Introduction to Machine Learning

Measures of Accuracy

Andres Mendez-Vazquez

February 9, 2023

Outline

Outline

Introduction

What did we see until now?

The design of learning machines from two main points:

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \,|i = 1, 2, ..., N\} \tag{1}$$

Introduction

What did we see until now?

The design of learning machines from two main points:

- Statistical Point of View
- Linear Algebra and Optimization Point of View

Going back to the probability models

We might think in the machine to be learned as a function $g(\boldsymbol{x}|\mathcal{D})$....

- Something as curve fitting...

Under a data set

$$\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \,|i = 1, 2, ..., N\} \tag{1}$$

Remark: Where the $\boldsymbol{x}_i \sim p(\boldsymbol{x}|\Theta)$!!!

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- For example, $E[y|\boldsymbol{x}]$ the optimal regression...

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- For example, $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- For example, $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Why?

The approximation may be very good for a specific training data set but very bad for another.

Thus, we have that

Two main functions

- A function $g(\boldsymbol{x}|\mathcal{D})$ obtained using some algorithm!!!
- For example, $E[y|\boldsymbol{x}]$ the optimal regression...

Important

The key factor here is the dependence of the approximation on $\mathcal{D}$.

Why?

The approximation may be very good for a specific training data set but very bad for another.

- This is the reason of studying fusion of information at decision level...

Outline

How do we measure the difference?

We can use the variance

$$Var(X) = E((X - \mu)^2)$$

How do we measure the difference?

We can use the variance

$$Var(X) = E((X - \mu)^2)$$

We can do that for our data

$$Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) = E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[y|\boldsymbol{x}\right]\right)^2\right)$$

How do we measure the difference?

We can use the variance

$$Var(X) = E((X - \mu)^2)$$

We can do that for our data

$$Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) = E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[y|\boldsymbol{x}\right]\right)^2\right)$$

Now, if we add and subtract

$$E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] \tag{2}$$

How do we measure the difference?

We can use the variance

$$Var(X) = E((X - \mu)^2)$$

We can do that for our data

$$Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) = E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[y|\boldsymbol{x}\right]\right)^2\right)$$

Now, if we add and subtract

$$E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] \tag{2}$$

Remark: The expected output of the machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$

Thus, we have that

Or Original variance

$$
\begin{aligned}
Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) &= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[y|\boldsymbol{x}\right]\right)^2\right) \\
&= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] + E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)^2\right) \\
&= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2 + ... \right. \\
&...2\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\right)\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right) + ... \\
&... \left.\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)^2\right)
\end{aligned}
$$

Thus, we have that

Or Original variance

$$\begin{aligned} Var_{\mathcal{D}}\left(g\left(\boldsymbol{x}|\mathcal{D}\right)\right) &= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E\left[y|\boldsymbol{x}\right]\right)^2\right) \\ &= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] + E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)^2\right) \\ &= E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2 + ... \\ &...2\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\right)\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right) + ... \\ &... \left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)^2\Big) \end{aligned}$$

Finally

$$E_D\left(\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)\right)\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)\right) =? \quad (3)$$

Outline

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

Where the variance

It represents the measure of the error between our machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$ and the expected output of the machine under $\boldsymbol{x}_i \sim p\left(\boldsymbol{x}|\Theta\right)$.

We have the Bias-Variance

Our Final Equation

$$E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E\left[y|\boldsymbol{x}\right]\right)^2\right)=\underbrace{E_D\left(\left(g\left(\boldsymbol{x}|\mathcal{D}\right)-E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right)}_{VARIANCE}+\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right]-E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS}$$

Where the variance

It represents the measure of the error between our machine $g\left(\boldsymbol{x}|\mathcal{D}\right)$ and the expected output of the machine under $\boldsymbol{x}_i\sim p\left(\boldsymbol{x}|\Theta\right)$.

Where the bias

It represents the quadratic error between the expected output of the machine under $\boldsymbol{x}_i\sim p\left(\boldsymbol{x}|\Theta\right)$ and the expected output of the optimal regression.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

The situation is more dire in a finite set of data $\mathcal{D}$

We have then a trade-off:

1. Increasing the bias decreases the variance and vice versa.

Remarks

We have then

Even if the estimator is unbiased, it can still result in a large mean square error due to a large variance term.

The situation is more dire in a finite set of data $\mathcal{D}$

We have then a trade-off:

1. Increasing the bias decreases the variance and vice versa.
2. This is known as the **bias–variance dilemma**.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

Furthermore

If N grows we can have a more complex model to be fitted which reduces bias and ensures low variance.

Similar to...

Curve Fitting

If, for example, the adopted model is complex (many parameters involved) with respect to the number N, the model will fit the idiosyncrasies of the specific data set.

Thus

Thus, it will result in low bias but will yield high variance, as we change from one data set to another data set.

Furthermore

If N grows we can have a more complex model to be fitted which reduces bias and ensures low variance.

- However, N is always finite!!!

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Allowing you to impose restrictions

Lowering the bias and the variance

Thus

You always need to compromise

However, you always have some a priori knowledge about the data

Allowing you to impose restrictions

Lowering the bias and the variance

Nevertheless

We have the following example to grasp better the bothersome **bias–variance dilemma**.

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

We know that

The optimum regressor is $E[y|x] = f(x)$

For this

Assume

The data is generated by the following function

$$y = f(x) + \epsilon,$$
$$\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$$

We know that

The optimum regressor is $E[y|x] = f(x)$

Furthermore

Assume that the randomness in the different training sets, $\mathcal{D}$, is due to the y_i's (Affected by noise), while the respective points, x_i, are fixed.

Outline

Sampling the Space

Imagine that $\mathcal{D} \subset [x_1, x_2]$ in which x lies

For example, you can choose $x_i = x_1 + \frac{x_2 - x_1}{N-1}(i-1)$ with $i = 1, 2, ..., N$

Case 1

Choose the estimate of $f(x)$, $g(x|\mathcal{D})$, to be independent of $\mathcal{D}$

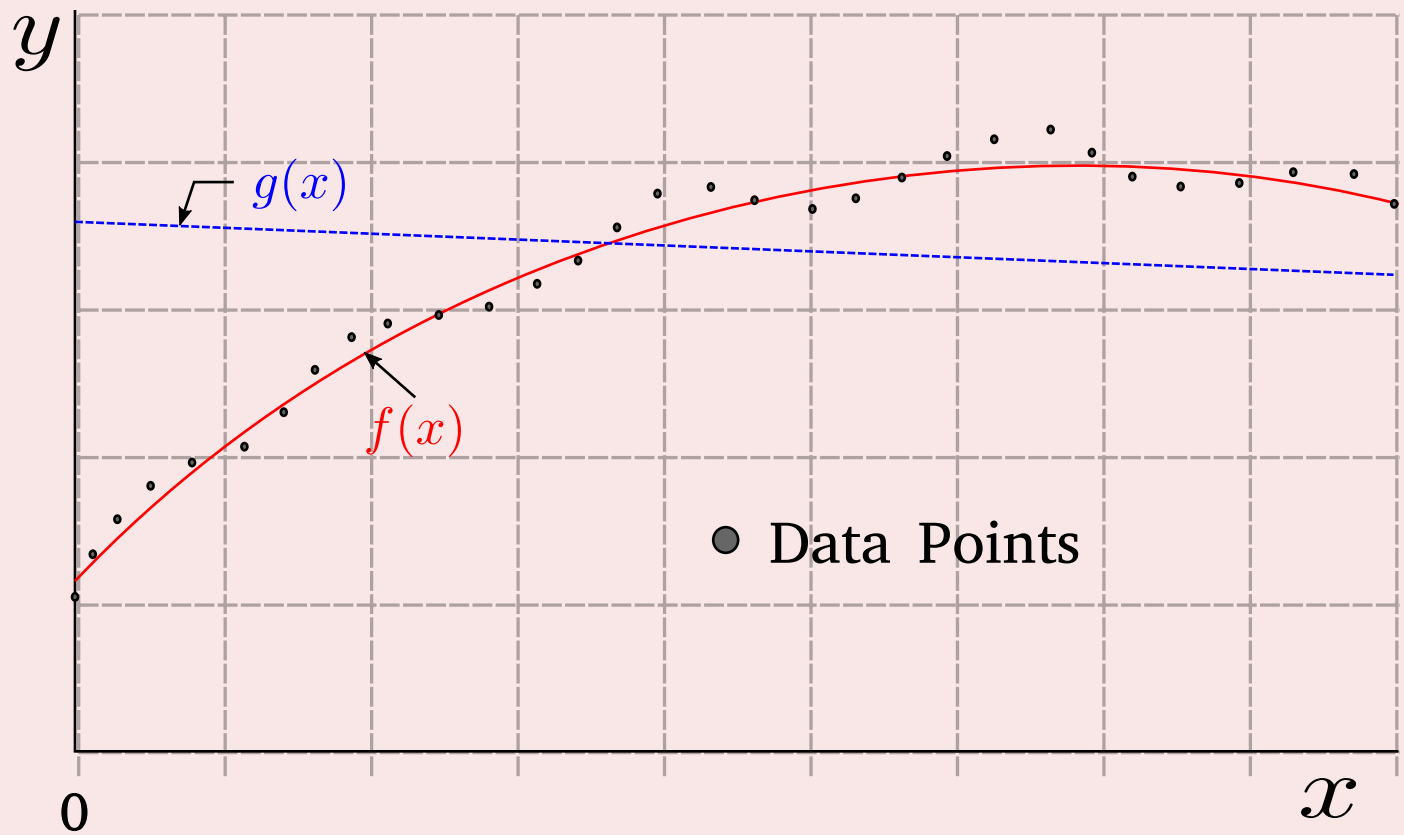
For example, $g(x) = w_1 x + w_0$

Case 1

Choose the estimate of $f(x)$, $g(x|\mathcal{D})$, to be independent of $\mathcal{D}$

For example, $g(x) = w_1 x + w_0$

For example, the points are spread around $(x, f(x))$



Case 1

Since $g(x)$ is fixed

$$E_{\mathcal{D}}\left[g\left(x|\mathcal{D}\right)\right] = g\left(x|\mathcal{D}\right) \equiv g\left(x\right) \tag{4}$$

Case 1

Since $g(x)$ is fixed

$$E_{\mathcal{D}}\left[g\left(x|\mathcal{D}\right)\right] = g\left(x|\mathcal{D}\right) \equiv g\left(x\right) \tag{4}$$

With

$$Var_{\mathcal{D}}\left[g\left(x|\mathcal{D}\right)\right] = 0 \tag{5}$$

Case 1

Since $g(x)$ is fixed

$$E_{\mathcal{D}}\left[g\left(x|\mathcal{D}\right)\right] = g\left(x|\mathcal{D}\right) \equiv g\left(x\right) \tag{4}$$

With

$$Var_{\mathcal{D}}\left[g\left(x|\mathcal{D}\right)\right] = 0 \tag{5}$$

On the other hand

Because $g(x)$ was chosen arbitrarily the expected bias must be large.

$$\underbrace{\left(E_D\left[g\left(\boldsymbol{x}|\mathcal{D}\right)\right] - E\left[y|\boldsymbol{x}\right]\right)^2}_{BIAS} \tag{6}$$

Case 2

In the other hand

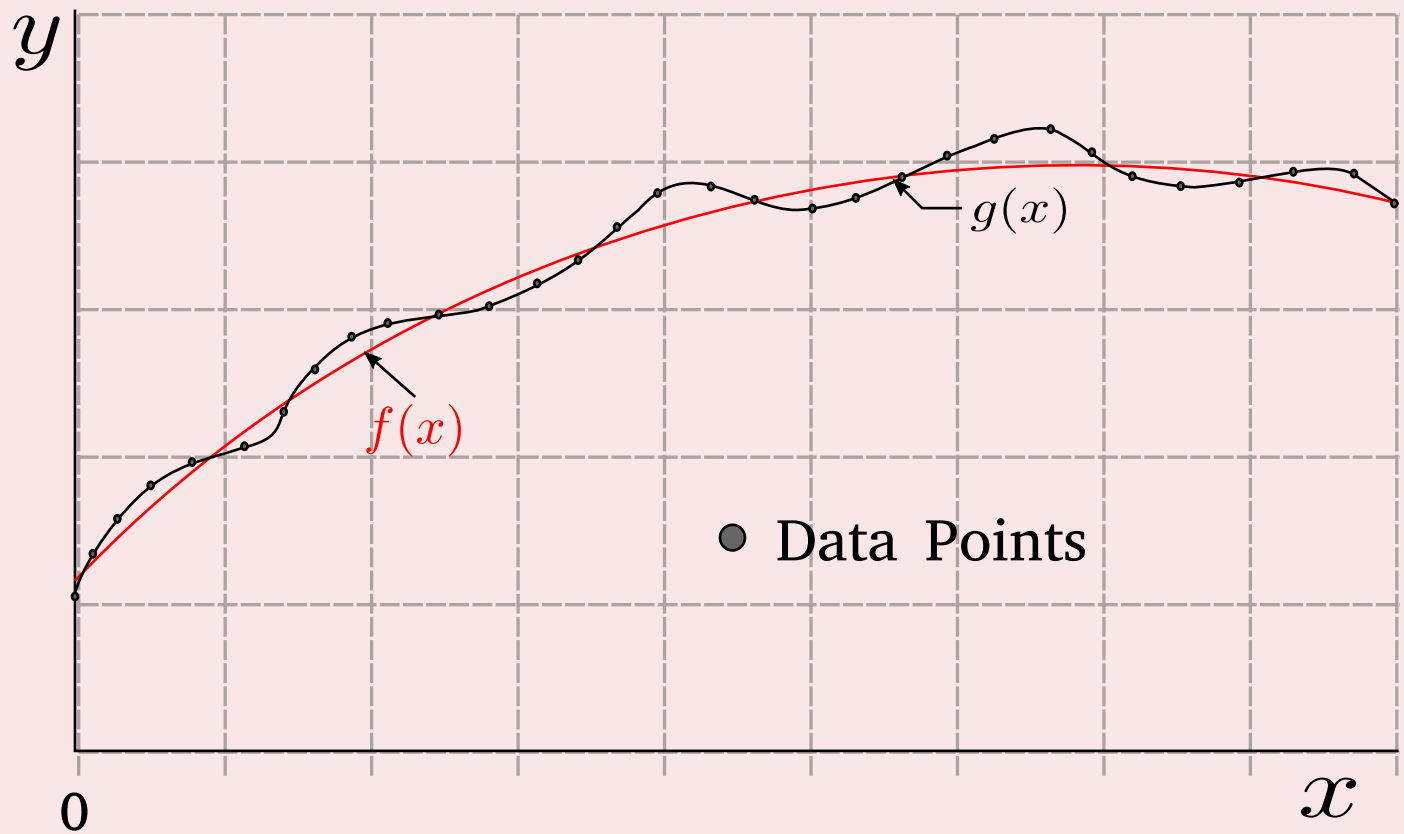
Now, $g_1(x)$ corresponds to a polynomial of high degree so it can pass through each training point in $\mathcal{D}$.

Case 2

In the other hand

Now, $g_1(x)$ corresponds to a polynomial of high degree so it can pass through each training point in $\mathcal{D}$.

Example of $g_1(x)$



Case 2

Due to the zero mean of the noise source

$$E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right] = f\left(x\right) = E\left[y|x\right] \text{ for any } x = x_i \tag{7}$$

Remark: At the training points the bias is zero.

Case 2

Due to the zero mean of the noise source

$$E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right] = f\left(x\right) = E\left[y|x\right] \text{ for any } x = x_i \tag{7}$$

Remark: At the training points the bias is zero.

However the variance increases

$$\begin{aligned} E_D\left[\left(g_1\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right] &= E_D\left[\left(f\left(x\right) + \epsilon - f\left(x\right)\right)^2\right] \\ &= \sigma_\epsilon^2, \text{ for } x = x_i, i = 1, 2, ..., N \end{aligned}$$

Case 2

Due to the zero mean of the noise source

$$E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right] = f\left(x\right) = E\left[y|x\right] \text{ for any } x = x_i \tag{7}$$

Remark: At the training points the bias is zero.

However the variance increases

$$\begin{aligned} E_D\left[\left(g_1\left(\boldsymbol{x}|\mathcal{D}\right) - E_D\left[g_1\left(\boldsymbol{x}|\mathcal{D}\right)\right]\right)^2\right] &= E_D\left[\left(f\left(x\right) + \epsilon - f\left(x\right)\right)^2\right] \\ &= \sigma_\epsilon^2, \text{ for } x = x_i, i = 1, 2, ..., N \end{aligned}$$

In other words

The bias becomes zero (or approximately zero) but the variance is now equal to the variance of the noise source.

Observations

First

Everything that has been said so far applies to both the regression and the classification tasks.

Observations

First

Everything that has been said so far applies to both the regression and the classification tasks.

However

Mean squared error is not the best way to measure the power of a classifier.

Observations

First

Everything that has been said so far applies to both the regression and the classification tasks.

However

Mean squared error is not the best way to measure the power of a classifier.

Think about

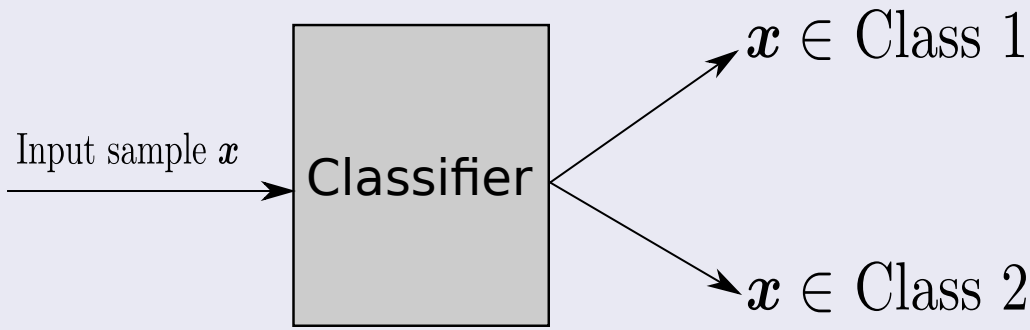
A classifier that sends everything far away of the hyperplane!!! Away from the values $+-1$!!!

Outline

Sooner of Latter you need to know how efficient is your algorithm

Thus, we need a measures of accuracy

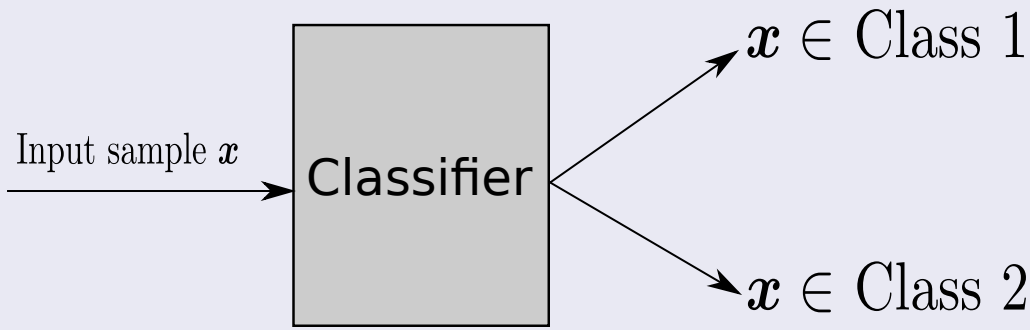
Thus, we begin with the classic classifier for two classes



Sooner of Latter you need to know how efficient is your algorithm

Thus, we need a measures of accuracy

Thus, we begin with the classic classifier for two classes



Here

A dataset used for performance evaluation is called a **test dataset.**

Therefore

It is a good idea to build a measure of performance

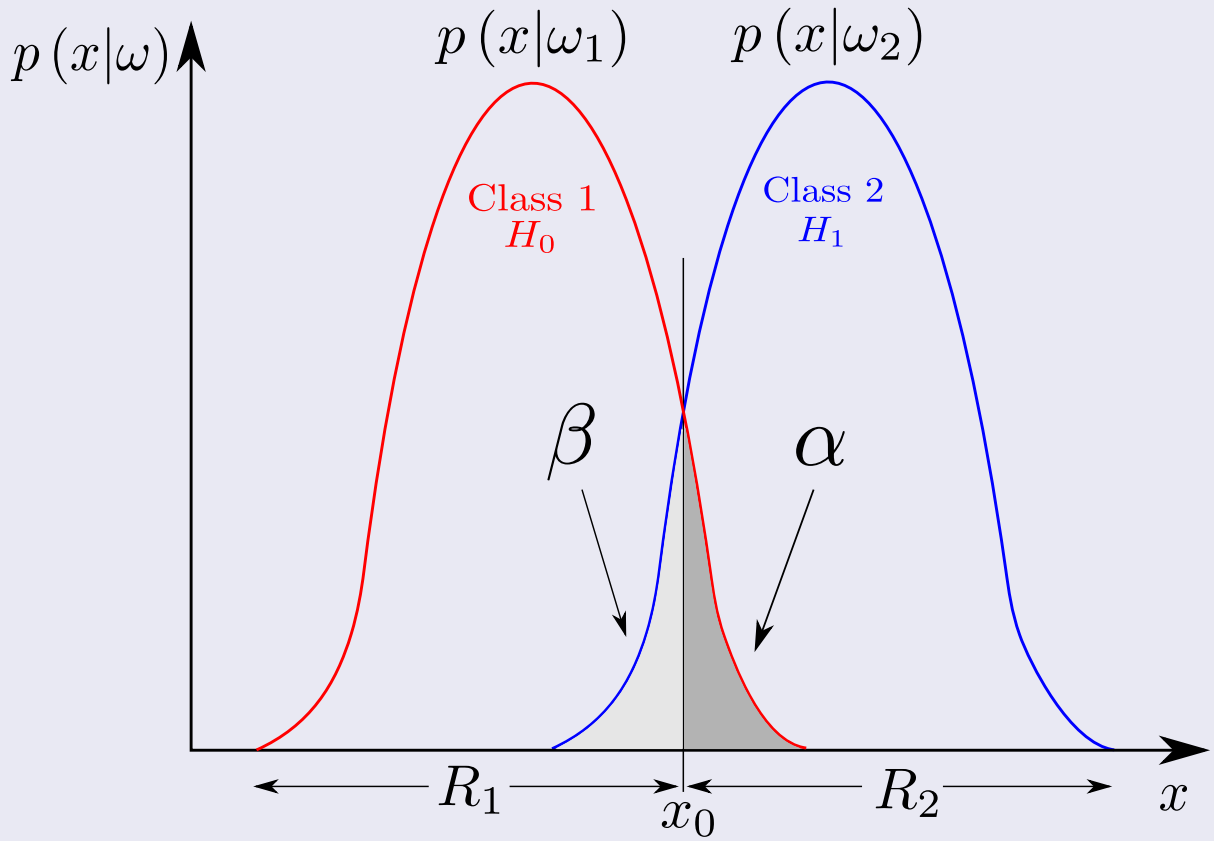
For this, we can use the idea of error in statistics.

Therefore

It is a good idea to build a measure of performance

For this, we can use the idea of error in statistics.

We have two distribution for each class



Outline

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"

α error

Definition (Type I Error - False Positive)

α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"
2. You have a device that fails $\alpha = 0.05$ meaning that it fails 5 of the time.

α error

Definition (Type I Error - False Positive)

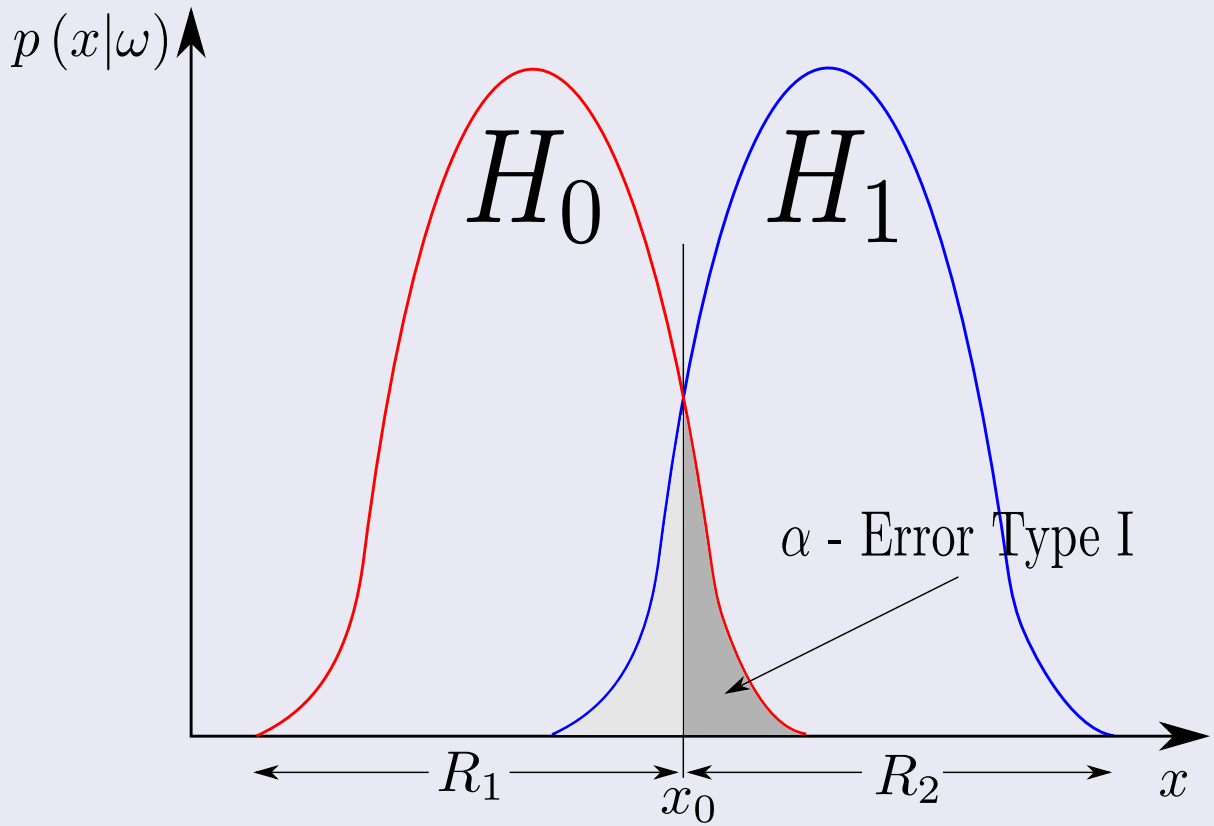
α is the probability that the test will lead to the rejection of the hypothesis H_0 when that hypothesis is true.

Example

1. H_0 : "You have a device that produce circuits with no error"
2. You have a device that fails $\alpha = 0.05$ meaning that it fails 5 of the time.
3. This says that you ha low chance of a wrong circuit.

Basically

We have



β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: “Adding fluoride to toothpaste protects against cavities."

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: “Adding fluoride to toothpaste protects against cavities."
2. Then $\beta = 0.05$ meaning that you have a chance of 5 of the time.

β error

Definition (Type II Error - False Negative)

β is the probability that the test will lead to the rejection of the hypothesis H_1 when that hypothesis is true.

Example

1. H_1: “Adding fluoride to toothpaste protects against cavities."
2. Then $\beta = 0.05$ meaning that you have a chance of 5 of the time.
3. This says that you have a low chance of having a cavity using fluoride in the water.

Outline

This can be seen as a table

Confusion Matrix

Table of error types		True H_0, False H_1	False H_0, True H_1
Decisions	Reject H_1	Correct Inference **True Positive**	Type I Error - α **False Positive**
	Reject H_0	Type II Error - β **False Negative**	Correct Inference **True Negative**

In the case of two classes, we have

We have finally the Confusion Matrix

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	**False Positives (FP)**
	Negative	**False Negatives (FN)**	**True Negatives (TN)**

Outline

Accuracy

Definition

- The proportion of getting correct classification of the Positive and Negative classes.

Accuracy

Definition

- The proportion of getting correct classification of the Positive and Negative classes.

Thus

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

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	**False Positives (FP)**
	Negative	**False Negatives (FN)**	**True Negatives (TN)**

Accuracy

Definition

- The proportion of getting correct classification of the Positive and Negative classes.

Thus

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

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	**False Positives (FP)**
	Negative	**False Negatives (FN)**	**True Negatives (TN)**

Problem - accuracy assumes equal cost for both kinds of errors

Is 99% accuracy good, bad or terrible? It depends on the problem.

Another Problem

You have only a way to measure the total correct answers

- Yes, we have at the numerator of $TP + TN$

Another Problem

You have only a way to measure the total correct answers

- Yes, we have at the numerator of $TP + TN$

We need to measure for the Positive Class/Class 1

- Yes, we need to measure the moments when the correct answers can be correct

Another Problem

You have only a way to measure the total correct answers

- Yes, we have at the numerator of $TP + TN$

We need to measure for the Positive Class/Class 1

- Yes, we need to measure the moments when the correct answers can be correct

The True Positive or

- The Recall Rate...

True Positive Rate

Also called

- Sensitivity or **Recall Rate**

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	False Positives (FP)
	Negative	**False Negatives (FN)**	True Negatives (TN)

True Positive Rate

Also called

- Sensitivity or **Recall Rate**

Defined as

- True Positive Rate is the proportion of getting a correct classification of the Positive Class vs the True Positive and False Negatives.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	False Positives (FP)
	Negative	**False Negatives (FN)**	True Negatives (TN)

We also need to measure the Class 2

For this, we have

- Specificity

True Negative Rate

Also known as

- Specificity

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	True Positive (TP)	**False Positives (FP)**
	Negative	False Negatives (FN)	**True Negatives (TN)**

True Negative Rate

Also known as

- Specificity

Defined as

- It is the proportion of True Negative vs the elements classified as True negatives.

$$\text{True Negative Rate} = \frac{TN}{FP + TN}$$

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	True Positive (TP)	**False Positives (FP)**
	Negative	False Negatives (FN)	**True Negatives (TN)**

We also need to measure the rate of TP

We have for this

- The Precision

Precision

Also known as

- **Positive Predictive Value**

Precision

Also known as

- **Positive Predictive Value**

Defined as

- The proportion of the elements classified as true positive vs the total of all the real true positives.

$$\text{Precision Predicted Value} = \frac{TP}{FP + TP}$$

		Actual Class	
		Positive	Negative
Predicted Classes	Positive	**True Positive (TP)**	**False Positives (FP)**
	Negative	False Negatives (FN)	True Negatives (TN)

Significance Level

Also known as

False Positive Rate.

Significance Level

Also known as

False Positive Rate.

Defined as

False Positive Rate is the probability of getting an incorrect classification of the Positive Class vs the True Negative and the False Positive.

$$\text{False positive rate} = \frac{FP}{TN + FP}$$

Outline

We can do better than these simple measures of accuracy

Given these initial measures of validity

it is possible to obtain a more precise model evaluation, the ROC curves.

We can do better than these simple measures of accuracy

Given these initial measures of validity

it is possible to obtain a more precise model evaluation, the ROC curves.

The ROC Curves plot

It is a model-wide evaluation measure that is based on two basic evaluation measures:

1. **Specificity** is a performance measure of the whole negative part of a dataset.
2. **Sensitivity** is a performance measure of the whole positive part.

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

$$\text{Specificity} = \text{False positive rate} = \frac{FP}{TN + FP}$$

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

What the ROC Curves uses

We have a plot where

The ROC plot uses specificity on the x-axis and sensitivity on the y-axis.

Basically

False Positive Rate (FPR) is identical with specificity, and True Positive Rate (TPR) is identical with sensitivity.

$$\text{Specificity} = \text{False positive rate} = \frac{FP}{TN + FP}$$

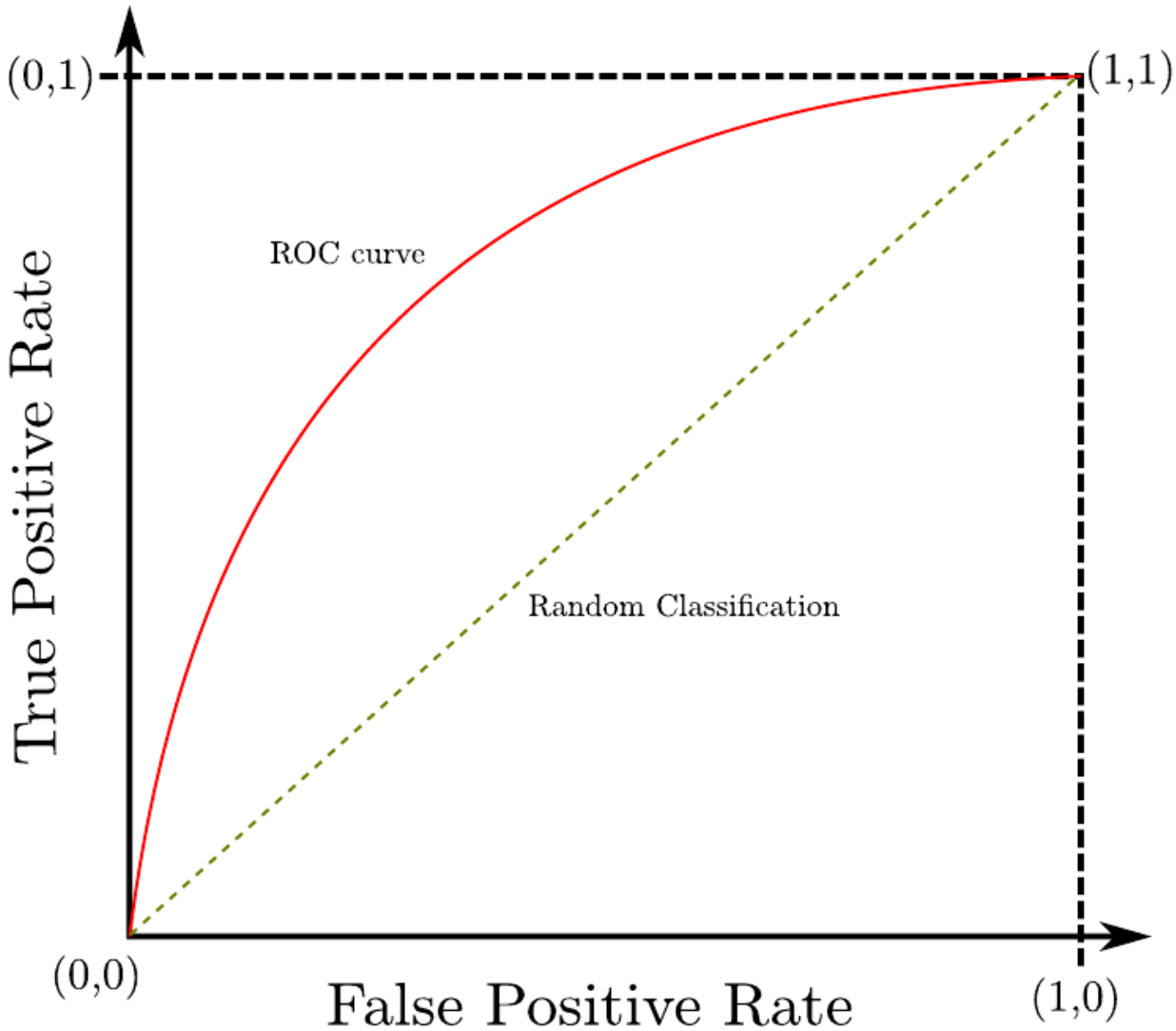
$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

Then

1. A ROC curve is created by connecting all ROC points of a classier in the ROC space.
2. Two adjacent ROC points can be connected by a straight line.
3. The curve starts at (0.0, 0.0) and ends at (1.0, 1.0).

Outline

For Example


(0,1)
(1,1)
ROC curve
Random Classification
True Positive Rate
(0,0)
False Positive Rate
(1,0)

Outline

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle ; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. **if** L_{sorted} **is a positive example then** $TP = TP + 1$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle ; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8. **else** $FP = FP + 1$

We have

Algorithm ROC point generation

Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle ; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8. **else** $FP = FP + 1$
9. $i \leftarrow i + 1$

We have

Algorithm ROC point generation

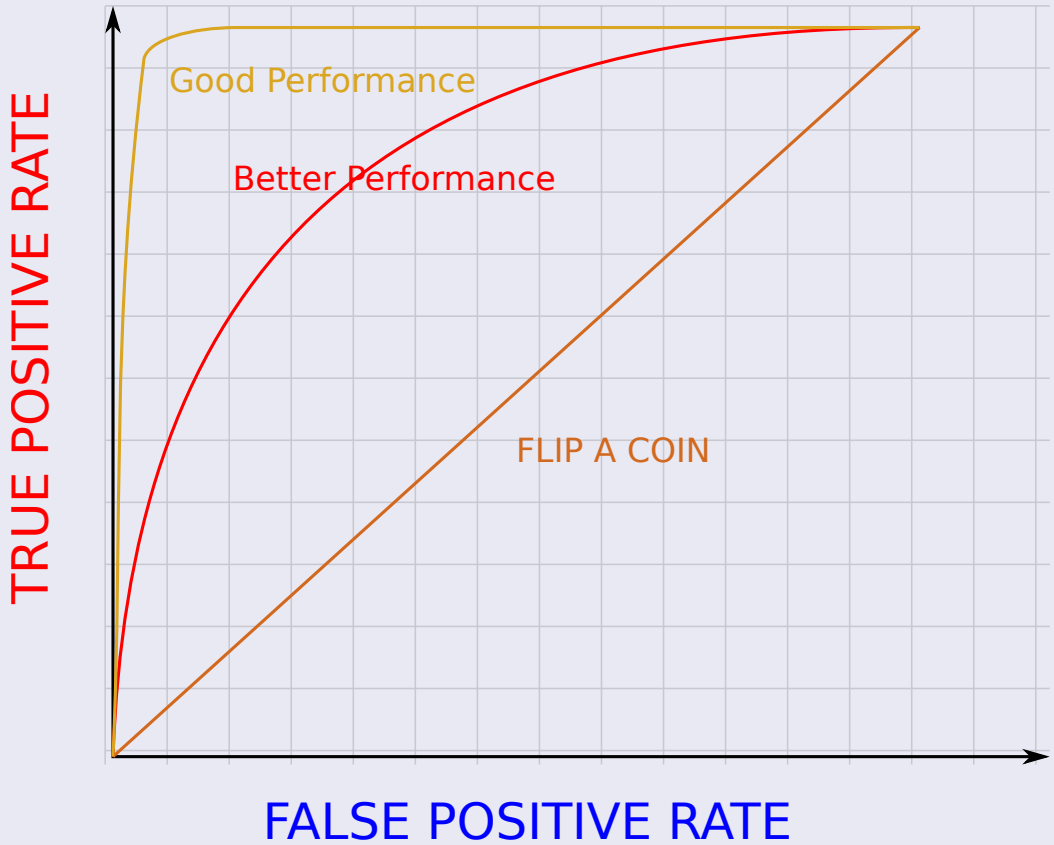
Input: L, the set of test examples; $f(i)$, the probabilistic classifier estimate that example i is positive; P and N, the number of positive and negative examples.

Ouput: R, a list of ROC points increasing by false positive rate.

1. $L_{sorted} \leftarrow L$ **sorted decreasing by** f **scores**
2. $FP \leftarrow TP \leftarrow 0; R \leftarrow \langle\rangle; fprev \leftarrow -\infty; i \leftarrow 1$
3. **while** $i \leq |L_{sorted}|$
4. **if** $f(i) \neq f_{prev}$ **then**
5. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$
6. $fprev \leftarrow f(i)$
7. **if** L_{sorted} **is a positive example then** $TP = TP + 1$
8. **else** $FP = FP + 1$
9. $i \leftarrow i + 1$
10. $R.append\left(\frac{FP}{N}, \frac{TP}{P}\right)$**, this is** $(1, 1)$

For Example

We could have multiple methods



Thus

Thus

- Thus, after generating the ROC Curve it is possible to use several metrics to validate using the ROC curves.

Thus

Thus

- Thus, after generating the ROC Curve it is possible to use several metrics to validate using the ROC curves.

A Partial List is

1. Area Under the Curve (AUC)
2. Equal Error Rate (EER)
3. Likelihood Ratio

Outline

A Simple Defintion

We have

$$AUC = \int ROC\left(p\right) dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

A Simple Defintion

We have

$$AUC = \int ROC\left(p\right) dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

This equation has the following meaning

- The probability that a randomly selected observation X from the **positive class** would have a higher score than a randomly selected observation Y from the **negative class.**

$$P\left(X > Y\right)$$

A Simple Defintion

We have

$$AUC = \int ROC\left(p\right) dp = \sum_{i=1}^{N} ROC\left(f\left(\frac{1}{i}\right)\right)\left[\frac{i}{N} - \frac{i-1}{N}\right]$$

This equation has the following meaning

- The probability that a randomly selected observation X from the **positive class** would have a higher score than a randomly selected observation Y from the **negative class.**

$$P\left(X > Y\right)$$

Thus

The AUC gives the mean **true positive** rate averaged uniformly across the **false positive** rate.

Therefore

AUC curves are a good measure of how good are our results

- However, we need to combine this results with something more powerful
  - Cross Validation - to understand the variation in the machine estimation

Outline

Also known as F_1 score

It is a measure of a test's accuracy

- It considers both the precision P and the recall R of the test to compute the score.

Also known as F_1 score

It is a measure of a test's accuracy

- It considers both the precision P and the recall R of the test to compute the score.

An interesting fact

- It computes some average of the information retrieval precision and recall.

Remember

Precision

- The proportion of the elements classified as true positive vs the total of all the real true positives.

$$\text{Precision Predicted Value} = \frac{TP}{FP + TP}$$

Remember

Precision

- The proportion of the elements classified as true positive vs the total of all the real true positives.

$$\text{Precision Predicted Value} = \frac{TP}{FP + TP}$$

Recall

- True Positive Rate is the proportion of getting a correct classification of the Positive Class vs the True Positive and False Negatives.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$

Building the F_1 score

Something Notable

$$Average = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$Harmonic = \frac{N}{\sum_{i=1}^{N} \frac{1}{x_i}}$$

Building the F_1 score

Something Notable

$$Average = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$Harmonic = \frac{N}{\sum_{i=1}^{N} \frac{1}{x_i}}$$

When $x_1 = Precision$ and $x_2 = Recall$

$$Average = \frac{1}{2}(P + R)$$

$$Harmonic = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

Thus

Important

- The harmonic mean is more intuitive than the arithmetic mean when computing a mean of ratios.

Thus

Important

- The harmonic mean is more intuitive than the arithmetic mean when computing a mean of ratios.

Example

- Suppose that you have a finger print recognition system and its precision and recall be 1.0 and 0.2

Therefore, we have

The Average will report

$$\frac{1}{2}(P+R) = \frac{1.0+0.2}{2} = 0.6$$

Therefore, we have

The Average will report

$$\frac{1}{2}(P+R) = \frac{1.0+0.2}{2} = 0.6$$

At the F_2 score

$$\frac{2PR}{P+R} = \frac{0.4}{1.2} = 0.33$$

General Form F_β

Then for Precision and Recall, we have a general function

$$F_\beta = \frac{(\beta^2 + 1)\, Precision \times Recall}{\beta^2 Precision + Recall} \quad (0 \leq \beta \leq +\infty)$$

General Form F_β

Then for Precision and Recall, we have a general function

$$F_\beta = \frac{(\beta^2 + 1)\, Precision \times Recall}{\beta^2 Precision + Recall} \quad (0 \leq \beta \leq +\infty)$$

Thus, for the basic case F_1

$$F_1 = 2\frac{Precision \times Recall}{Precision + Recall}$$

Outline

What we want

We want to measure

A quality measure to measure different classifiers (for different parameter values).

What we want

We want to measure

A quality measure to measure different classifiers (for different parameter values).

We call that as

$$R(f) = E_{\mathcal{D}}\left[L\left(y, f\left(\boldsymbol{x}\right)\right)\right]. \tag{8}$$

Example: $L\left(y, f\left(x\right)\right) = \left\|y - f\left(\boldsymbol{x}\right)\right\|_2^2$

What we want

We want to measure

A quality measure to measure different classifiers (for different parameter values).

We call that as

$$R(f) = E_{\mathcal{D}}\left[L\left(y, f\left(\boldsymbol{x}\right)\right)\right]. \tag{8}$$

Example: $L\left(y, f\left(x\right)\right) = \left\|y - f\left(\boldsymbol{x}\right)\right\|_2^2$

More precisely

For different values γ_j of the parameter, we train a classifier $f\left(\boldsymbol{x}|\gamma_j\right)$ on the training set.

Then, calculate the empirical Risk

Do you have any ideas?

Give me your best shot!!!

Then, calculate the empirical Risk

Do you have any ideas?

Give me your best shot!!!

Empirical Risk

We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i, f\left(\boldsymbol{x}_i|\gamma\right)\right) \tag{9}$$

Then, calculate the empirical Risk

Do you have any ideas?

Give me your best shot!!!

Empirical Risk

We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i, f\left(\boldsymbol{x}_i|\gamma\right)\right) \tag{9}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.

Then, calculate the empirical Risk

Do you have any ideas?

Give me your best shot!!!

Empirical Risk

We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i, f\left(\boldsymbol{x}_i|\gamma\right)\right) \tag{9}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.
2. Re-train the classifier with parameter γ^* on all data except the test set (i.e. train + validation data).

Then, calculate the empirical Risk

Do you have any ideas?

Give me your best shot!!!

Empirical Risk

We use the validation set to estimate

$$\hat{R}\left(f\left(x|\gamma\right)\right) = \frac{1}{N_v}\sum_{i=1}^{N_v} L\left(y_i, f\left(\boldsymbol{x}_i|\gamma\right)\right) \tag{9}$$

Thus, you follow the following procedure

1. Select the value γ^* which achieves the smallest estimated error.
2. Re-train the classifier with parameter γ^* on all data except the test set (i.e. train + validation data).
3. Report error estimate $\hat{R}\left(f\left(x|\gamma_i\right)\right)$ computed on the test set.

Idea

Something Notable

- Each of the **error estimates computed on validation set** is computed from a single example of a trained classifier.
  - Can we improve the estimate?

Idea

Something Notable

- Each of the **error estimates computed on validation set** is computed from a single example of a trained classifier.
  - Can we improve the estimate?

K-fold Cross Validation

To estimate the risk of a classifier f:

Idea

Something Notable

- Each of the **error estimates computed on validation set** is computed from a single example of a trained classifier.
  - Can we improve the estimate?

K-fold Cross Validation

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds"), N_v.

Idea

Something Notable

- Each of the **error estimates computed on validation set** is computed from a single example of a trained classifier.
  - Can we improve the estimate?

K-fold Cross Validation

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds"), N_v.
2. Train an instance f_k of the classifier, using all folds except fold k as training data.

Idea

Something Notable

- Each of the **error estimates computed on validation set** is computed from a single example of a trained classifier.
  - Can we improve the estimate?

K-fold Cross Validation

To estimate the risk of a classifier f:

1. Split data into K equally sized parts (called "folds"), N_v.
2. Train an instance f_k of the classifier, using all folds except fold k as training data.
3. Compute the Cross Validation (CV) estimate:

$$\hat{R}_{CV}\left(f\left(x|\gamma\right)\right) = \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, f_k\left(\boldsymbol{x}_{k(i)}|\gamma\right)\right) \tag{10}$$

where $k\left(i\right)$ is the fold containing $\boldsymbol{x}_i$.

Example

$K = 5, k = 3$

Train	Train	Testing	Train	Train
1	2	3	4	5

Example

$K = 5, k = 3$

Train	Train	Testing	Train	Train
1	2	3	4	5

Actually, we have

- A more general setup

$\overbrace{\text{SPLIT All Train Set}}^{\text{Train Data + Validation Data}}$	Test

Outline

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.

How to choose K

Extremal cases

- $K = N$, called leave one out cross validation (loocv)
- $K = 2$

An often-cited problem with loocv is that we have to train many ($= N$) classifiers, but there is also a deeper problem.

Argument 1: K should be small, e.g. $K = 2$

1. Unless we have a lot of data, variance between two distinct training sets may be considerable.
2. Important concept: By removing substantial parts of the sample in turn and at random, we can simulate this variance.
3. By removing a single point (loocv), we cannot make this variance visible.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.

How to choose K

Argument 2: K should be large, e.g. $K = N$

1. Classifiers generally perform better when trained on larger data sets.
2. A small K means we substantially reduce the amount of training data used to train each f_k, so we may end up with weaker classifiers.
3. This way, we will systematically overestimate the risk.

Common recommendation: $K = 5$ to $K = 10$

Intuition:

1. $K = 10$ means number of samples removed from training is one order of magnitude below training sample size.
2. This should not weaken the classifier considerably, but should be large enough to make measure variance effects.

Outline

Outline

Leave p out cross-validation

Definition

- It involves using p-observation as validation data, and remaining data is used to train the model.

Leave p out cross-validation

Definition

- It involves using p-observation as validation data, and remaining data is used to train the model.

Basically

- This is repeated in all ways to cut the original sample on a validation set of p observations and a training set.

Leave p out cross-validation

Definition

- It involves using p-observation as validation data, and remaining data is used to train the model.

Basically

- This is repeated in all ways to cut the original sample on a validation set of p observations and a training set.

Notes

- A variant of LpOCV with $p = 2$ known as leave-pair-out cross-validation has been recommended as a nearly unbiased method for estimating the area under ROC curve of a binary classifier.

Leave-one-out cross-validation (LOOCV)

Definition

- It is a category of LpOCV with the case of $p = 1$.

Leave-one-out cross-validation (LOOCV)

Definition

- It is a category of LpOCV with the case of $p = 1$.

Basically

Train	Train	Train	Test $= 1$	Train	Train

Pros and Cons

Pros

- Simple, easy to understand, and implement.

Pros and Cons

Pros

- Simple, easy to understand, and implement.

Cons

- The model may lead to a low bias.
- The computation time required is high.

Outline

Holdout cross-validation

Definition

- The holdout technique is an exhaustive cross-validation method.
- It randomly splits the dataset into train and test data.
  - For example, 70% for train and 30% for Validation

Holdout cross-validation

Definition

- The holdout technique is an exhaustive cross-validation method.
- It randomly splits the dataset into train and test data.
  - For example, 70% for train and 30% for Validation

Pros

- Simple to understand

Holdout cross-validation

Definition

- The holdout technique is an exhaustive cross-validation method.
- It randomly splits the dataset into train and test data.
  - For example, 70% for train and 30% for Validation

Pros

- Simple to understand

Cons

- Not suitable for an imbalanced dataset.
- Requieres large amount of data

Outline

K-Fold Cross Validation

Definition

- In k-fold cross-validation, the original dataset is equally partitioned into k subparts or folds.

K-Fold Cross Validation

Definition

- In k-fold cross-validation, the original dataset is equally partitioned into k subparts or folds.

Thus

- Out of the k-folds or groups, for each iteration, one group is selected as validation data,
- The remaining $(k-1)$ groups are selected as training data.

Finally

We take the mean accuracy of the k-folds

$$acc_{cv} = \frac{1}{K} \sum_{i=1}^{K} acc_i$$

Finally

We take the mean accuracy of the k-folds

$$acc_{cv} = \frac{1}{K}\sum_{i=1}^{K} acc_i$$

Pros

- The model has low bias and Low time complexity
- The entire dataset is utilized for both training and validation.

Finally

We take the mean accuracy of the k-folds

$$acc_{cv} = \frac{1}{K} \sum_{i=1}^{K} acc_i$$

Pros

- The model has low bias and Low time complexity
- The entire dataset is utilized for both training and validation.

Cons

- Not suitable for an imbalanced dataset.

Outline

Repeated Subsampling

Definition

- Repeated random subsampling validation also referred to as Monte Carlo cross-validation splits the dataset randomly into training and validation.

Repeated Subsampling

Definition

- Repeated random subsampling validation also referred to as Monte Carlo cross-validation splits the dataset randomly into training and validation.

Something Notable

- Unlikely k-fold cross-validation split of the dataset into not in groups or folds but splits in this case in random.
- Using multiple Iterations to performa an average accuracy

Finally

Pros

- The proportion of train and validation splits is not dependent on the number of iterations or partitions.

Finally

Pros

- The proportion of train and validation splits is not dependent on the number of iterations or partitions.

Cons

- Some samples may not be selected for either training or validation.
- Not suitable for an imbalanced dataset.

Outline

Stratisfied K-fold Cross-Validation

Something Notable

- For all the cross-validation techniques discussed above, they may not work well with an imbalanced dataset.
  - Stratified k-fold cross-validation solved the problem of an imbalanced dataset.

Stratisfied K-fold Cross-Validation

Something Notable

- For all the cross-validation techniques discussed above, they may not work well with an imbalanced dataset.
  - Stratified k-fold cross-validation solved the problem of an imbalanced dataset.

Definition

- In Stratified k-fold cross-validation, the dataset is partitioned into k groups or folds
  - The validation data has an equal number of instances of target class label.

Therefore

Final Score

- The final score is computed by taking the mean of scores of each fold.

Therefore

Final Score

- The final score is computed by taking the mean of scores of each fold.

Pros

- Works well for an imbalanced dataset.

Therefore

Final Score

- The final score is computed by taking the mean of scores of each fold.

Pros

- Works well for an imbalanced dataset.

Cons

- Now suitable for time series dataset.

Outline

A Combination

An Image Better than 100 words