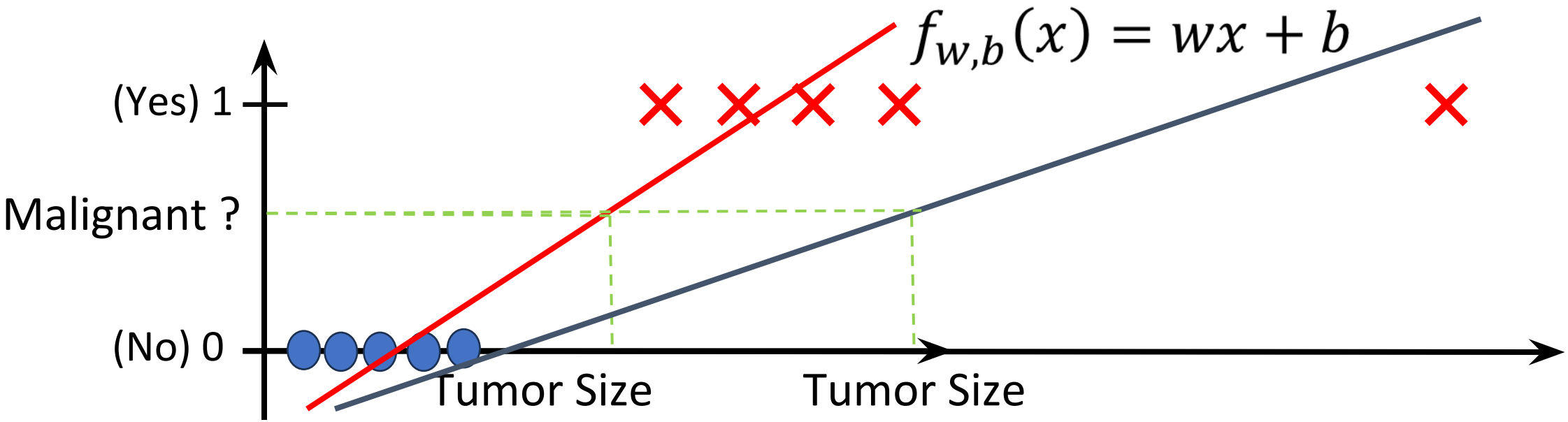
Logistic Regression

Supervised Learning Algorithm- Classification

Classification

Question (x)	Ans (y)	
Is email Spam or not?	Yes	No
Is transaction fraudulent or not?	Yes	No
Is tumor malignant?	Yes	No

y can only have two values **Binary classification**

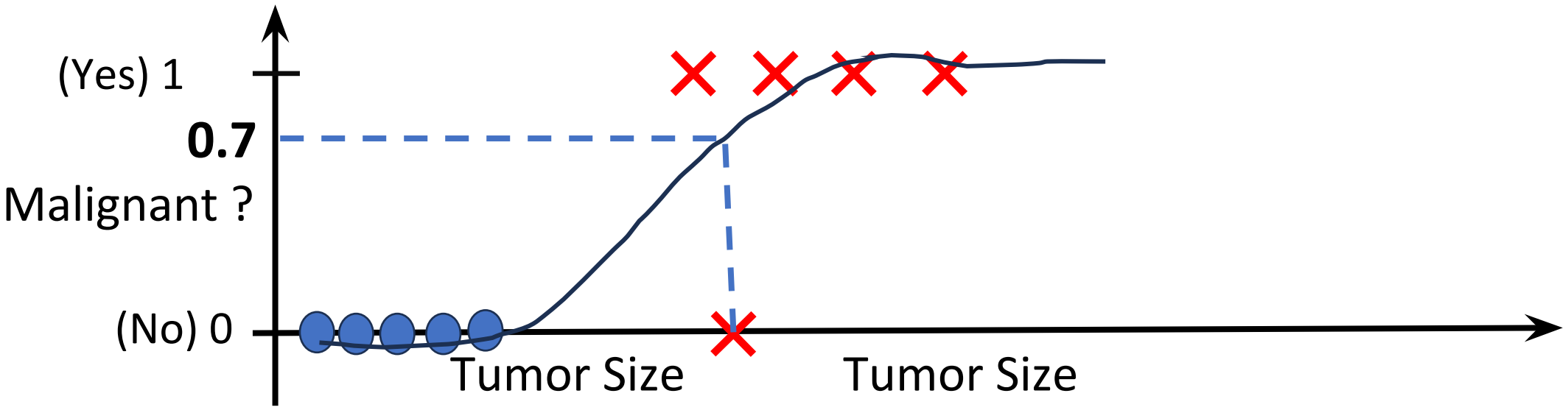


Threshold classifier output $f_{w,b}(x)$ at 0.5:

if $f_{w,b}(x)$>= 0.5 then y=1

if $f_{w,b}(x)$< 0.5 then y=0

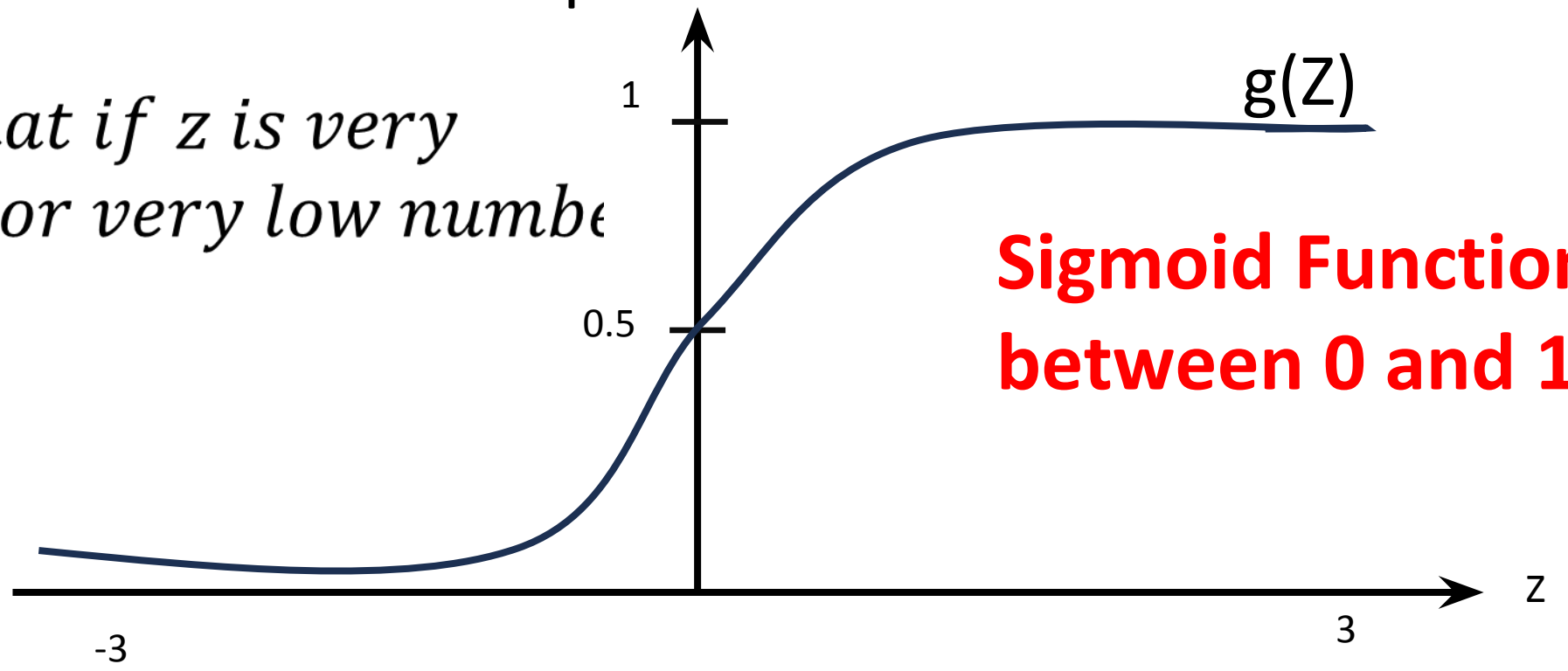
Apply Linear Regression in Classification problem


(Yes) 1
0.7
Malignant ?
(No) 0
Tumor Size
Tumor Size

Sigmoid Function/Logistic Function

Wants output between 0 and 1

what if z is very large or very low numbe

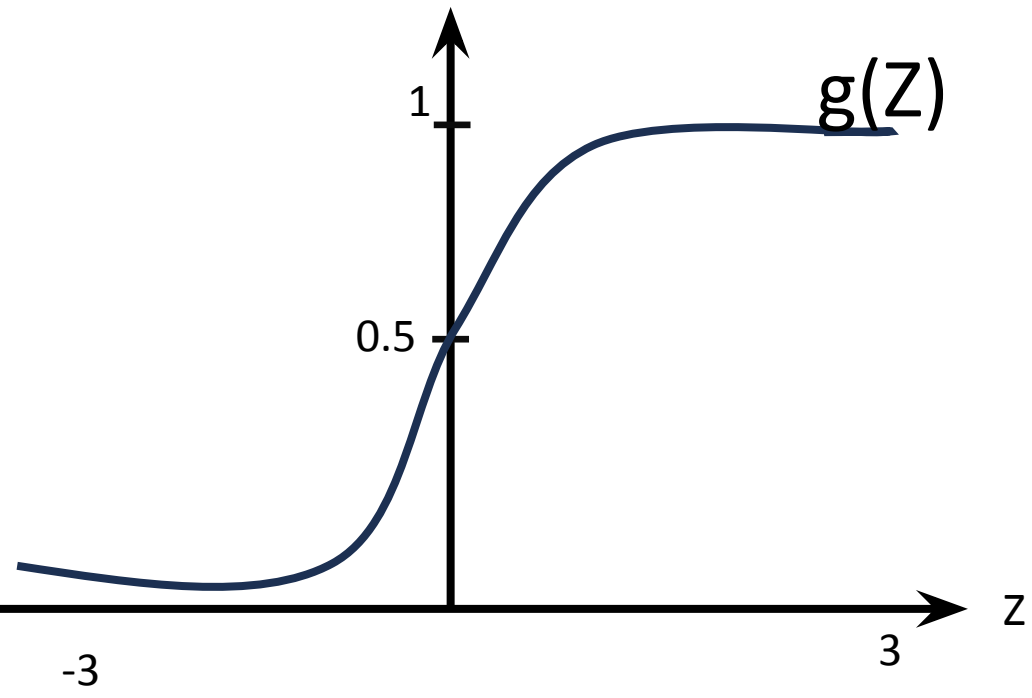


Sigmoid Function outputs between 0 and 1

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

$0<g(z)<1$

Sigmoid Function/Logistic Function



$$g(z) = \frac{1}{1+e^{-z}} \quad 0<g(z)<1$$

Sigmoid Function outputs between 0 and 1

Logistic Regression

$$f_{(\vec{w},b)}(\vec{x})$$

$$z = \vec{w}\vec{x} + b$$

Z

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

Logistic Regression

$$f_{(\vec{w},b)}(\vec{x})$$

$$z = \vec{w}\vec{x} + b$$

z

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

$$f_{(\vec{w},b)}(\vec{x}) = g(\underbrace{\vec{w}\vec{x} + b}_{z}) = \frac{1}{1 + e^{-(\vec{w}\vec{x}+b}}$$

$f_{(\vec{w},b)}(\vec{x})$ outputs value between 0 and 1

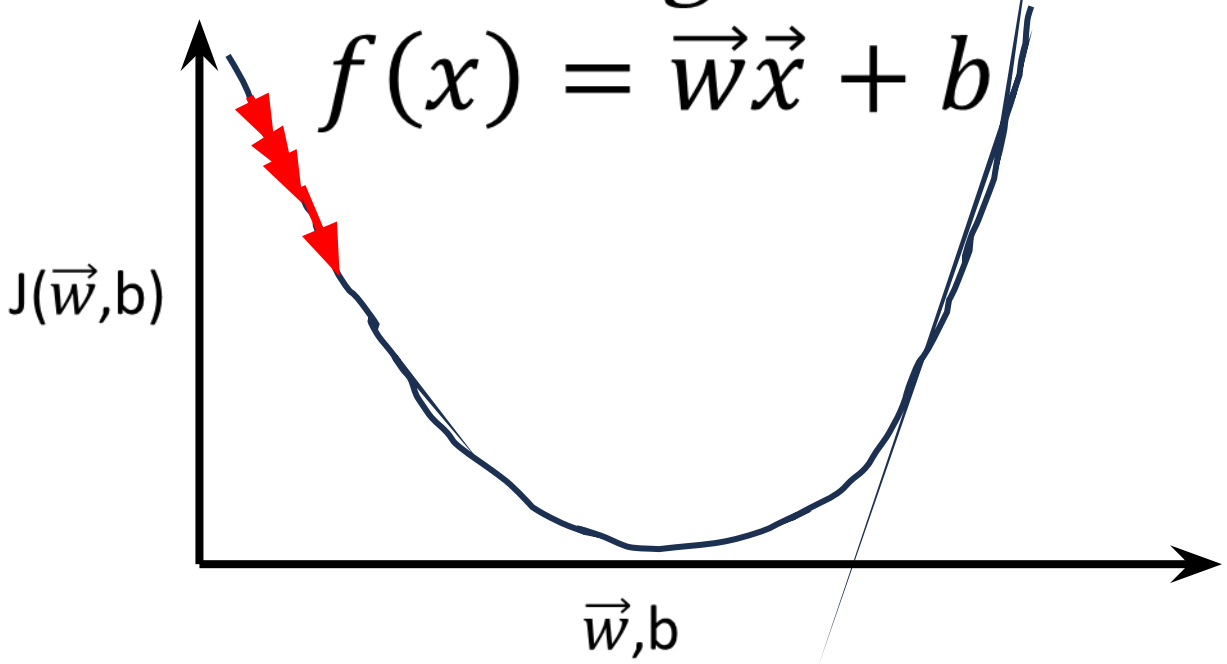
e.g. In Cancer data set
f(x) = 0.7 it means that there are 70% chances that cancer is malignant.

Logistic Regression: Cost Function

In **linear regression** **squared error cost function** was the **convex** function which helps the gradient descend to converge at global minimum.

Linear regression

$$f(x) = \vec{w}\vec{x} + b$$



$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f(x) - y^{(i)}\right)^2$$

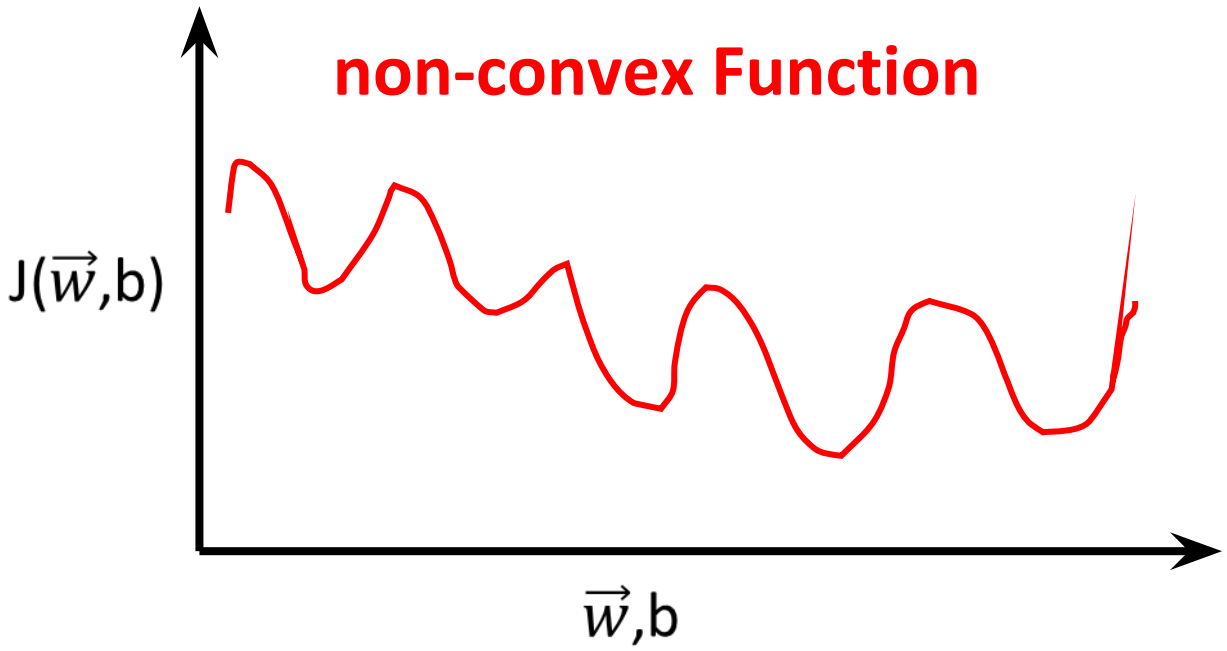
Logistic regression

$$f(x) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-(\vec{w}\vec{x}+b)}}$$

Logistic Regression: Cost Function

In **Logistic regression** **squared error cost function** is **non-convex** function which helps the gradient descend to converge at global minimum.

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f(x) - y^{(i)}\right)^2$$



Logistic regression

$$f(x) = \frac{\mathbf{1}}{\mathbf{1} + \boldsymbol{e}^{-(\vec{w}\vec{x}+b)}}$$

Logistic Regression: Loss Function

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\boxed{\frac{1}{2}\left(f\left(x^{(i)}\right) - y^{(i)}\right)^2} \quad \boxed{L\left(f(x), y^{(i)}\right)}$$

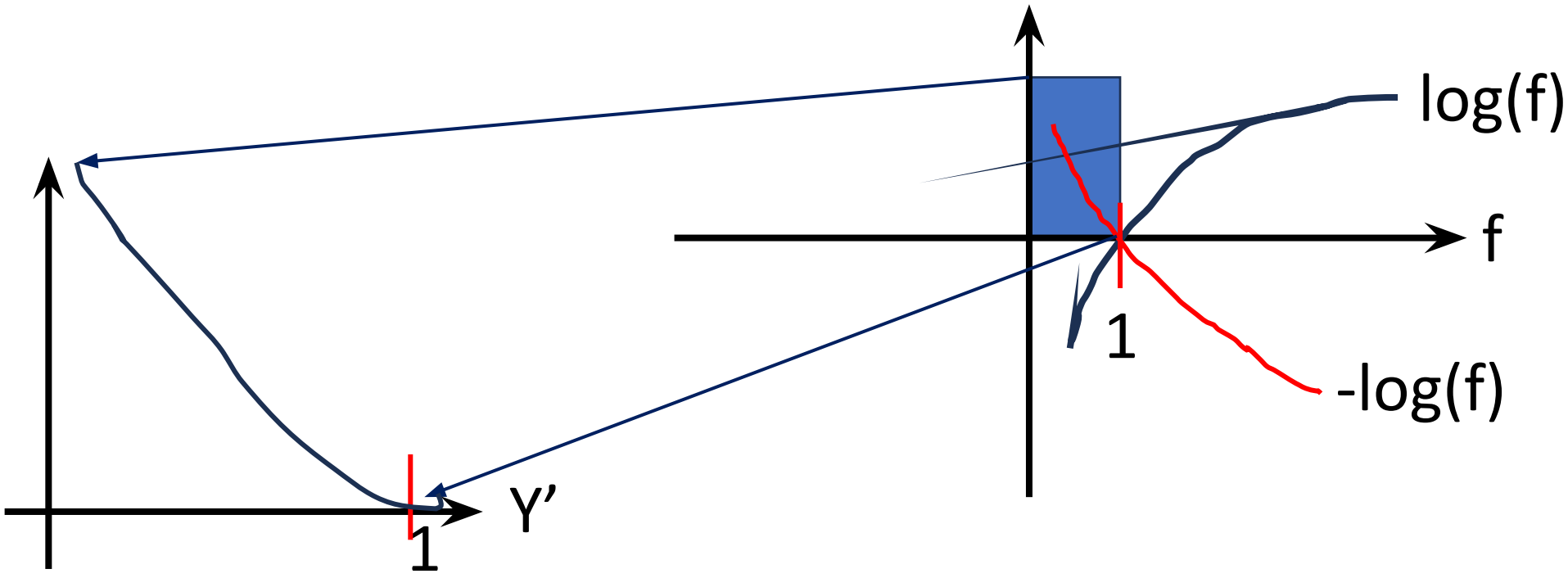
$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(f\left(x^{(i)}\right), y^{(i)}\right)$$

https://arunaddagatla.medium.com/maximum-likelihood-estimation-in-logistic-regression-f86ff1627b67

$$L\left(f\left(x^{(i)}\right), y^{(i)}\right) = \begin{Bmatrix} -\log(f(x))\ if\ y^{(i)} = 1 \\ -\log(1 - f(x))\ if\ y^{(i)} = 0 \end{Bmatrix}$$

Logistic Regression: Loss Function

$$L\left(f\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f\left(x^{(i)}\right)\right) if\ y^{(i)} = 1 \\ -\log\left(1 - f\left(x^{(i)}\right)\right) if\ y^{(i)} = 0 \end{cases}$$



If original label is 1 then loss is near to 0 and vice-versa

Derive the proof for $-\log(1 - f\left(x^{(i)}\right))if\ y^{(i)} = 0$

Can we simplify cost and loss function

$$L\left(f\left(x^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f\left(x^{(i)}\right)\right) & if\ y^{(i)} = 1 \\ -\log\left(1 - f\left(x^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$

$$L\left(f\left(x^{(i)}\right), y^{(i)}\right) = -y^{(i)} \log\left(f\left(x^{(i)}\right)\right) - (1 - y^{(i)}) \log\left(1 - f\left(x^{(i)}\right)\right)$$

$$if\ y^{(i)} = 1$$

$$L\left(f\left(x^{(i)}\right), 1\right) = -\log\left(f\left(x^{(i)}\right)\right)$$

$$if\ y^{(i)} = 0$$

$$L\left(f\left(x^{(i)}\right), 0\right) = -(1 - 0) \log\left(1 - f\left(x^{(i)}\right)\right)$$

$$L\left(f\left(x^{(i)}\right), 0\right) = -\log\left(1 - f\left(x^{(i)}\right)\right)$$

Logistic Regression: Cost Function

$$L\left(f\left(x^{(i)}\right), y^{(i)}\right) = -y^{(i)} \log\left(f\left(x^{(i)}\right)\right) - (1 - y^{(i)}) \log\left(1 - f\left(x^{(i)}\right)\right)$$

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} L\left(f_{\overrightarrow{(w,b)}}\left(x^{(i)}\right), y^{(i)}\right)$$

$$J(\vec{w}, b) = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log\left(f\left(x^{(i)}\right)\right) + (1 - y^{(i)}) \log\left(1 - f\left(x^{(i)}\right)\right)$$

Logistic Regression using sklearn

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

data = load_breast_cancer()
df=pd.DataFrame(data.data,columns=data.feature_names)
X_train, X_test, y_train, y_test = train_test_split(data.data,data.target, test_size=0.33, random_state=42)
LR=LogisticRegression(max_iter=10000)
LR.fit(X_train,y_train)
y_predict = LR.predict(X_test)
CM=metrics.confusion_matrix(y_test,y_predict)
print(CM)
plt.figure(figsize=(6,6))
sns.heatmap(CM,annot=True)
report = classification_report(y_test, y_predict)
print(report)
```

Logistic Regression

Model $$f(x) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-(\vec{w}\vec{x}+b)}}$$

Cost Function

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\left(f\left(x^{(i)}\right)\right) + (1-y^{(i)})\log\left(1-f\left(x^{(i)}\right)\right)$$

Cost Function

$$J(\vec{w},b) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\left(y'^{(i)}\right) + (1-y^{(i)})\log\left(1-y'^{(i)}\right)$$

Gradient Descent:

Repeat until converge{

$$w_i = w_i - \alpha\frac{\partial}{\partial w_i}J(\vec{w},b) \quad \text{for i=1....n}$$

$$\text{b} = b - \alpha\frac{\partial}{\partial b}J(\vec{w},b)$$

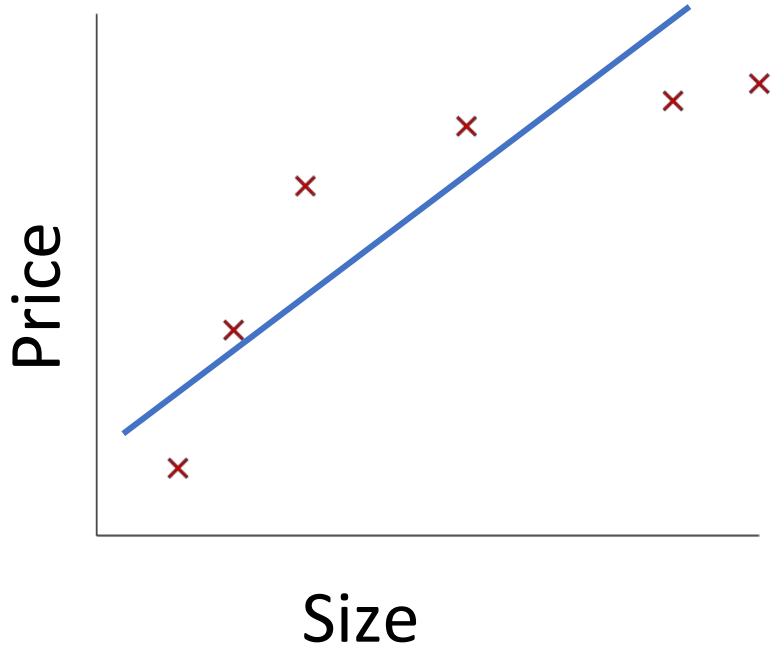
}

Some imp. Terminology

- Underfitting
- Overfitting
- Bias
- Variance

Bias: **Bias refers to the error due to overly simplistic assumptions in the learning algorithm.** These assumptions make the model easier to comprehend and learn but might not capture the underlying complexities of the data. It is the error due to the model's inability to represent the true relationship between input and output accurately. **When a model has poor performance both on the training and testing data means high bias because of the simple model.**

Variance: **Variance, on the other hand, is the error due to the model's sensitivity to fluctuations in the training data.** High variance occurs when a model learns the training data's noise and random fluctuations rather than the underlying pattern.

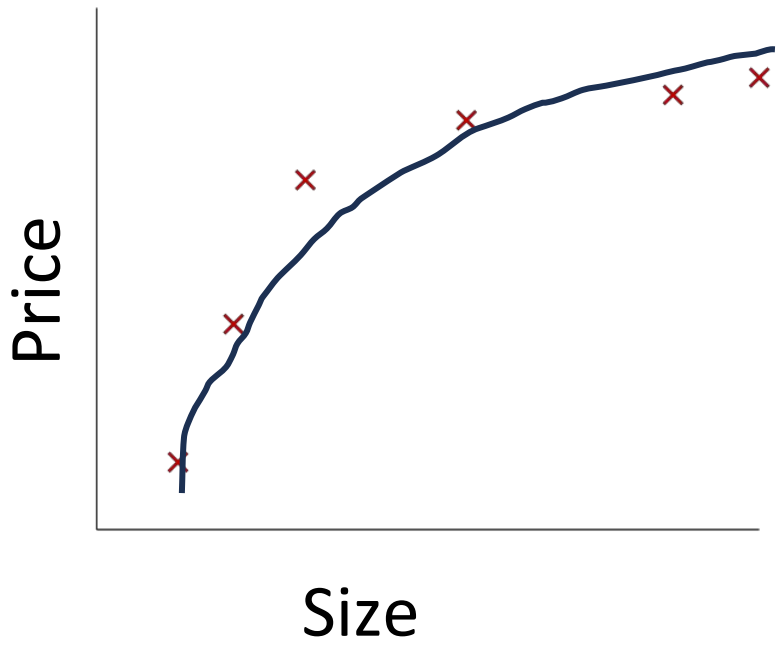


$$w_1 x_1 + b$$

High Bias
Low variance

Underfitting: machine learning algorithm is said to have underfitting when a model is too simple to capture data complexities. It represents the inability of the model to learn the training data effectively result in poor performance both on the training and testing data.

The underfitting model has High bias and low variance



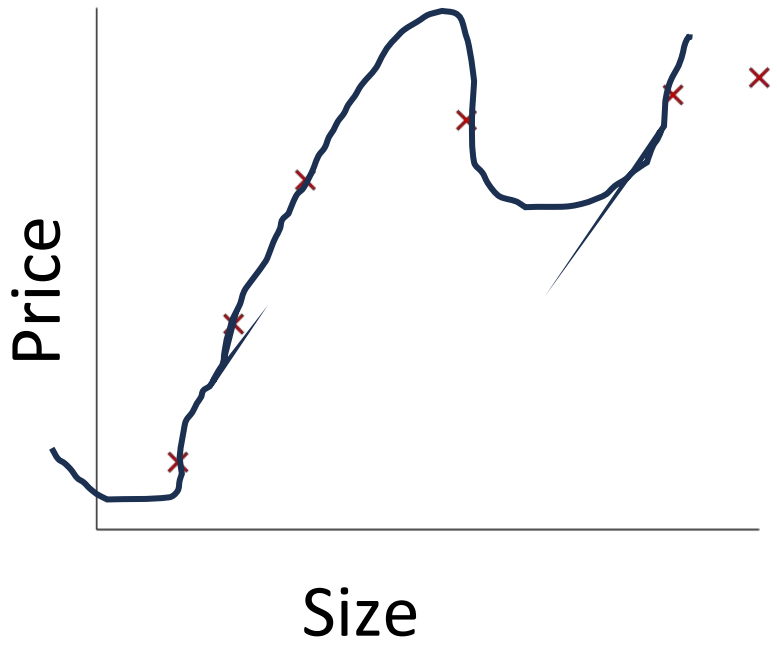
$$w_1 x_1 + w_2 x^2{}_1 + b$$

Low Bias

Low variance

Balanced fit: machine learning algorithm is said to have balanced fit when a model has marginally capture data complexities.

The Balanced fit model has low bias and low variance



$$w_1 x_1 + w_2 x^2{}_1 + w_3 x^3{}_1 + \cdots + b$$

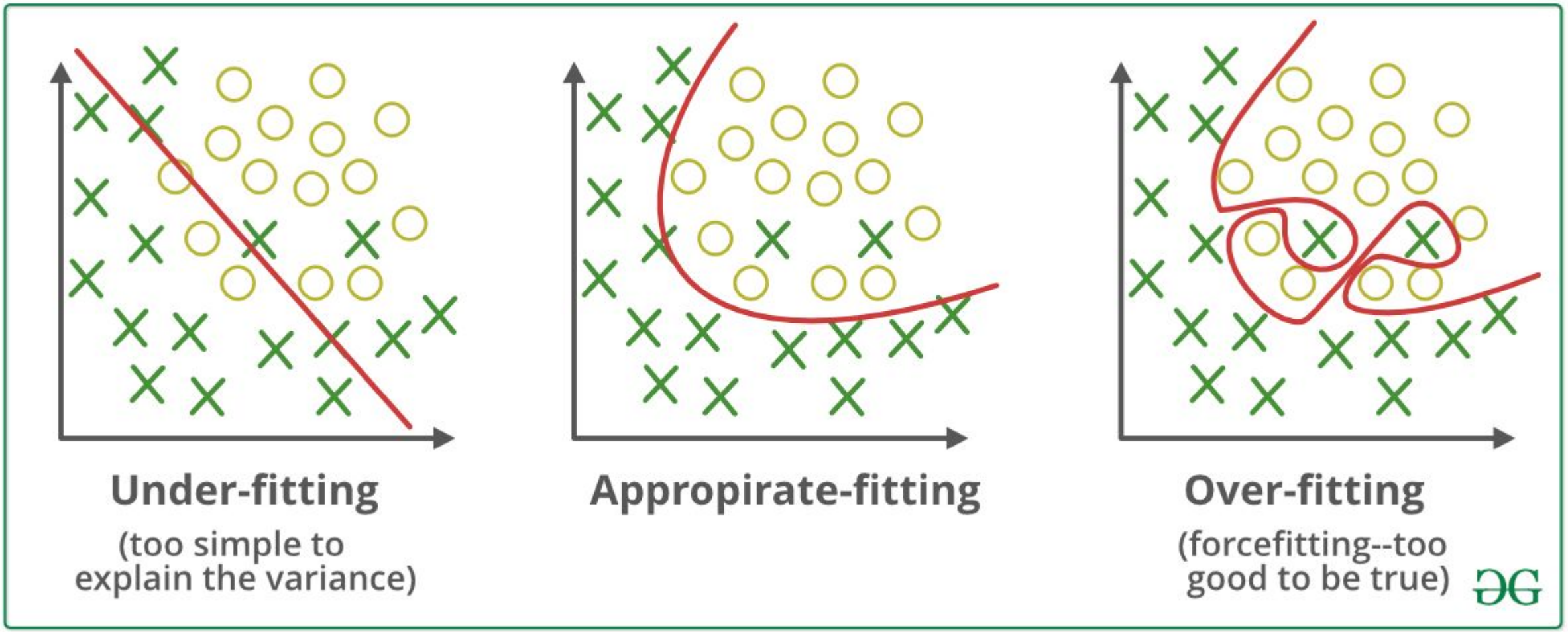
Low Bias

High variance

Overfitting is a problem where the evaluation of machine learning algorithms on training data is different from unseen data.

The Overfitting has low bias and High variance

Solution of Overfitting and underfitting is Regularization



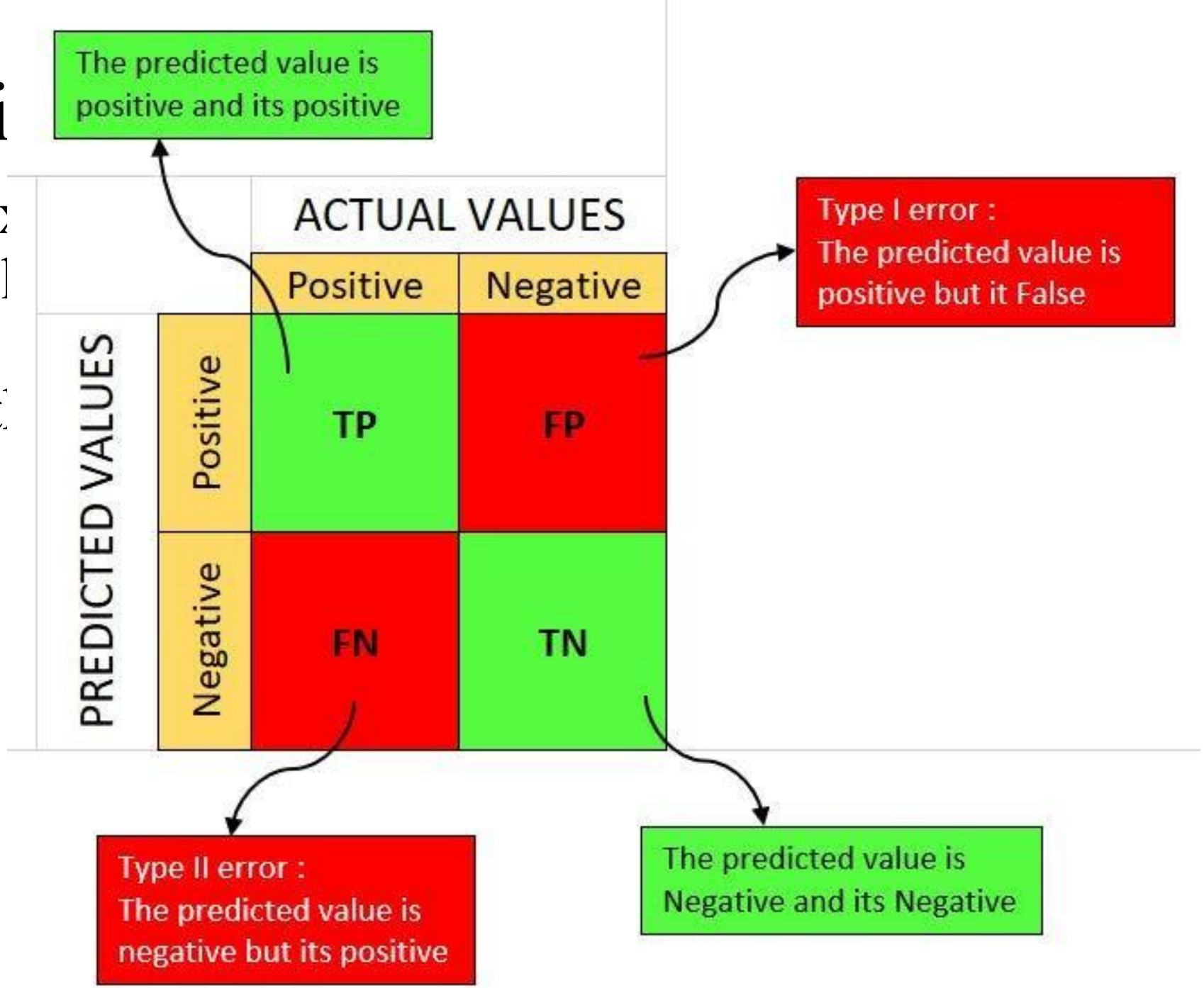
Source: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Performance Measures/Metrics

- Confusion Matrix
- Accuracy
- Precision
- Recall (TPR, Sensitivity)
- F-score

Confusion Matri

- A Confusion matri
  performance of a cl
  target classes. The
  those predicted by t

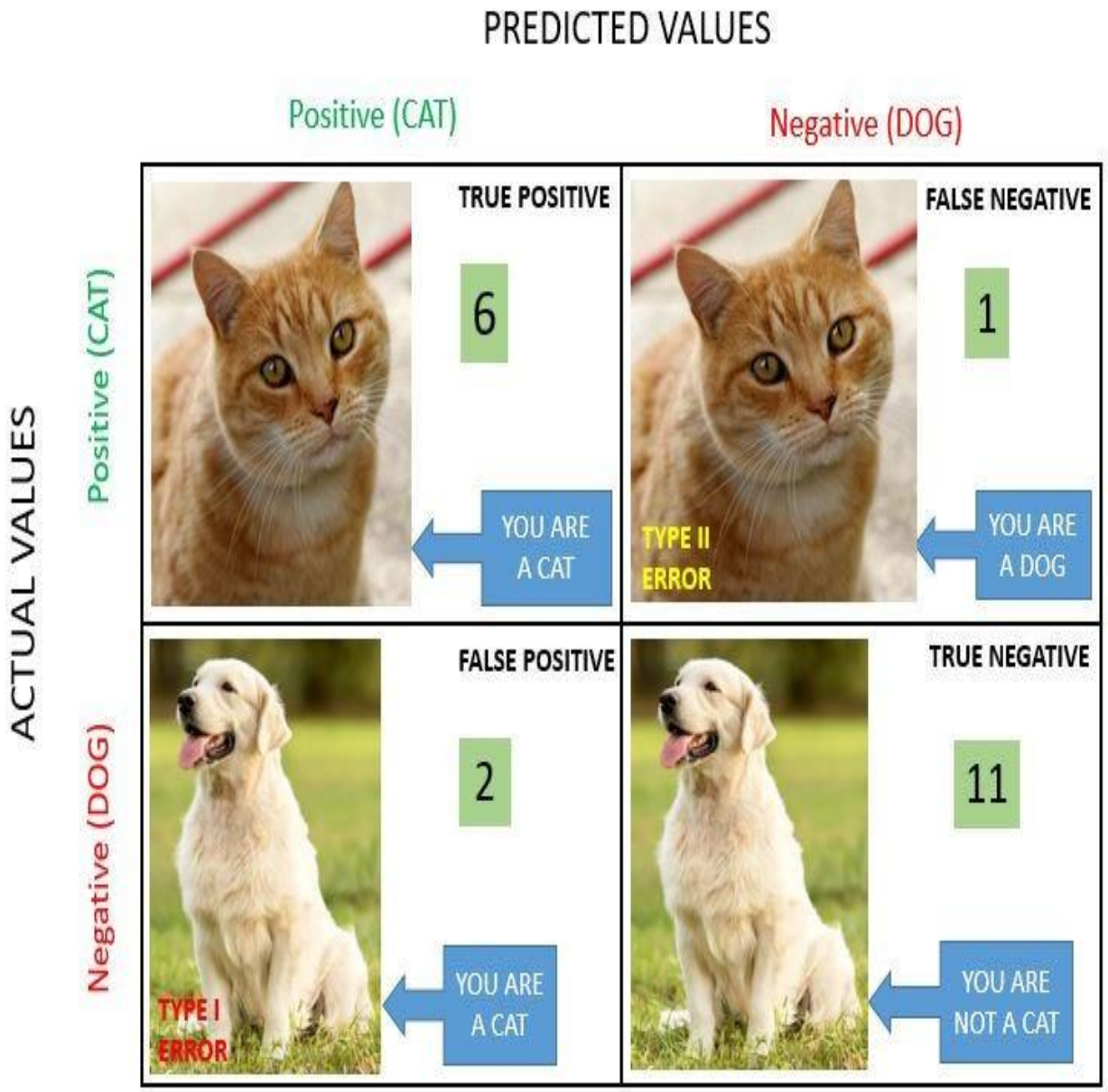


1. A **good model** is one which has ***high TP and TN rates***, while ***low FP and FN rates***.
2. If you have an ***imbalanced dataset*** to work with, it's always better to use ***confusion matrix*** as your evaluation criteria for your machine learning model.

Understanding Confusion Matrix

Actual values = [‘dog’, ‘cat’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’]

Predicted values = [‘dog’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘cat’, ‘cat’, ‘cat’, ‘dog’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’, ‘dog’, ‘dog’, ‘cat’]



Classification Measure

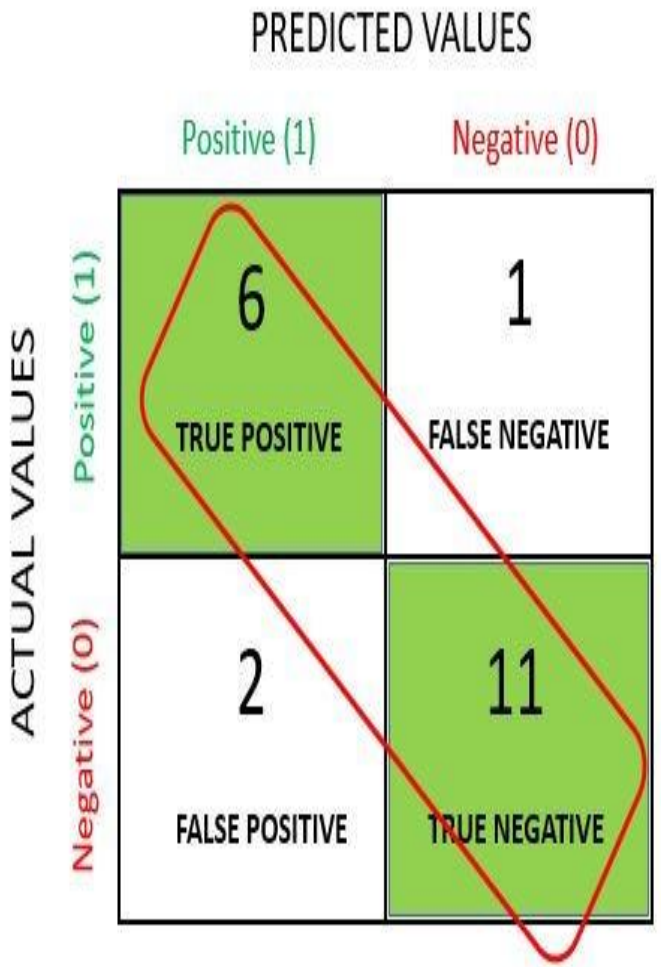
Basically, it is an extended version of the confusion matrix. There are measures other than the confusion matrix which can help achieve better understanding and analysis of our model and its performance.

- Accuracy
- Precision
- Recall (TPR, Sensitivity)
- F-score

Accuracy simply measures how often the classifier makes the **correct prediction**. It's the ratio between the number of correct predictions and the total number of predictions.

It is a measure of **correctness** that is achieved in **true prediction**. In simple words, it tells us how many predictions are ***actually positive*** out of all the ***total positive predicted***.

Accuracy is a valid choice of ***evaluation for classification problems*** which are ***well balanced*** and ***not skewed or there is***



Accuracy

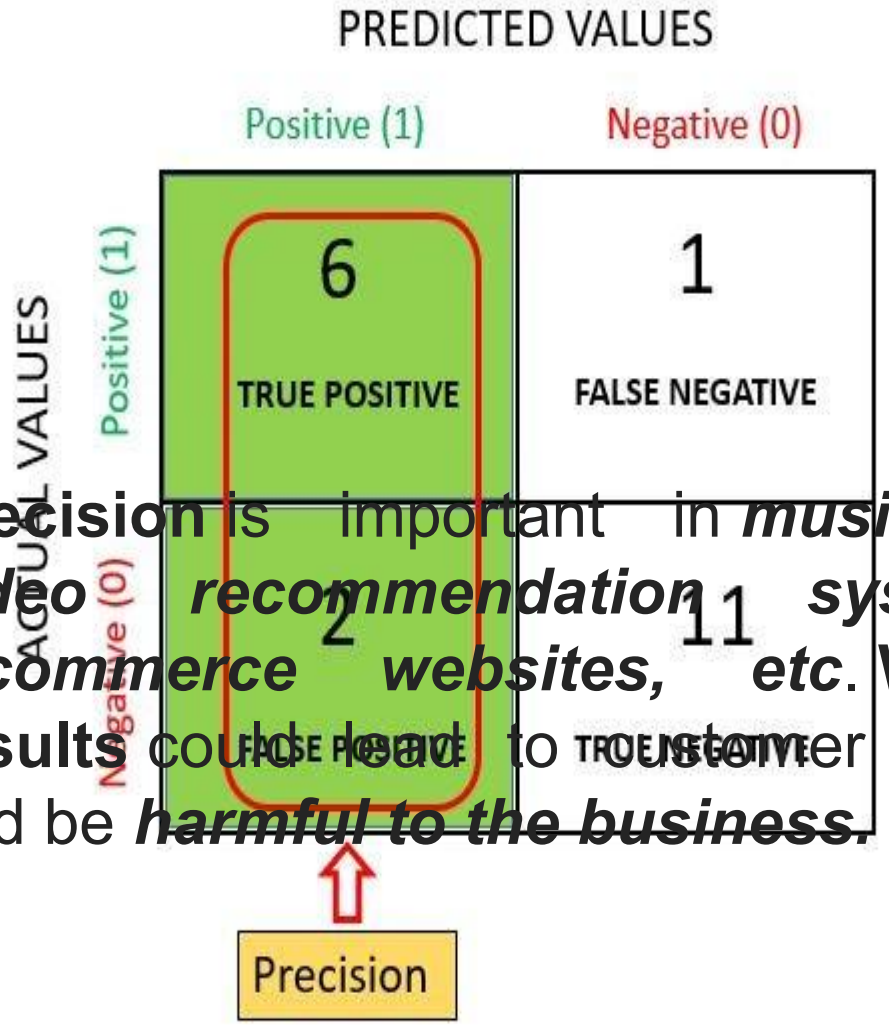
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{6 + 11}{6 + 11 + 2 + 1} = 85\%$$

Precision:

It is a measure of **correctness** that is achieved in **true prediction**. In simple words, it tells us how many predictions are ***actually positive*** out of all the ***total positive predicted***.

Precision is defined as the **ratio** of the total number of ***correctly classified positive classes*** divided by the ***total number of predicted positive classes***.

*"**Precision** is a useful metric in cases where **False Positive** is a higher concern than **False Negatives**"*



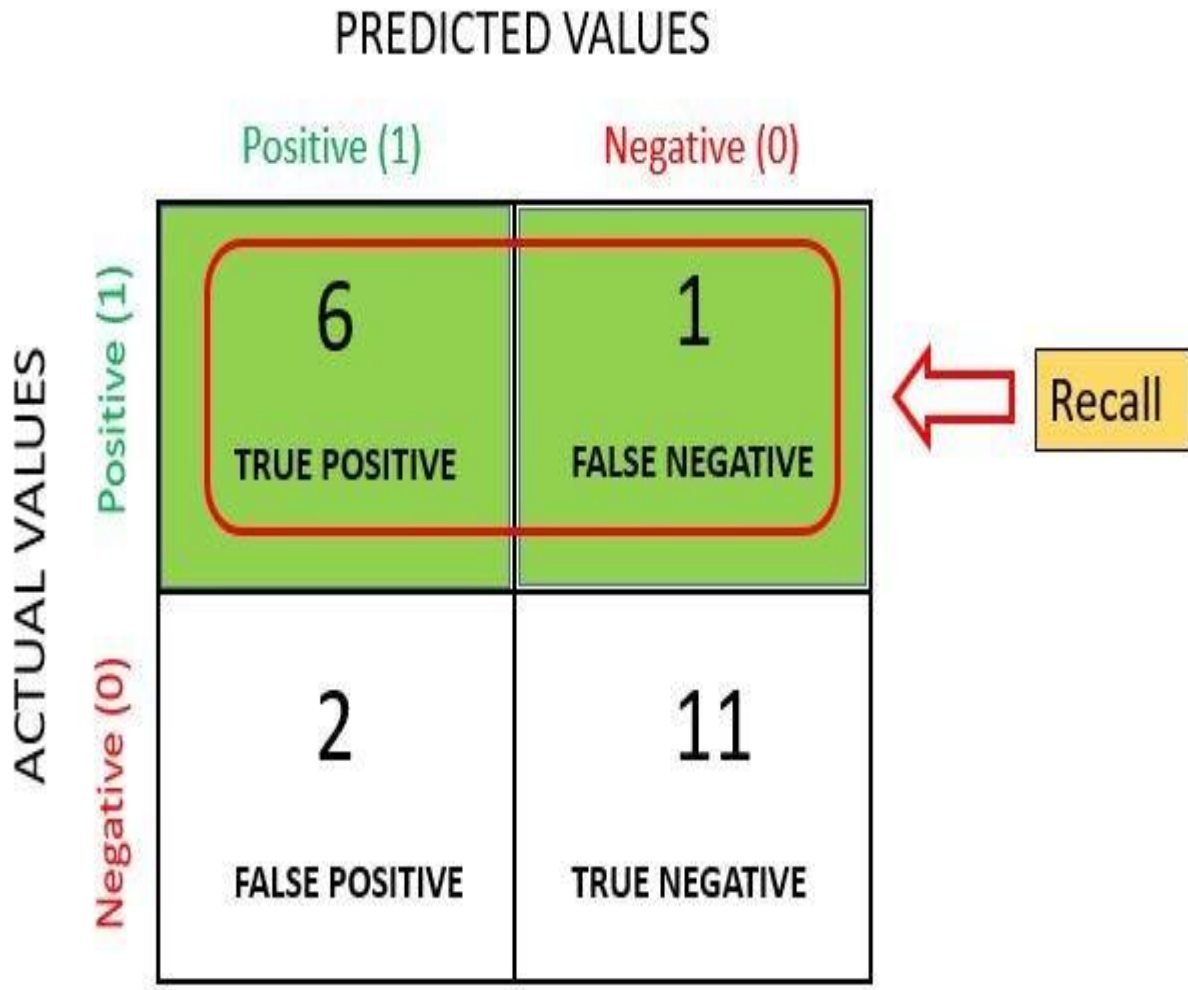
Precision is important in ***music or video recommendation systems, e-commerce websites, etc***. **Wrong results** could lead to customer churn and be ***harmful to the business.***

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Predictions Actually Positive}}{\text{Total Predicted positive}} = \frac{6}{6+2} = 0.75$$

Recall

It is a measure of **actu observations** which are predicted **correctly**, i.e. how mar observations of positive class are actua predicted as positive. It is also know as **Sensitivity**. ***Recall*** is a valid choi of evaluation metric when we want capture ***as many positives*** possible.

Recall is defined as the ratio of t total number of *correctly classifie positive classes* divide by the *tot number of positive classes*.



$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Predictions Actually Positive}}{\text{Total Actual positive}} = \frac{6}{6+1} = 0.85$$

F-measure / F1-Score

The **F1 score** is a number between **0 and 1** and is the ***harmonic mean of precision and recall***. We use harmonic mean because it is not sensitive to extremely large values unlike simple averages.

F1 score sort of maintains a **balance** between the ***precision and recall*** for your classifier. If your ***precision is low***, the ***F1 is low*** and if the ***recall is low*** again your ***F1 score is low***.

$$\text{F1-Score} = 2 * \frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} = 2 * \frac{(0.85 * 0.75)}{(0.85 + 0.75)} = 0.79$$

Some Important Questions

Is it necessary to check for recall (or) precision if you already have a high accuracy?

We can not rely on a single value of accuracy in classification when the classes are imbalanced. For example, we have a dataset of 100 patients in which 5 have diabetes and 95 are healthy. However, if our model only predicts the majority class i.e. all 100 people are healthy even though we have a classification accuracy of 95%.

Some Important Questions

When to use Accuracy / Precision / Recall / F1-Score?

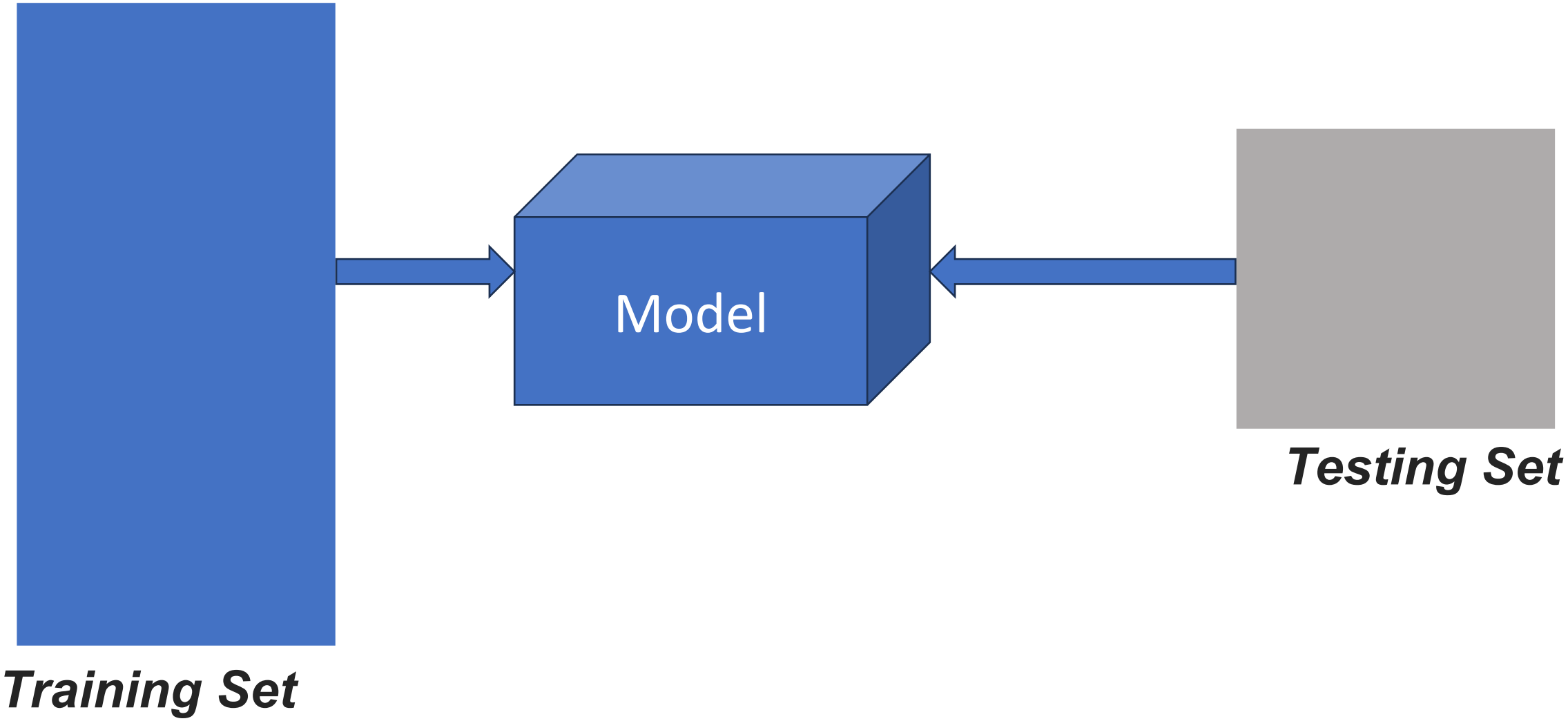
a. ***Accuracy*** is used when the ***True Positives and True Negatives*** are more important. ***Accuracy*** is a better metric for ***Balanced Data.***

b. Whenever **False Positive** is much more important use ***Precision.***

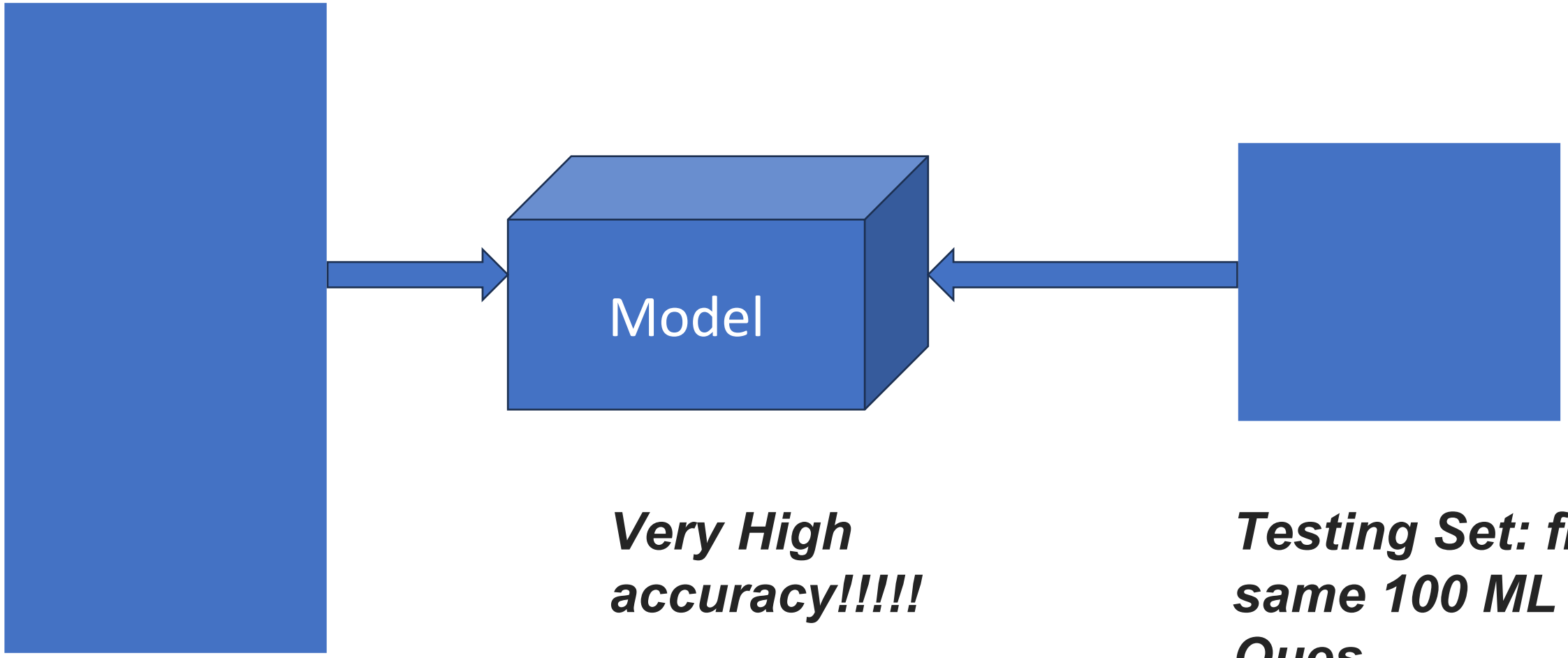
c. Whenever **False Negative** is much more important use ***Recall.***

d. ***F1-Score*** is used when the ***False Negatives and False Positives*** are important. ***F1-Score*** is a better metric for ***Imbalanced Data.***

Cross Validation



Option 1: Use all data in both training and testing

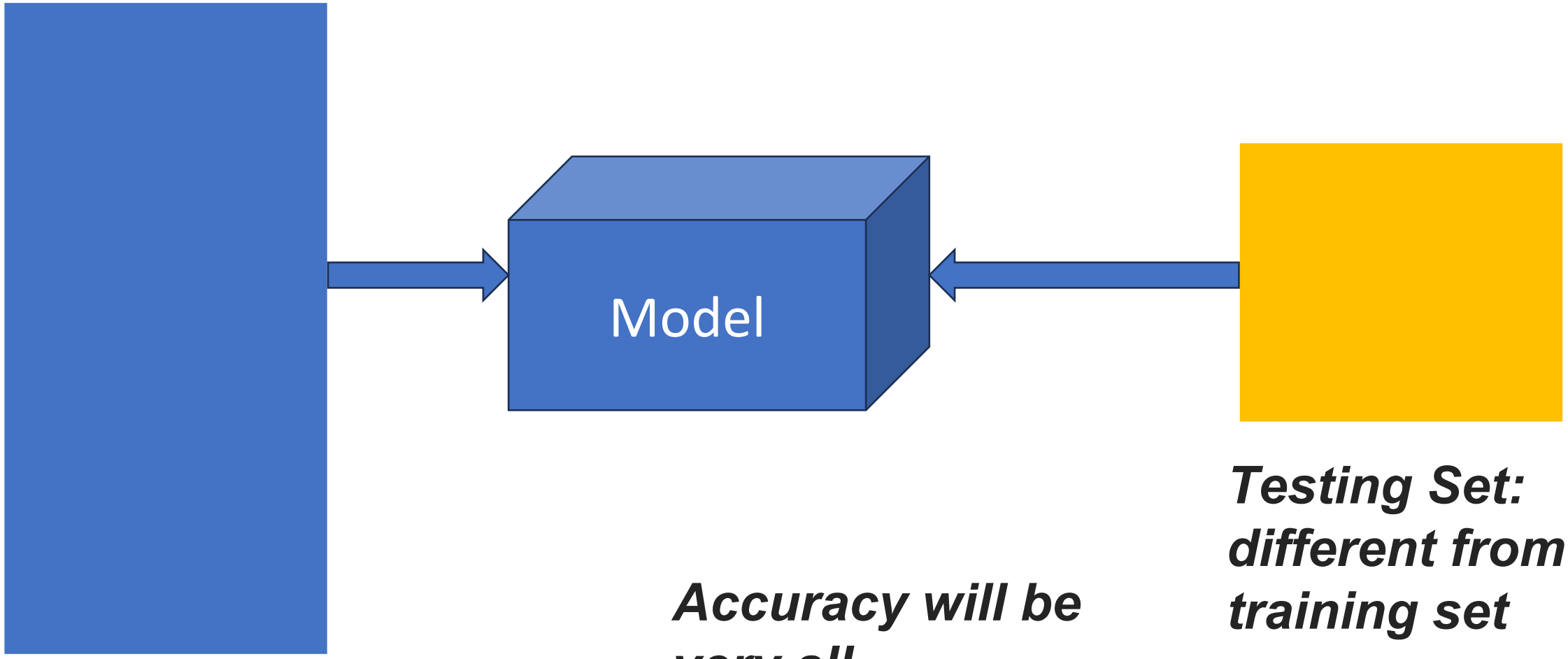


Very High accuracy!!!!!

Testing Set: from same 100 ML Ques.

Training Set: 100 ML Qeus

Option 2: Different data in training and testing



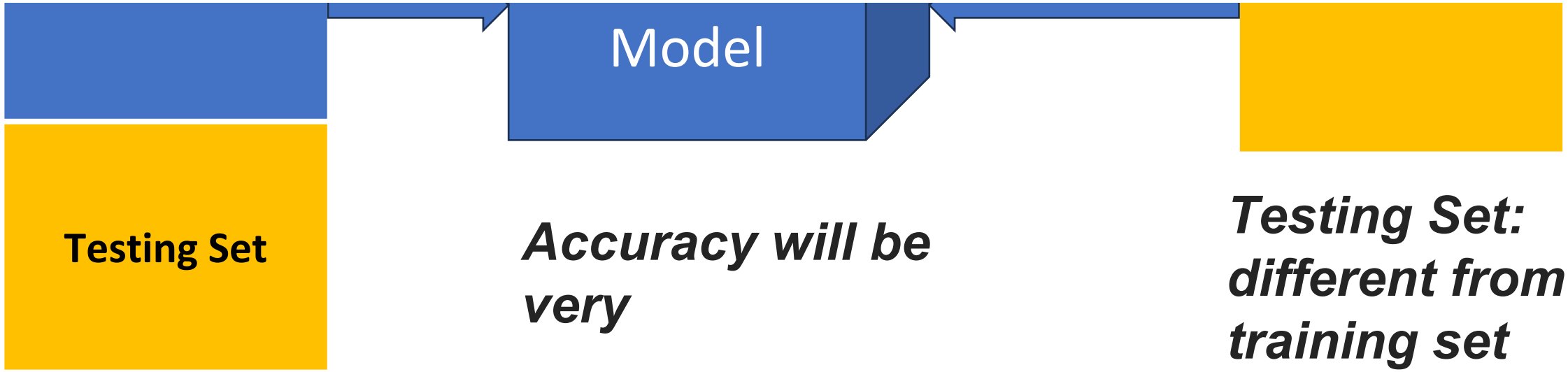
Training Set: 100 ML Qeus

Accuracy will be very all

Testing Set: different from training set

Option 3: Split data into training and testing/Handout Method

```
from sklearn.model_selection import train_test_split
X_train,X_test, y_train,y_test = train_test_split(data.data,
                                                  data.target,test_size=0.30)
```

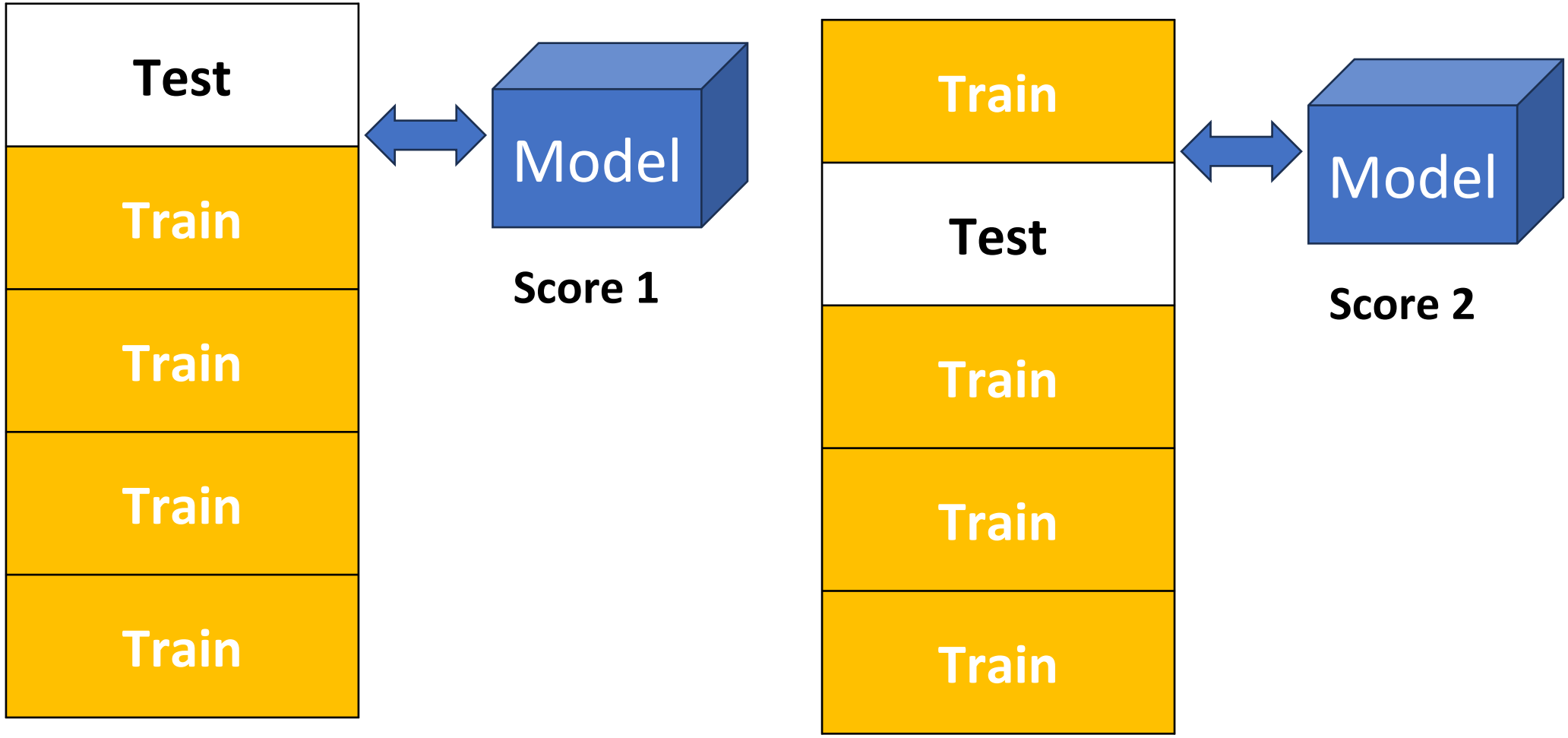


Accuracy will be very

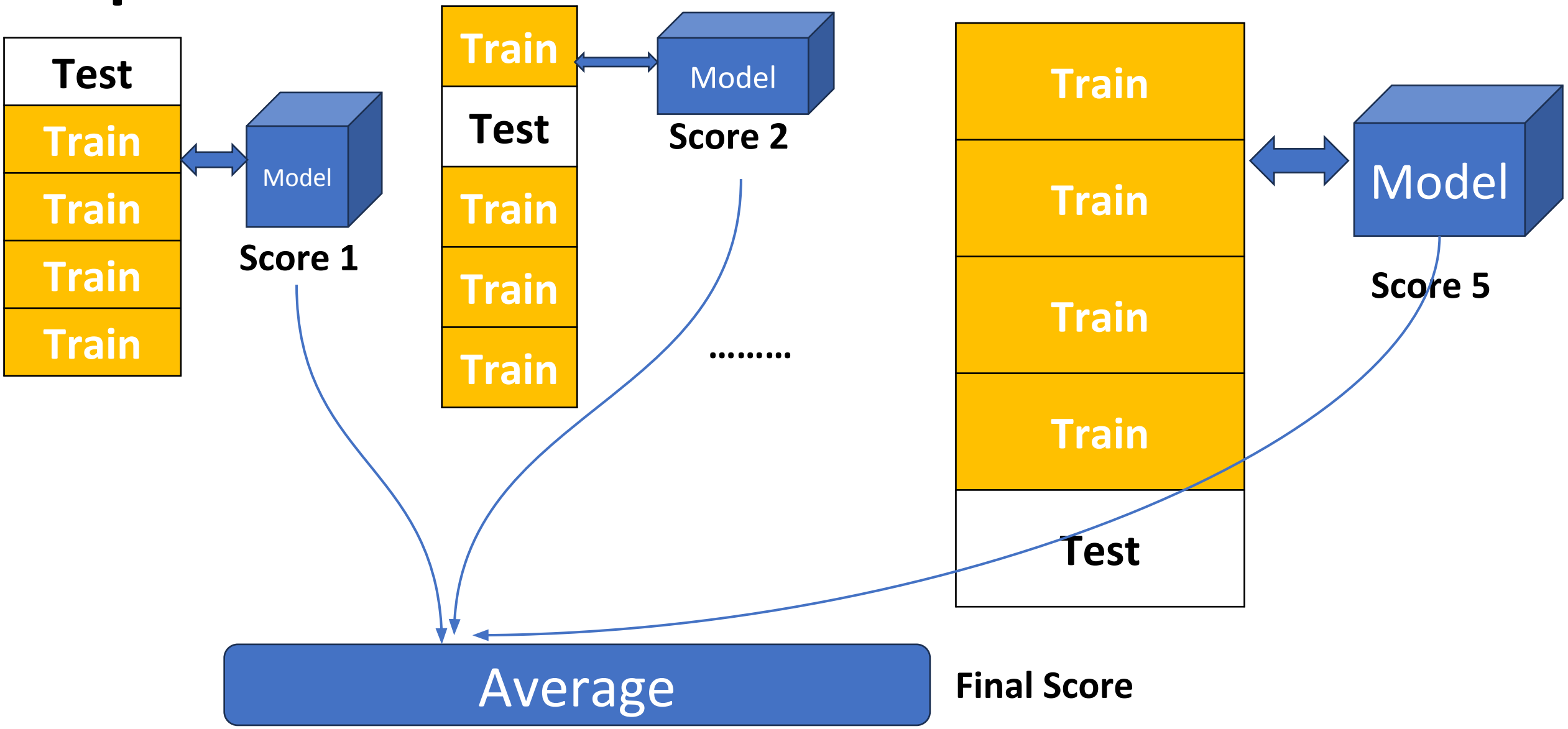
Testing Set: different from training set

Training Set: 70 ML Qeus

Option 4: K fold cross validation



Option 4: K fold cross validation



What is K-Fold Cross Validation

K-Fold CV is where a given data set is split into a K number of sections/folds where each fold is used as a testing set at some point. Lets take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.

```python
from sklearn.model_selection import KFold
import numpy as np

x = np.array([1,2,3,4,5,6,7,8,9])

kf = KFold(n_splits=3)
for x_index, y_index in kf.split(x):
    print(x_index,y_index)
```

```
[3 4 5 6 7 8] [0 1 2]
[0 1 2 6 7 8] [3 4 5]
[0 1 2 3 4 5] [6 7 8]
```

```python
from sklearn.preprocessing import StandardScaler
kfold_cancer = KFold(n_splits=10,shuffle=True,random_state=42)
score=[]
ss = StandardScaler()
data.data=ss.fit_transform(data.data)
lr_cancer = LogisticRegression(max_iter=100)
for train_index, test_index in kfold_cancer.split(data.data):
    X_train,y_train,X_test,y_test=data.data[train_index],data.target[train_index],data.data[test_index],data.target[test_index]
    lr_cancer.fit(X_train,y_train)
    score.append(lr_cancer.score(X_test,y_test))

print(score)
print("Avarege score is ",np.mean(score))
```

```
[0.9824561403508771, 0.9824561403508771, 0.9824561403508771, 0.9824561403508771, 0.9824561403508771, 0.9298245614035088, 1.0,
0.9824561403508771, 0.9473684210526315, 0.9821428571428571]
Avarege score is  0.975407268170426
```

What is Stratified K-Fold Cross Validation?

Stratified k-fold cross-validation is the same as just k-fold cross-validation, But Stratified k-fold cross-validation, it does stratified sampling instead of random sampling.

Stratified Sampling:

In stratified sampling, The training_set consists of 64 negative class{0} (80% of 80) and 16 positive class {1} (80% of 20) i.e. 64{0}+16{1}=80 samples in training_set which represents the original dataset in equal proportion and similarly test_set consists of 16 negative class {0} (20% of 80) and 4 positive class{1} (20% of 20) i.e. 16{0}+4{1}=20 samples in test_set which also represents the entire dataset in equal proportion. This type of train-test-split results in good accuracy.

Leave-one-out cross-validation (LOOCV) is an extreme case of k-fold cross-validation using **one record or data instance** at a time as a test data. This is done to maximize the count of data used to train the model. It is obvious that the number of iterations for which it has to be run is equal to the total number of data in the input data set. Hence, obviously, it is computationally **very expensive** and not used much in practice.

Bootstrapping randomly picks data instances from the input data set, with the possibility of the same data instance to be picked multiple times. This essentially means that from the input data set having ‘n’ data instances, bootstrapping can create one or more training data sets having ‘n’ data instances, some of the data instances being repeated multiple times

Bootstrap sampling or simply bootstrapping is a popular way to identify training and test data sets from the input data set. It uses the technique of **Simple Random Sampling with Replacement (SRSWR),** which is a well-known technique in sampling theory for drawing random samples.