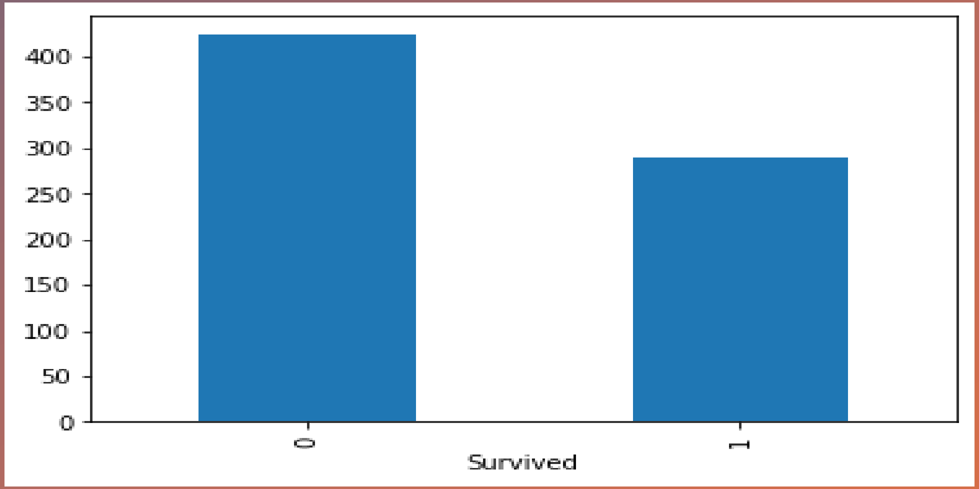
Example I:

```
# Make a prediction about a 45-year-old female traveling in 1st class will survive?
female = [[30, 1, 0, 1, 0, 0]]
model.predict(female)[0]
probability = model.predict_proba(female)[0][1]
print(f'Probability of survival in this case is: {probability:.1%}’)

# Make a prediction about a 60-year-old male traveling in 3rd class will survive?
male = [[60, 0, 1, 0, 0, 1]]
probability = model.predict_proba(male)[0][1]
print(f'Probability of survival in this case is: {probability:.1%}')
```

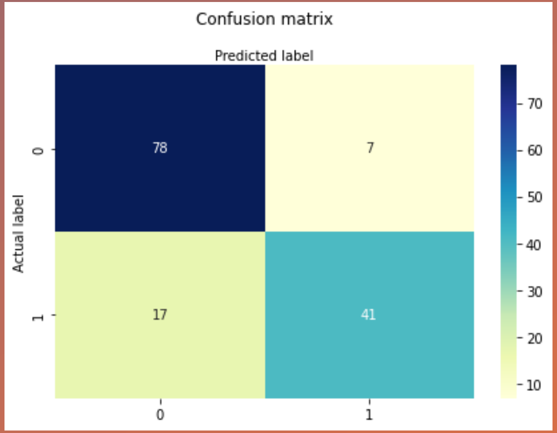
Example I:



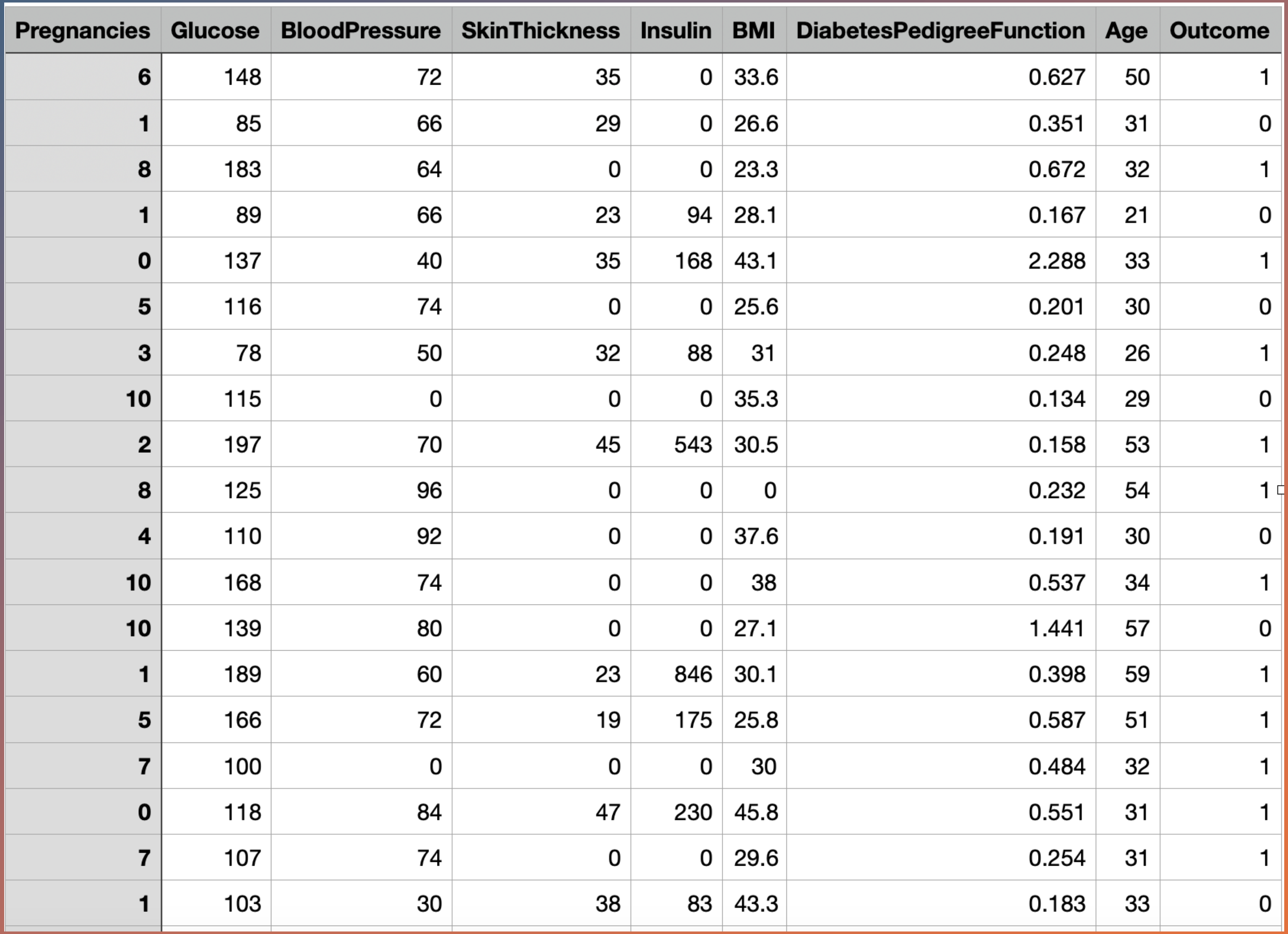
```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476  ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```



Our model predicts 84% of passengers' survival correctly (precision).



Example II:



Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0

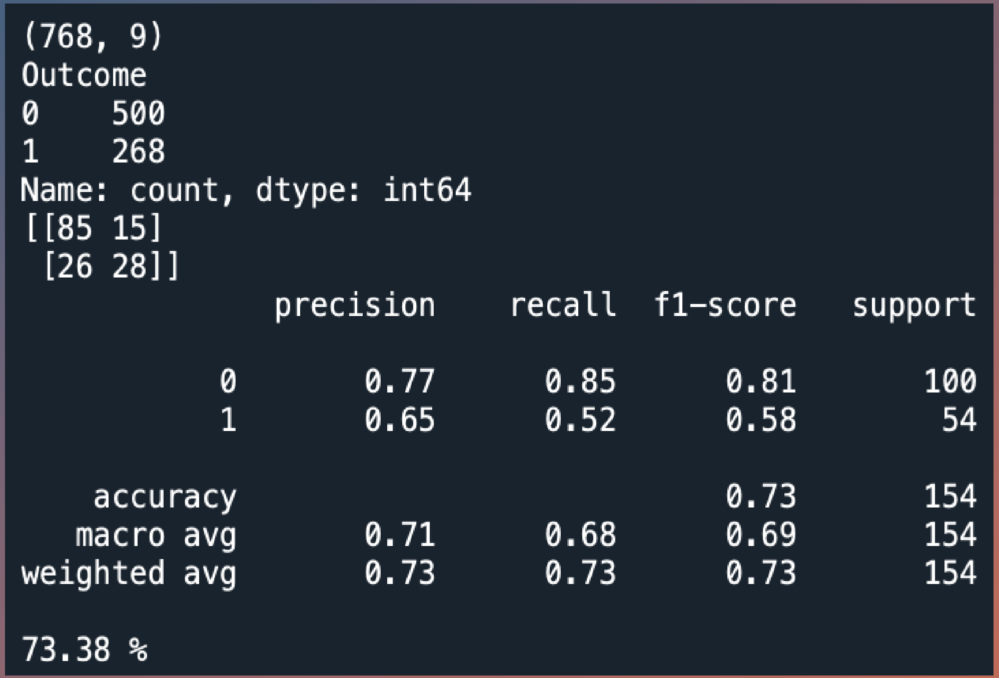
Example II:

```
import pandas as pd

from sklearn.model_selection import train_test_split #To split data into random train and test subsets.
from sklearn.linear_model import LogisticRegression #Logistic Regression classifier.
from sklearn.preprocessing import StandardScaler #Removing the mean and scaling to unit variance.
from sklearn.metrics import classification_report,accuracy_score #Build Report showing the main classification metrics.
from sklearn.metrics import confusion_matrix #Build Confusion Matrix

diabetes = pd.read_csv('diabetes.csv')

print(diabetes.shape)
print(diabetes['Outcome'].value_counts())

diabetes['Outcome'].value_counts().plot(kind='bar')

#Build Model
x = diabetes.drop(['Outcome'],axis=1)
y = diabetes['Outcome']

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=11)
```

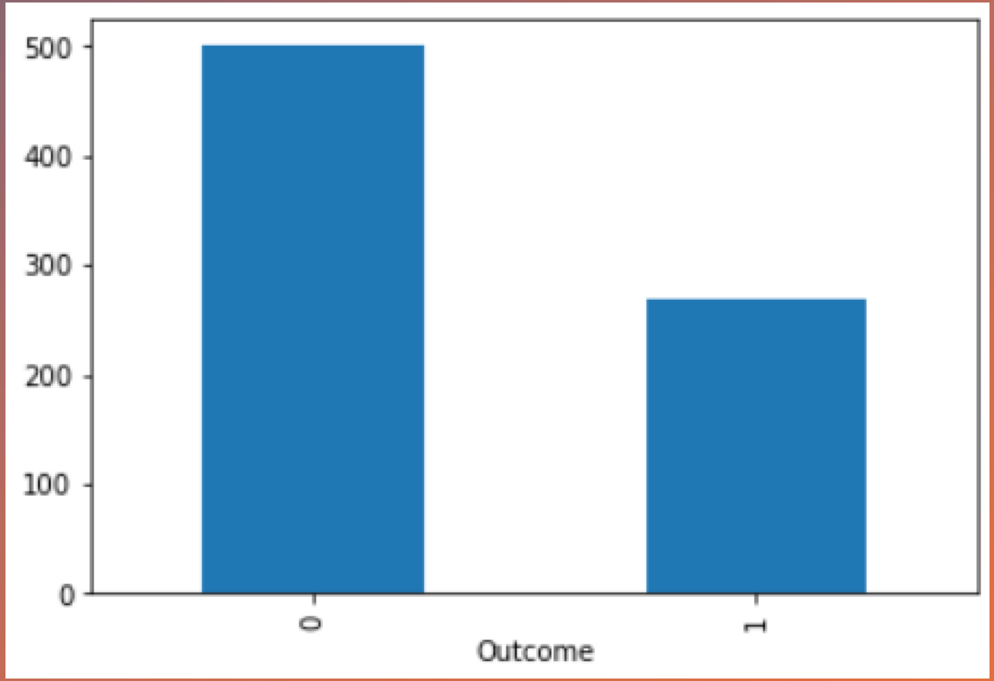
Example II:

```
s = StandardScaler()

#Learn the parameters and apply the transformation to new data
x_train = s.fit_transform(x_train)
x_test = s.fit_transform(x_test)

model = LogisticRegression()
model.fit(x_train,y_train)

predict_test = model.predict(x_test)

print(confusion_matrix(y_test,predict_test))
print(classification_report(y_test,predict_test))
print(round(accuracy_score(y_test,predict_test)*100,2),"%")
```

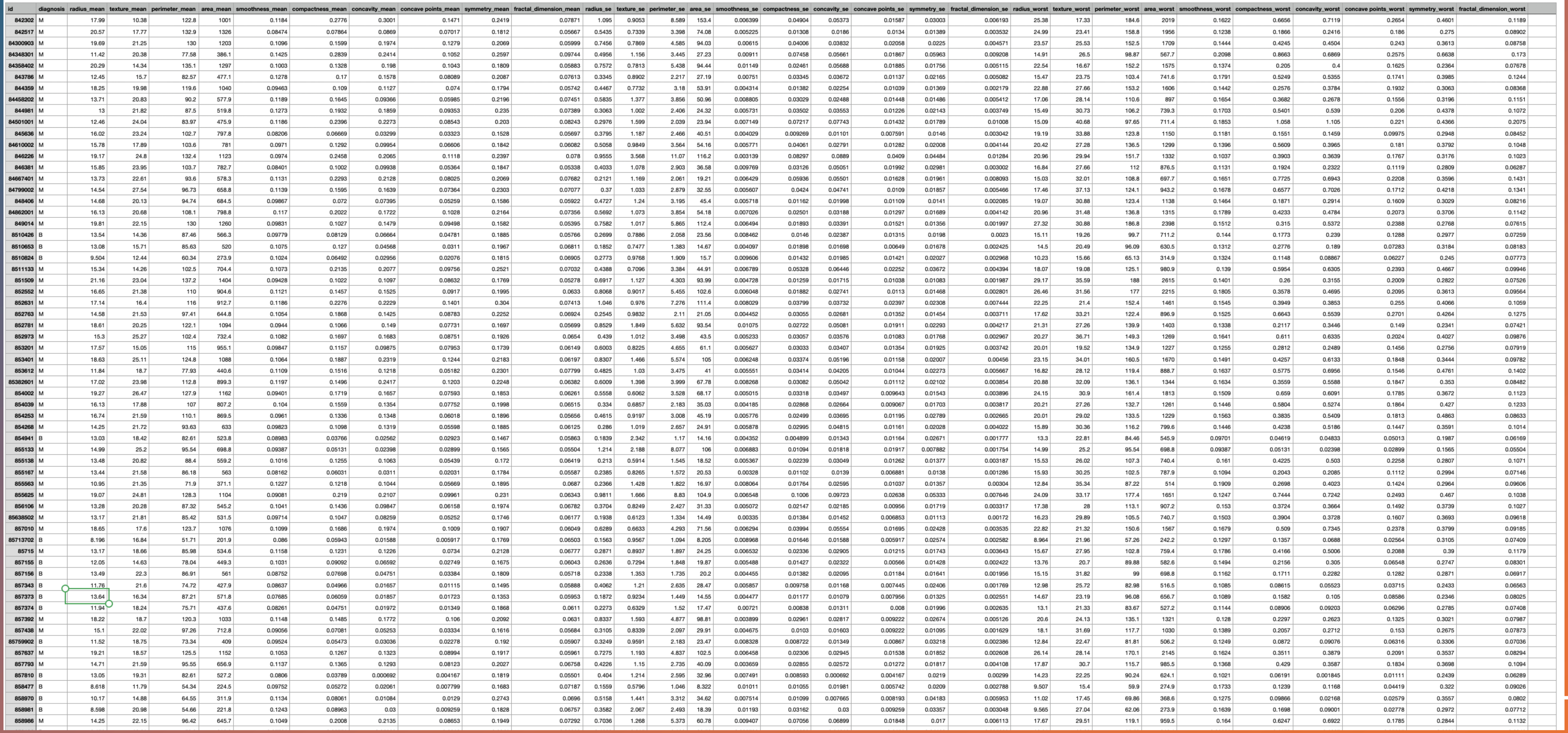
Example II:



```
(768, 9)
Outcome
0    500
1    268
Name: count, dtype: int64
[[85 15]
 [26 28]]
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       100
           1       0.65      0.52      0.58        54

    accuracy                           0.73       154
   macro avg       0.71      0.68      0.69       154
weighted avg       0.73      0.73      0.73       154

73.38 %
```



Example III:



id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871
842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999
84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744
84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883
843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613
844359	M	18.25	19.98	119.6	1040	0.09463	0.109	0.1127	0.074	0.1794	0.05742
84458202	M	13.71	20.83	90.2	577.9	0.1189	0.1645	0.09366	0.05985	0.2196	0.07451
844981	M	13	21.82	87.5	519.8	0.1273	0.1932	0.1859	0.09353	0.235	0.07389
84501001	M	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203	0.08243
845636	M	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528	0.05697
84610002	M	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842	0.06082
846226	M	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397	0.078
846381	M	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847	0.05338
84667401	M	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069	0.07682
84799002	M	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303	0.07077
848406	M	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586	0.05922
84862001	M	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164	0.07356
849014	M	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582	0.05395
8510426	B	13.54	14.36	87.46	566.3	0.09779	0.08129	0.06664	0.04781	0.1885	0.05766
8510653	B	13.08	15.71	85.63	520	0.1075	0.127	0.04568	0.0311	0.1967	0.06811
8510824	B	9.504	12.44	60.34	273.9	0.1024	0.06492	0.02956	0.02076	0.1815	0.06905
8511133	M	15.34	14.26	102.5	704.4	0.1073	0.2135	0.2077	0.09756	0.2521	0.07032
851509	M	21.16	23.04	137.2	1404	0.09428	0.1022	0.1097	0.08632	0.1769	0.05278
852552	M	16.65	21.38	110	904.6	0.1121	0.1457	0.1525	0.0917	0.1995	0.0633
852631	M	17.14	16.4	116	912.7	0.1186	0.2276	0.2229	0.1401	0.304	0.07413
852763	M	14.58	21.53	97.41	644.8	0.1054	0.1868	0.1425	0.08783	0.2252	0.06924
852781	M	18.61	20.25	122.1	1094	0.0944	0.1066	0.149	0.07731	0.1697	0.05699
852973	M	15.3	25.27	102.4	732.4	0.1082	0.1697	0.1683	0.08751	0.1926	0.0654
853201	M	17.57	15.05	115	955.1	0.09847	0.1157	0.09875	0.07953	0.1739	0.06149
853401	M	18.63	25.11	124.8	1088	0.1064	0.1887	0.2319	0.1244	0.2183	0.06197
853612	M	11.84	18.7	77.93	440.6	0.1109	0.1516	0.1218	0.05182	0.2301	0.07799
85382601	M	17.02	23.98	112.8	899.3	0.1197	0.1496	0.2417	0.1203	0.2248	0.06382
854002	M	19.27	26.47	127.9	1162	0.09401	0.1719	0.1657	0.07593	0.1853	0.06261
854039	M	16.13	17.88	107	807.2	0.104	0.1559	0.1354	0.07752	0.1998	0.06515
854253	M	16.74	21.59	110.1	869.5	0.0961	0.1336	0.1348	0.06018	0.1896	0.05656
854268	M	14.25	21.72	93.63	633	0.09823	0.1098	0.1319	0.05598	0.1885	0.06125
854941	B	13.03	18.42	82.61	523.8	0.08983	0.03766	0.02562	0.02923	0.1467	0.05863
855133	M	14.99	25.2	95.54	698.8	0.09387	0.05131	0.02398	0.02899	0.1565	0.05504
855138	M	13.48	20.82	88.4	559.2	0.1016	0.1255	0.1063	0.05439	0.172	0.06419
855167	M	13.44	21.58	86.18	563	0.08162	0.06031	0.0311	0.02031	0.1784	0.05587
855563	M	10.95	21.35	71.9	371.1	0.1227	0.1218	0.1044	0.05669	0.1895	0.0687
855625	M	19.07	24.81	128.3	1104	0.09081	0.219	0.2107	0.09961	0.231	0.06343
856106	M	13.28	20.28	87.32	545.2	0.1041	0.1436	0.09847	0.06158	0.1974	0.06782
85638502	M	13.17	21.81	85.42	531.5	0.09714	0.1047	0.08259	0.05252	0.1746	0.06177
857010	M	18.65	17.6	123.7	1076	0.1099	0.1686	0.1974	0.1009	0.1907	0.06049
85713702	B	8.196	16.84	51.71	201.9	0.086	0.05943	0.01588	0.005917	0.1769	0.06503
85715	M	13.17	18.66	85.98	534.6	0.1158	0.1231	0.1226	0.0734	0.2128	0.06777
857155	B	12.05	14.63	78.04	449.3	0.1031	0.09092	0.06592	0.02749	0.1675	0.06043
857156	B	13.49	22.3	86.91	561	0.08752	0.07698	0.04751	0.03384	0.1809	0.05718
857343	B	11.76	21.6	74.72	427.9	0.08637	0.04966	0.01657	0.01115	0.1495	0.05888
857373	B	13.64	16.34	87.21	571.8	0.07685	0.06059	0.01857	0.01723	0.1353	0.05953
857374	B	11.94	18.24	75.71	437.6	0.08261	0.04751	0.01972	0.01349	0.1868	0.0611
857392	M	18.22	18.7	120.3	1033	0.1148	0.1485	0.1772	0.106	0.2092	0.0631
857438	M	15.1	22.02	97.26	712.8	0.09056	0.07081	0.05253	0.03334	0.1616	0.05684
85759902	B	11.52	18.75	73.34	409	0.09524	0.05473	0.03036	0.02278	0.192	0.05907
857637	M	19.21	18.57	125.5	1152	0.1053	0.1267	0.1323	0.08994	0.1917	0.05961
857793	M	14.71	21.59	95.55	656.9	0.1137	0.1365	0.1293	0.08123	0.2027	0.06758
857810	B	13.05	19.31	82.61	527.2	0.0806	0.03789	0.000692	0.004167	0.1819	0.05501
858477	B	8.618	11.79	54.34	224.5	0.09752	0.05272	0.02061	0.007799	0.1683	0.07187
858970	B	10.17	14.88	64.55	311.9	0.1134	0.08061	0.01084	0.0129	0.2743	0.0696
858981	B	8.598	20.98	54.66	221.8	0.1243	0.08963	0.03	0.009259	0.1828	0.06757
858986	M	14.25	22.15	96.42	645.7	0.1049	0.2008	0.2135	0.08653	0.1949	0.07292

id	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se
842302	1.095	0.9053	8.589	153.4	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193
842517	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.0186	0.0134	0.01389	0.003532
84300903	0.7456	0.7869	4.585	94.03	0.00615	0.04006	0.03832	0.02058	0.0225	0.004571
84348301	0.4956	1.156	3.445	27.23	0.00911	0.07458	0.05661	0.01867	0.05963	0.009208
84358402	0.7572	0.7813	5.438	94.44	0.01149	0.02461	0.05688	0.01885	0.01756	0.005115
843786	0.3345	0.8902	2.217	27.19	0.00751	0.03345	0.03672	0.01137	0.02165	0.005082
844359	0.4467	0.7732	3.18	53.91	0.004314	0.01382	0.02254	0.01039	0.01369	0.002179
84458202	0.5835	1.377	3.856	50.96	0.008805	0.03029	0.02488	0.01448	0.01486	0.005412
844981	0.3063	1.002	2.406	24.32	0.005731	0.03502	0.03553	0.01226	0.02143	0.003749
84501001	0.2976	1.599	2.039	23.94	0.007149	0.07217	0.07743	0.01432	0.01789	0.01008
845636	0.3795	1.187	2.466	40.51	0.004029	0.009269	0.01101	0.007591	0.0146	0.003042
84610002	0.5058	0.9849	3.564	54.16	0.005771	0.04061	0.02791	0.01282	0.02008	0.004144
846226	0.9555	3.568	11.07	116.2	0.003139	0.08297	0.0889	0.0409	0.04484	0.01284
846381	0.4033	1.078	2.903	36.58	0.009769	0.03126	0.05051	0.01992	0.02981	0.003002
84667401	0.2121	1.169	2.061	19.21	0.006429	0.05936	0.05501	0.01628	0.01961	0.008093
84799002	0.37	1.033	2.879	32.55	0.005607	0.0424	0.04741	0.0109	0.01857	0.005466
848406	0.4727	1.24	3.195	45.4	0.005718	0.01162	0.01998	0.01109	0.0141	0.002085
84862001	0.5692	1.073	3.854	54.18	0.007026	0.02501	0.03188	0.01297	0.01689	0.004142
849014	0.7582	1.017	5.865	112.4	0.006494	0.01893	0.03391	0.01521	0.01356	0.001997
8510426	0.2699	0.7886	2.058	23.56	0.008462	0.0146	0.02387	0.01315	0.0198	0.0023
8510653	0.1852	0.7477	1.383	14.67	0.004097	0.01898	0.01698	0.00649	0.01678	0.002425
8510824	0.2773	0.9768	1.909	15.7	0.009606	0.01432	0.01985	0.01421	0.02027	0.002968
8511133	0.4388	0.7096	3.384	44.91	0.006789	0.05328	0.06446	0.02252	0.03672	0.004394
851509	0.6917	1.127	4.303	93.99	0.004728	0.01259	0.01715	0.01038	0.01083	0.001987
852552	0.8068	0.9017	5.455	102.6	0.006048	0.01882	0.02741	0.0113	0.01468	0.002801
852631	1.046	0.976	7.276	111.4	0.008029	0.03799	0.03732	0.02397	0.02308	0.007444
852763	0.2545	0.9832	2.11	21.05	0.004452	0.03055	0.02681	0.01352	0.01454	0.003711
852781	0.8529	1.849	5.632	93.54	0.01075	0.02722	0.05081	0.01911	0.02293	0.004217
852973	0.439	1.012	3.498	43.5	0.005233	0.03057	0.03576	0.01083	0.01768	0.002967
853201	0.6003	0.8225	4.655	61.1	0.005627	0.03033	0.03407	0.01354	0.01925	0.003742
853401	0.8307	1.466	5.574	105	0.006248	0.03374	0.05196	0.01158	0.02007	0.00456
853612	0.4825	1.03	3.475	41	0.005551	0.03414	0.04205	0.01044	0.02273	0.005667
85382601	0.6009	1.398	3.999	67.78	0.008268	0.03082	0.05042	0.01112	0.02102	0.003854
854002	0.5558	0.6062	3.528	68.17	0.005015	0.03318	0.03497	0.009643	0.01543	0.003896
854039	0.334	0.6857	2.183	35.03	0.004185	0.02868	0.02664	0.009067	0.01703	0.003817
854253	0.4615	0.9197	3.008	45.19	0.005776	0.02499	0.03695	0.01195	0.02789	0.002665
854268	0.286	1.019	2.657	24.91	0.005878	0.02995	0.04815	0.01161	0.02028	0.004022
854941	0.1839	2.342	1.17	14.16	0.004352	0.004899	0.01343	0.01164	0.02671	0.001777
855133	1.214	2.188	8.077	106	0.006883	0.01094	0.01818	0.01917	0.007882	0.001754
855138	0.213	0.5914	1.545	18.52	0.005367	0.02239	0.03049	0.01262	0.01377	0.003187
855167	0.2385	0.8265	1.572	20.53	0.00328	0.01102	0.0139	0.006881	0.0138	0.001286
855563	0.2366	1.428	1.822	16.97	0.008064	0.01764	0.02595	0.01037	0.01357	0.00304
855625	0.9811	1.666	8.83	104.9	0.006548	0.1006	0.09723	0.02638	0.05333	0.007646
856106	0.3704	0.8249	2.427	31.33	0.005072	0.02147	0.02185	0.00956	0.01719	0.003317
85638502	0.1938	0.6123	1.334	14.49	0.00335	0.01384	0.01452	0.006853	0.01113	0.00172
857010	0.6289	0.6633	4.293	71.56	0.006294	0.03994	0.05554	0.01695	0.02428	0.003535
85713702	0.1563	0.9567	1.094	8.205	0.008968	0.01646	0.01588	0.005917	0.02574	0.002582
85715	0.2871	0.8937	1.897	24.25	0.006532	0.02336	0.02905	0.01215	0.01743	0.003643
857155	0.2636	0.7294	1.848	19.87	0.005488	0.01427	0.02322	0.00566	0.01428	0.002422
857156	0.2338	1.353	1.735	20.2	0.004455	0.01382	0.02095	0.01184	0.01641	0.001956
857343	0.4062	1.21	2.635	28.47	0.005857	0.009758	0.01168	0.007445	0.02406	0.001769
857373	0.1872	0.9234	1.449	14.55	0.004477	0.01177	0.01079	0.007956	0.01325	0.002551
857374	0.2273	0.6329	1.52	17.47	0.00721	0.00838	0.01311	0.008	0.01996	0.002635
857392	0.8337	1.593	4.877	98.81	0.003899	0.02961	0.02817	0.009222	0.02674	0.005126
857438	0.3105	0.8339	2.097	29.91	0.004675	0.0103	0.01603	0.009222	0.01095	0.001629
85759902	0.3249	0.9591	2.183	23.47	0.008328	0.008722	0.01349	0.00867	0.03218	0.002386
857637	0.7275	1.193	4.837	102.5	0.006458	0.02306	0.02945	0.01538	0.01852	0.002608
857793	0.4226	1.15	2.735	40.09	0.003659	0.02855	0.02572	0.01272	0.01817	0.004108
857810	0.404	1.214	2.595	32.96	0.007491	0.008593	0.000692	0.004167	0.0219	0.00299
858477	0.1559	0.5796	1.046	8.322	0.01011	0.01055	0.01981	0.005742	0.0209	0.002788
858970	0.5158	1.441	3.312	34.62	0.007514	0.01099	0.007665	0.008193	0.04183	0.005953
858981	0.3582	2.067	2.493	18.39	0.01193	0.03162	0.03	0.009259	0.03357	0.003048
858986	0.7036	1.268	5.373	60.78	0.009407	0.07056	0.06899	0.01848	0.017	0.006113

id	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
842302	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601	0.1189
842517	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275	0.08902
84300903	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613	0.08758
84348301	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.173
84358402	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364	0.07678
843786	15.47	23.75	103.4	741.6	0.1791	0.5249	0.5355	0.1741	0.3985	0.1244
844359	22.88	27.66	153.2	1606	0.1442	0.2576	0.3784	0.1932	0.3063	0.08368
84458202	17.06	28.14	110.6	897	0.1654	0.3682	0.2678	0.1556	0.3196	0.1151
844981	15.49	30.73	106.2	739.3	0.1703	0.5401	0.539	0.206	0.4378	0.1072
84501001	15.09	40.68	97.65	711.4	0.1853	1.058	1.105	0.221	0.4366	0.2075
845636	19.19	33.88	123.8	1150	0.1181	0.1551	0.1459	0.09975	0.2948	0.08452
84610002	20.42	27.28	136.5	1299	0.1396	0.5609	0.3965	0.181	0.3792	0.1048
846226	20.96	29.94	151.7	1332	0.1037	0.3903	0.3639	0.1767	0.3176	0.1023
846381	16.84	27.66	112	876.5	0.1131	0.1924	0.2322	0.1119	0.2809	0.06287
84667401	15.03	32.01	108.8	697.7	0.1651	0.7725	0.6943	0.2208	0.3596	0.1431
84799002	17.46	37.13	124.1	943.2	0.1678	0.6577	0.7026	0.1712	0.4218	0.1341
848406	19.07	30.88	123.4	1138	0.1464	0.1871	0.2914	0.1609	0.3029	0.08216
84862001	20.96	31.48	136.8	1315	0.1789	0.4233	0.4784	0.2073	0.3706	0.1142
849014	27.32	30.88	186.8	2398	0.1512	0.315	0.5372	0.2388	0.2768	0.07615
8510426	15.11	19.26	99.7	711.2	0.144	0.1773	0.239	0.1288	0.2977	0.07259
8510653	14.5	20.49	96.09	630.5	0.1312	0.2776	0.189	0.07283	0.3184	0.08183
8510824	10.23	15.66	65.13	314.9	0.1324	0.1148	0.08867	0.06227	0.245	0.07773
8511133	18.07	19.08	125.1	980.9	0.139	0.5954	0.6305	0.2393	0.4667	0.09946
851509	29.17	35.59	188	2615	0.1401	0.26	0.3155	0.2009	0.2822	0.07526
852552	26.46	31.56	177	2215	0.1805	0.3578	0.4695	0.2095	0.3613	0.09564
852631	22.25	21.4	152.4	1461	0.1545	0.3949	0.3853	0.255	0.4066	0.1059
852763	17.62	33.21	122.4	896.9	0.1525	0.6643	0.5539	0.2701	0.4264	0.1275
852781	21.31	27.26	139.9	1403	0.1338	0.2117	0.3446	0.149	0.2341	0.07421
852973	20.27	36.71	149.3	1269	0.1641	0.611	0.6335	0.2024	0.4027	0.09876
853201	20.01	19.52	134.9	1227	0.1255	0.2812	0.2489	0.1456	0.2756	0.07919
853401	23.15	34.01	160.5	1670	0.1491	0.4257	0.6133	0.1848	0.3444	0.09782
853612	16.82	28.12	119.4	888.7	0.1637	0.5775	0.6956	0.1546	0.4761	0.1402
85382601	20.88	32.09	136.1	1344	0.1634	0.3559	0.5588	0.1847	0.353	0.08482
854002	24.15	30.9	161.4	1813	0.1509	0.659	0.6091	0.1785	0.3672	0.1123
854039	20.21	27.26	132.7	1261	0.1446	0.5804	0.5274	0.1864	0.427	0.1233
854253	20.01	29.02	133.5	1229	0.1563	0.3835	0.5409	0.1813	0.4863	0.08633
854268	15.89	30.36	116.2	799.6	0.1446	0.4238	0.5186	0.1447	0.3591	0.1014
854941	13.3	22.81	84.46	545.9	0.09701	0.04619	0.04833	0.05013	0.1987	0.06169
855133	14.99	25.2	95.54	698.8	0.09387	0.05131	0.02398	0.02899	0.1565	0.05504
855138	15.53	26.02	107.3	740.4	0.161	0.4225	0.503	0.2258	0.2807	0.1071
855167	15.93	30.25	102.5	787.9	0.1094	0.2043	0.2085	0.1112	0.2994	0.07146
855563	12.84	35.34	87.22	514	0.1909	0.2698	0.4023	0.1424	0.2964	0.09606
855625	24.09	33.17	177.4	1651	0.1247	0.7444	0.7242	0.2493	0.467	0.1038
856106	17.38	28	113.1	907.2	0.153	0.3724	0.3664	0.1492	0.3739	0.1027
85638502	16.23	29.89	105.5	740.7	0.1503	0.3904	0.3728	0.1607	0.3693	0.09618
857010	22.82	21.32	150.6	1567	0.1679	0.509	0.7345	0.2378	0.3799	0.09185
85713702	8.964	21.96	57.26	242.2	0.1297	0.1357	0.0688	0.02564	0.3105	0.07409
85715	15.67	27.95	102.8	759.4	0.1786	0.4166	0.5006	0.2088	0.39	0.1179
857155	13.76	20.7	89.88	582.6	0.1494	0.2156	0.305	0.06548	0.2747	0.08301
857156	15.15	31.82	99	698.8	0.1162	0.1711	0.2282	0.1282	0.2871	0.06917
857343	12.98	25.72	82.98	516.5	0.1085	0.08615	0.05523	0.03715	0.2433	0.06563
857373	14.67	23.19	96.08	656.7	0.1089	0.1582	0.105	0.08586	0.2346	0.08025
857374	13.1	21.33	83.67	527.2	0.1144	0.08906	0.09203	0.06296	0.2785	0.07408
857392	20.6	24.13	135.1	1321	0.128	0.2297	0.2623	0.1325	0.3021	0.07987
857438	18.1	31.69	117.7	1030	0.1389	0.2057	0.2712	0.153	0.2675	0.07873
85759902	12.84	22.47	81.81	506.2	0.1249	0.0872	0.09076	0.06316	0.3306	0.07036
857637	26.14	28.14	170.1	2145	0.1624	0.3511	0.3879	0.2091	0.3537	0.08294
857793	17.87	30.7	115.7	985.5	0.1368	0.429	0.3587	0.1834	0.3698	0.1094
857810	14.23	22.25	90.24	624.1	0.1021	0.06191	0.001845	0.01111	0.2439	0.06289
858477	9.507	15.4	59.9	274.9	0.1733	0.1239	0.1168	0.04419	0.322	0.09026
858970	11.02	17.45	69.86	368.6	0.1275	0.09866	0.02168	0.02579	0.3557	0.0802
858981	9.565	27.04	62.06	273.9	0.1639	0.1698	0.09001	0.02778	0.2972	0.07712
858986	17.67	29.51	119.1	959.5	0.164	0.6247	0.6922	0.1785	0.2844	0.1132

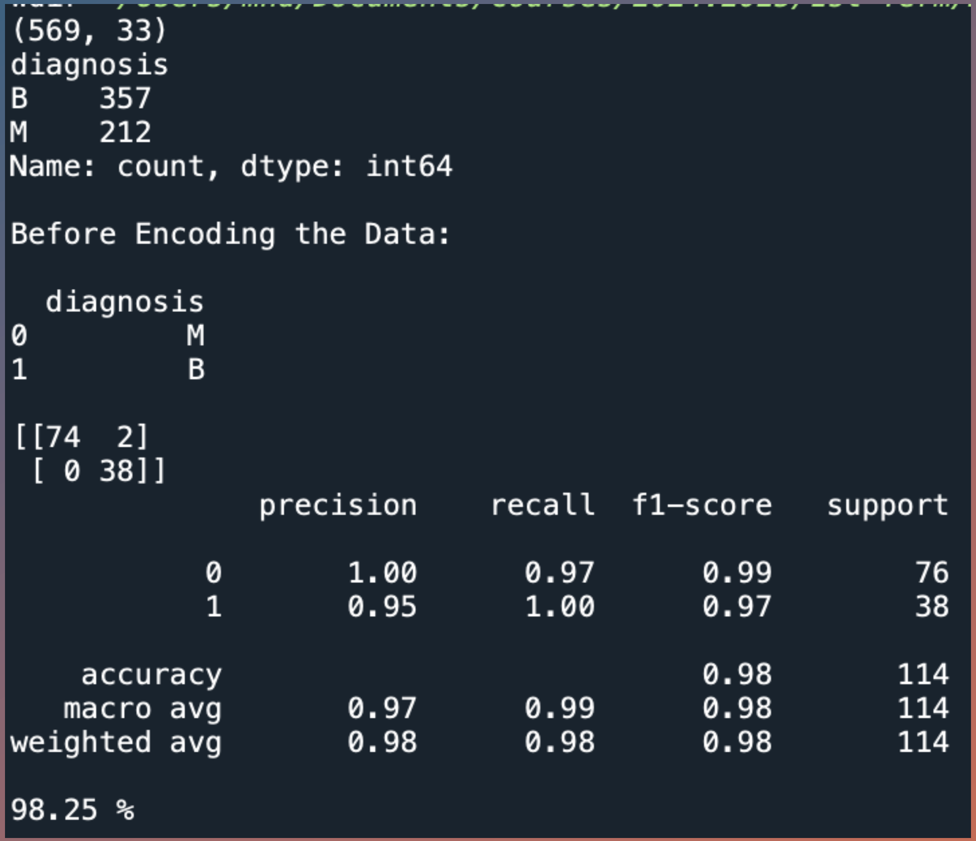
Example III:

```
import pandas as pd
from sklearn.model_selection import train_test_split #To split data into random train and test subsets.
from sklearn.linear_model import LogisticRegression #Logistic Regression classifier.
from sklearn.preprocessing import StandardScaler #Removing the mean and scaling to unit variance.
from sklearn.metrics import classification_report,accuracy_score #Build Report showing the main classification metrics.
from sklearn.metrics import confusion_matrix #Build Confusion Matrix

cancer = pd.read_csv('data.csv')

print(cancer.shape)

print(cancer['diagnosis'].value_counts())

cancer['diagnosis'].value_counts().plot(kind='bar’)

from sklearn.preprocessing import LabelEncoder

#Create a sample dataframe with categorical data
diagnosiss = pd.DataFrame({'diagnosis': ['M', 'B']})

print(f"\nBefore Encoding the Data:\n\n{diagnosiss}\n")

#Create a LabelEncoder object
le = LabelEncoder()
```

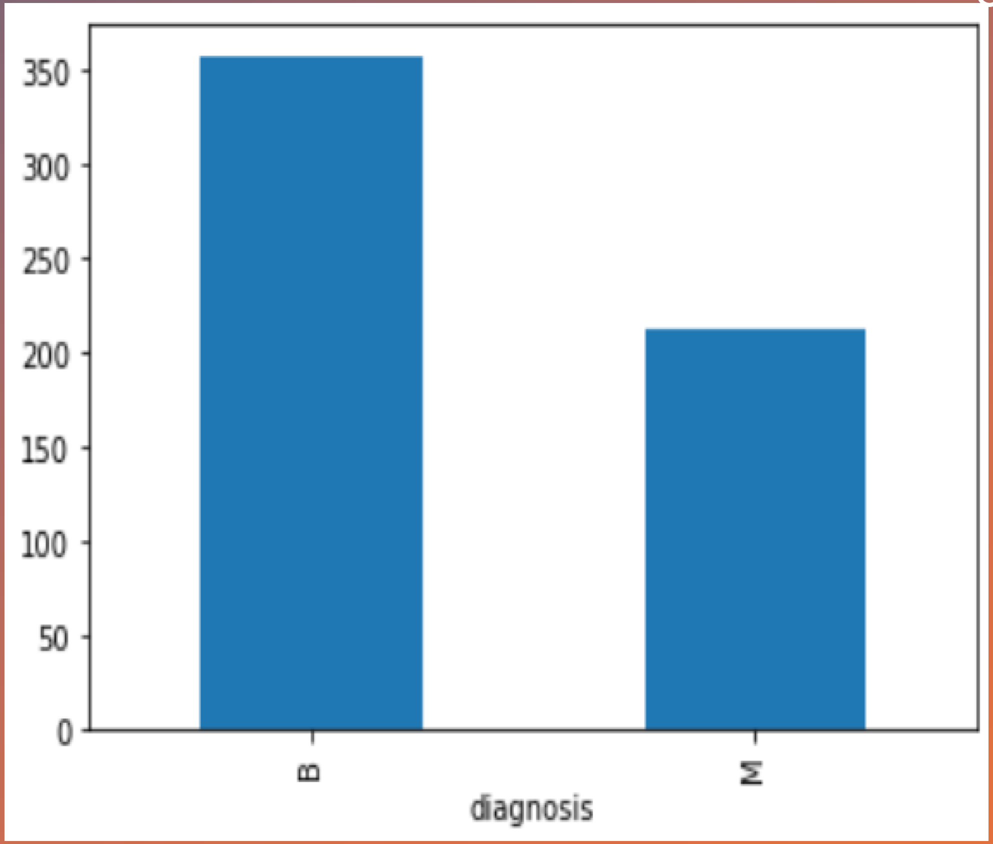
Example III:

```
#Fit and transform the categorical data
cancer['diagnosis'] = le.fit_transform(cancer['diagnosis'])

#Build Model
x = cancer.drop(['diagnosis','id','Unnamed: 32'],axis=1)
y = cancer['diagnosis']

x_train,x_test,y_train,y_test = train_test_split(x, y, test_size=0.2, random_state=11)
s = StandardScaler()

#Learn the parameters and apply the transformation to new data
x_train = s.fit_transform(x_train)
x_test = s.fit_transform(x_test)

model = LogisticRegression()
model.fit(x_train,y_train)

predict_test = model.predict(x_test)
#print(predict_test)

print(confusion_matrix(y_test,predict_test))
print(classification_report(y_test,predict_test))
print(round(accuracy_score(y_test,predict_test)*100,2),"%")
```

Example III:

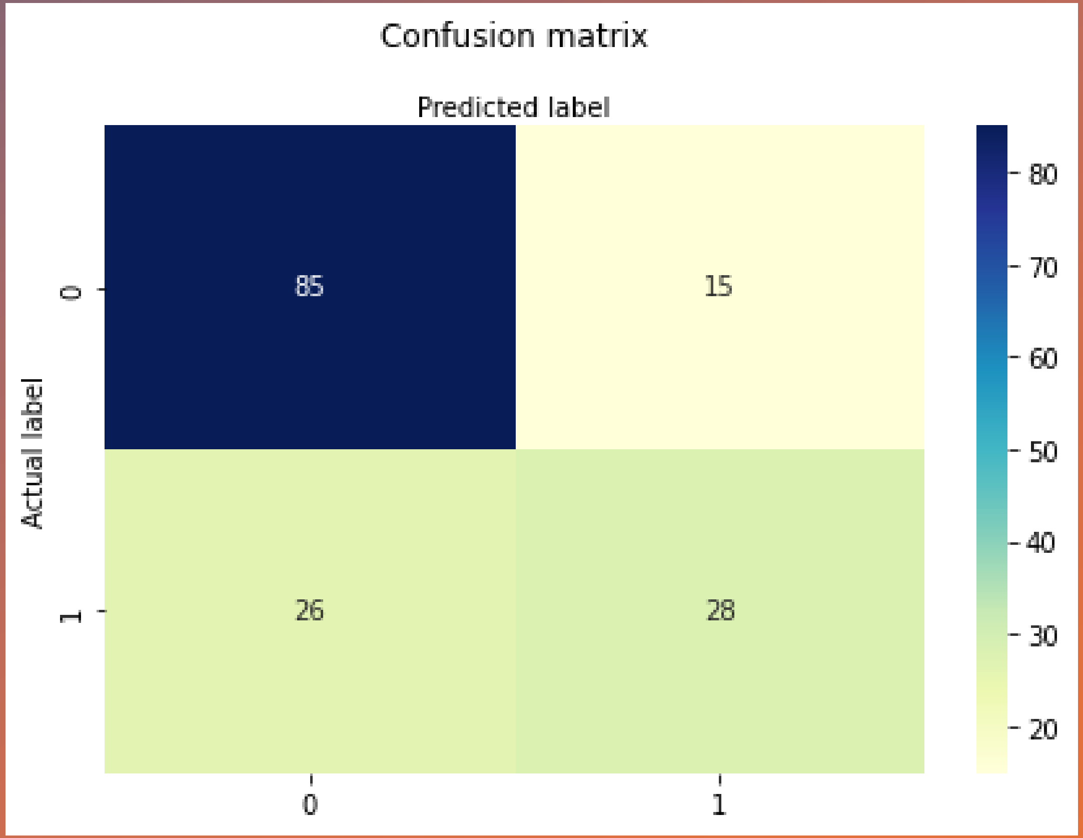


```
(569, 33)
diagnosis
B    357
M    212
Name: count, dtype: int64

Before Encoding the Data:

  diagnosis
0         M
1         B

[[74  2]
 [ 0 38]]
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        76
           1       0.95      1.00      0.97        38

    accuracy                           0.98       114
   macro avg       0.97      0.99      0.98       114
weighted avg       0.98      0.98      0.98       114

98.25 %
```



Example III:

```
import seaborn as sns
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

sns.heatmap(pd.DataFrame(confusion_matrix(y_test,pred
ict_test)), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

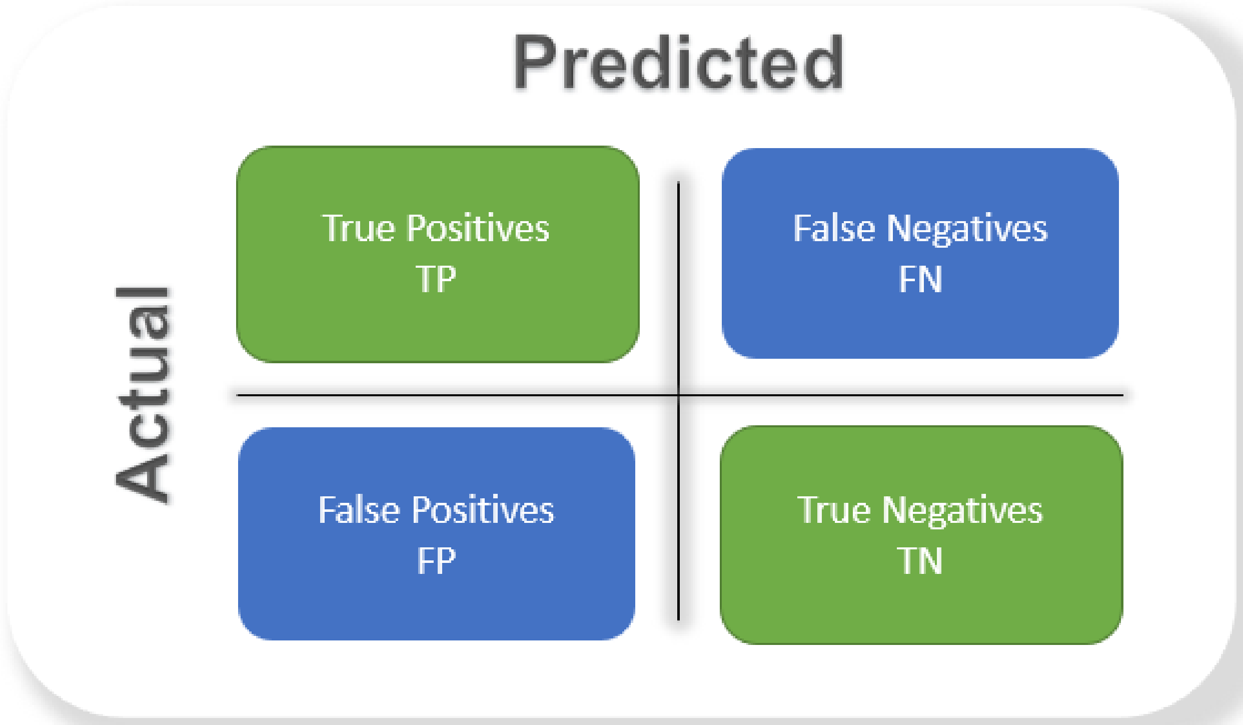


How many **wrong prediction we have => 15+26**

Model Evaluation:

- **True positives (TP)**: Predicted positive and are actually positive.
- **False positives (FP)**: Predicted positive and are actually negative.
- **True negatives (TN)**: Predicted negative and are actually negative.
- **False negatives (FN)**: Predicted negative and are actually positive.

Confusion Matrix:

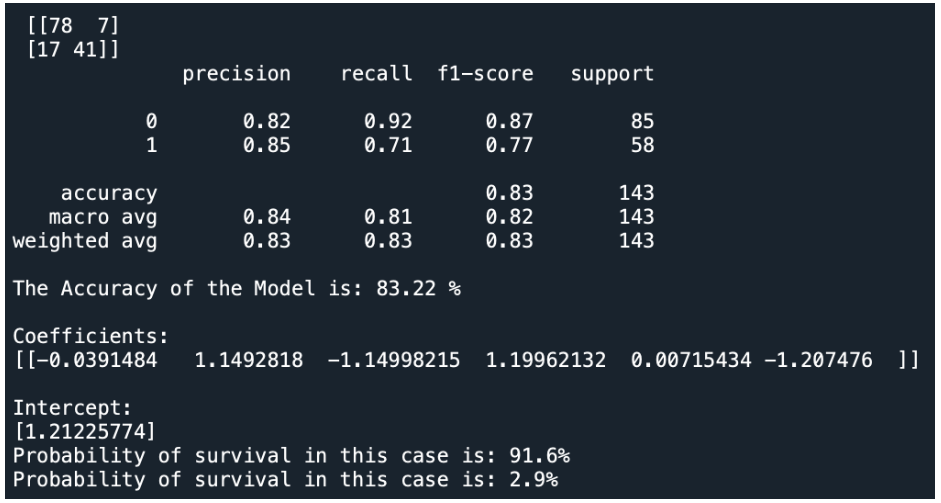


Accuracy:

The most commonly used metric to judge a model and is actually not a clear indicator of the performance.

$$= \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476  ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

The most commonly used metric to judge a model and is actually not a clear indicator of the performance.

$$= \frac{TP + TN}{TP + FP + TN + FN}$$

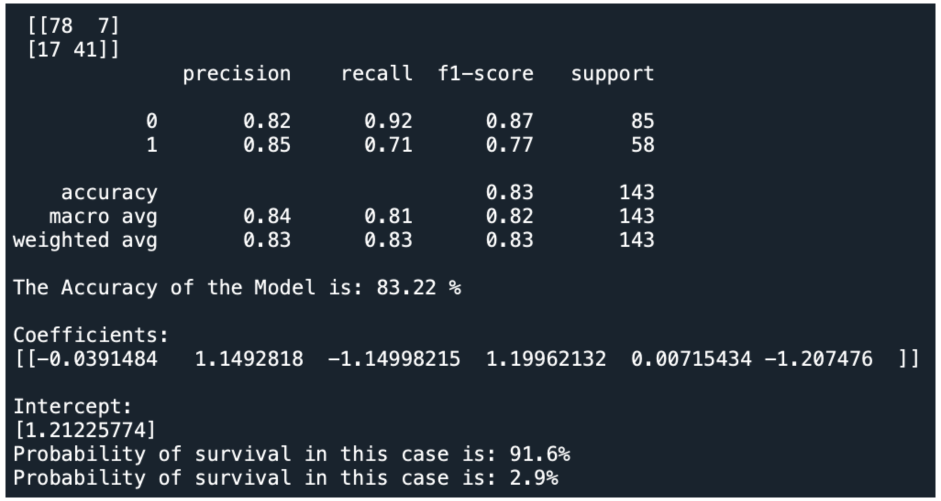
$$= \frac{78+41}{78+17+41+7} = 0.83$$

Precision:

Percentage of positive instances out of the total predicted positive instances. Here denominator is the model prediction done as positive from the whole given dataset. Take it as to find out '*how much the model is right when it says it is right*'.

$$= \frac{TP}{TP + FP}$$

Precision:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476   ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

Percentage of positive instances out of the total predicted positive instances. Here denominator is the model prediction done as positive from the whole given dataset. Take it as to find out '*how much the model is right when it says it is right*'.

$$= \frac{TP}{TP + FP}$$

$$= \frac{78}{78+17} = 0.82 \text{ (For 0)}$$

$$= \frac{TN}{TN+FN} = \frac{41}{41+7} = 0.85 \text{ (For 1)}$$

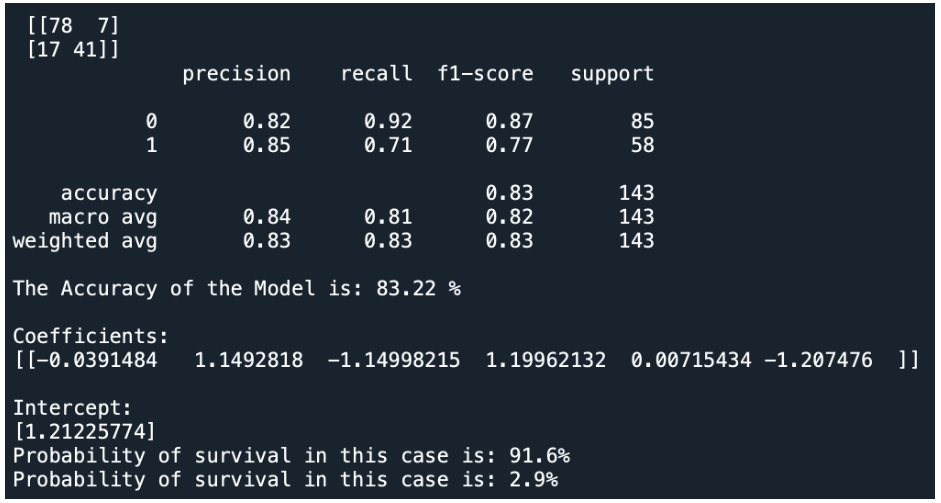
- It means that the predicating was correct in 82% when predicting "Not Survived".
- It means that the predicating was correct in 85% when predicting "Survived".

Recall/Sensitivity/True Positive Rate:

Percentage of positive instances out of the ***total actual positive*** instances. Therefore denominator (*TP* + *FN)* here is the *actual* number of positive instances present in the dataset. Take it as to find out '*how much extra right ones, the model missed when it showed the right ones*'.

$$= \frac{TP}{TP + FN}$$

Recall/Sensitivity/True Positive Rate:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476  ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

Percentage of positive instances out of the ***total actual positive*** instances. Therefore denominator (*TP* + *FN)* here is the *actual* number of positive instances present in the dataset. Take it as to find out '*how much extra right ones, the model missed when it showed the right ones'*.

$$= \frac{TP}{TP + FN}$$

$$= \frac{78}{78+7} = 0.92 \text{ (For 0)}$$

$$= \frac{TN}{TN+FP} = \frac{41}{41+17} = 0.71 \text{ (For 1)}$$

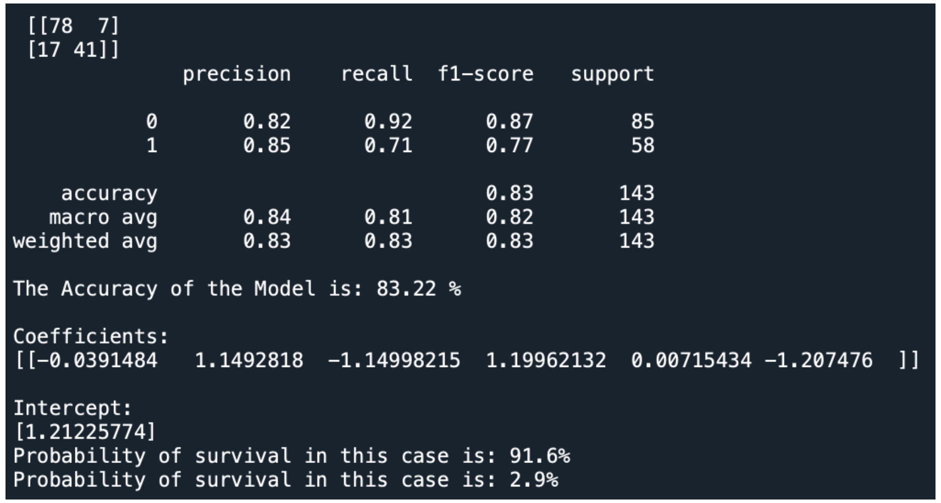
- The model correctly found 92% of "Not Survived" case.
- The model correctly found 71% of "Survived" case.

F1 Score:

It is the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. See that due to the product in the numerator if one goes low, the final F1 score goes down significantly. So, a model does well in F1 score if the positive predicted are actually positives (precision) and doesn't miss out on positives and predicts them negative (recall).

$$\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall}$$

F1 Score:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476  ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

- The F1-score of 87% indicates a good balance between precision and recall when predicting "Not Survived".
- The F1-score of 77% suggests that there's room for improvement, particularly in capturing more "Survived" cases (improving recall).

It is the harmonic mean of precision and recall. This takes the contribution of both, so higher the F1 score, the better. See that due to the product in the numerator if one goes low, the final F1 score goes down significantly. So a model does well in F1 score if the positive predicted are actually positives (precision) and doesn't miss out on positives and predicts them negative (recall).

$$\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} = \frac{2 * Precision\ * Recall}{Precision + Recall}$$

$$= \frac{2 * 0.82 * 0.92}{0.82+0.92} = 0.87 \text{ (For 0)}$$

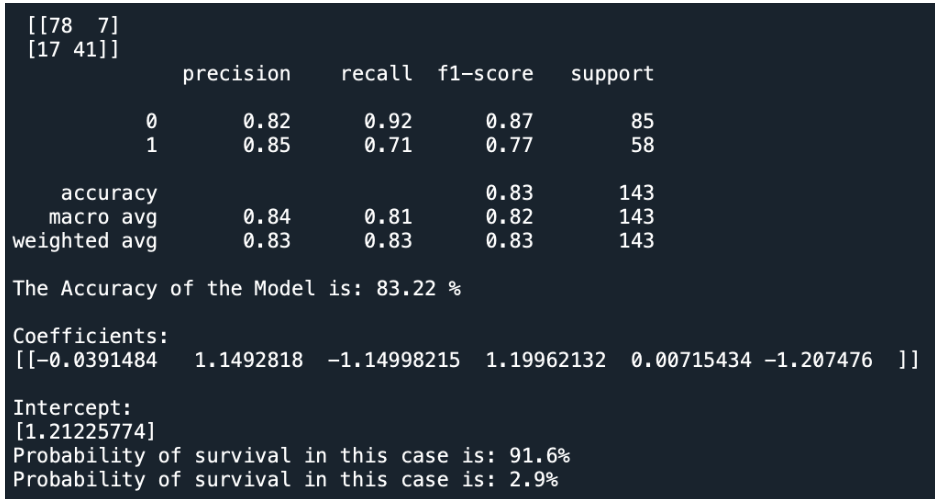
$$= \frac{2 * 0.85 * 0.71}{0.85+0.71} = 0.77 \text{ (For 1)}$$

Macro AVG:

It is perhaps the most straightforward among the numerous averaging methods. The macro-averaged F1 score (or macro F1 score) is computed by taking the arithmetic mean (aka **unweighted** mean) of all the per-class F1 scores. This method treats all classes equally regardless of their **support** values.

$$= \frac{F1_{classi} + F1_{classj} + F1_{class..}}{No.of\ Classes}$$

Macro AVG:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476   ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

It is perhaps the most straightforward among the numerous averaging methods. The macro-averaged F1 score (or macro F1 score) is computed by taking the arithmetic mean (aka **unweighted** mean) of all the per-class F1 scores (*it takes the average of the precision, recall, and F1-scores across all classes, treating each class equally*). This method treats all classes equally regardless of their **support** values.

$$= \frac{F1_{classi} + F1_{classj} + F1_{class..}}{No.of\ Classes}$$

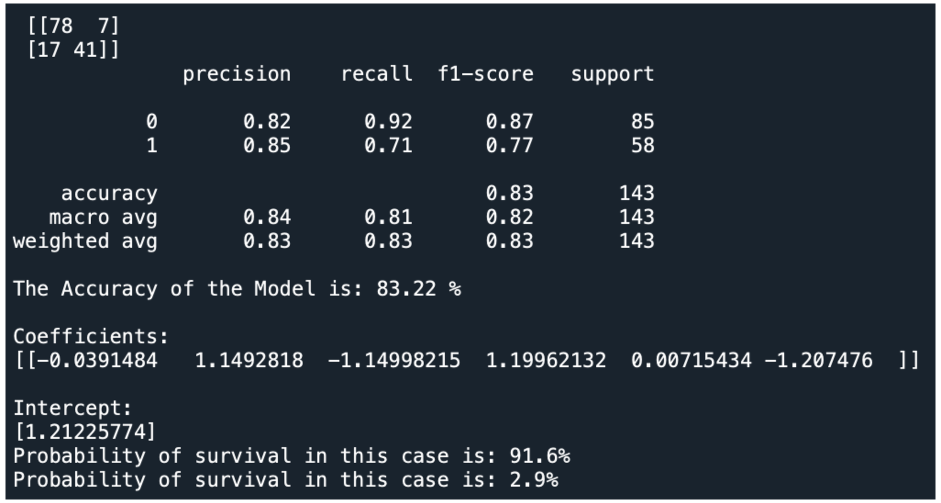
$$= \frac{0.87 + 0.77}{2} = 0.82$$

Weighted Average:

is calculated by taking the mean of all per-class F1 scores **while considering each class's support**. **Support** refers to the number of actual occurrences of the class in the dataset.

= $(F1_{classi} * Supportclassi) + (F1_{classj} * Supportclassj) + (F1_{class_{..}} * Supportclass_{..})$

Weighted Average:



```
 [[78  7]
 [17 41]]
              precision    recall  f1-score   support

           0       0.82      0.92      0.87        85
           1       0.85      0.71      0.77        58

    accuracy                           0.83       143
   macro avg       0.84      0.81      0.82       143
weighted avg       0.83      0.83      0.83       143

The Accuracy of the Model is: 83.22 %

Coefficients:
[[-0.0391484   1.1492818  -1.14998215  1.19962132  0.00715434 -1.207476  ]]

Intercept:
[1.21225774]
Probability of survival in this case is: 91.6%
Probability of survival in this case is: 2.9%
```

is calculated by taking the mean of all per-class F1 scores ***while considering each class's support***. ***Support*** refers to the number of actual occurrences of the class in the dataset.

$$= (F1_{classi} * Supportclassi) + (F1_{classj} * Supportclassj) + (F1_{class_{..}} * Supportclass_{..})$$

$$= 0.87 * \frac{85}{143} + 0.77 * \frac{58}{143} = 0.83$$

Model Evaluation Example (Confusion Matrix and Accuracy):

No	Actual	Predicted	Match
1	1	0	FN
2	1	0	FN
3	1	1	TP
4	1	1	TP
5	1	1	TP
6	1	1	TP
7	1	1	TP
8	1	1	TP
9	0	1	FP
10	0	0	TN
11	0	0	TN
12	0	0	TN

Model Evaluation Example (Confusion Matrix and Accuracy):

		Predicted condition	
	Total 8 + 4 = 12	**Cancer** 7	**Non-cancer** 5
Actual condition	**Cancer** 8	6_{TP}	2_{FN}
	Non-cancer 4	1_{FP}	3_{TN}

Model Evaluation Example:

No	Actual	Predicted	Match
1	Airplane	Airplane	✓
2	Car	Boat	✗
3	Car	Car	✓
4	Car	Car	✓
5	Car	Boat	✗
6	Airplane	Boat	✗
7	Boat	Boat	✓
8	Car	Airplane	✗
9	Airplane	Airplane	✓
10	Car	Car	✓

Model Evaluation Example:

	Predicted			
	Label	Airplane	Boat	Car
Actual	Airplane	2	1	0
	Boat	0	1	0
	Car	1	2	3

Model Evaluation Example:

Label	True Positive (TP)	False Positive (FP)	False Negative (FN)
Airplane	2$_{(A-A)}$	1$_{(X-A)}$	1$_{(A-X)}$
Boat	1$_{(B-B)}$	3$_{(X-B)}$	0$_{(B-X)}$
Car	3$_{(C-C)}$	0$_{(X-C)}$	3$_{(C-X)}$

Model Evaluation Example:

Label	TP	FP	FN	Precision	Recall	F1 Score
Airplane	2	1	1	0.67	0.67	2*(0.67*0.67)/(0.67+0.67) = **0.67**
Boat	1	3	0	0.25	1.00	2*(0.25*1.00)/(0.25+1.00) = **0.40**
Car	3	0	3	1.00	0.50	2*(1.00*0.50)/(1.00+0.50) = **0.67**

Model Evaluation Example:

Label	F1 Score	Macro-AVG
Airplane	0.67	$\frac{0.67 + 0.40 + 0.6}{3} = 0.58$
Boat	0.40	
Car	0.67	

Model Evaluation Example:

Label	F1 Score	Support	Support Proportion	Weighted Average F1
Airplane	0.67	3	0.3	(0.67 * 0.3) + (0.40 * 0.1) + (0.67 * 0.6) = **0.64**
Boat	0.40	1	0.1	
Car	0.67	6	0.6	
Total	-	10	1.0	

Note that, we didn't do the confusion matrix and accuracy measurements for this data! (FT is missing)

Thanks