Model Evaluation and Classification Metrics

Model Evaluation and Classification metrics

- **Classification Metrics**
  - **Confusion matrix**
  - **Accuracy**
  - **Precision and Recall**
  - **F1 Score**
  - **Sensitivity , specificity nd Gmean**
  - **AUC curve**
- **Regression metrics,**
  - **Mean Squared Error or MSE**
  - **Root Mean Squared Error or RMSE**
  - **Mean Absolute Error or MAE**
  - **Root Mean Squared Log Error or RMSLE**

Introduction

- Comparing the suitability of alternative machine-learning techniques for induction in a given domain
- Dividing the set of pre-classified examples randomly into two subsets (one for induction, the other for testing) may not be the best thing to do, especially if the training set is small; random division may then result in subsets that do not represent the given domain properly.
- To obtain more reliable results, *repeated random runs are necessary*.

Confusion Matrix (Error matrix.)

- A **confusion matrix** is a performance evaluation tool in machine learning, representing the accuracy of a classification model.
- It displays the number of **true positives, true negatives, false positives, and false negatives.**
- This matrix aids **in analyzing model performance, identifying mis-classifications, and improving predictive accuracy**.

Confusion Matrix (Error matrix.)

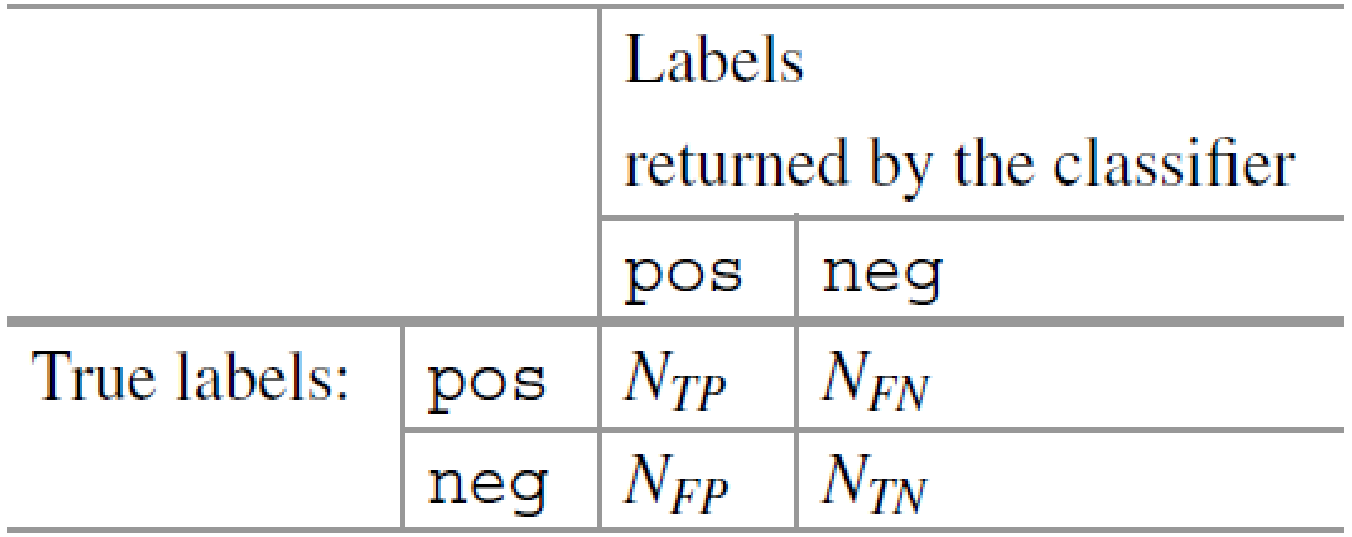
- A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the total number of target classes.
- The matrix compares the actual target values with those predicted by the machine learning model.
- This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

Why Do We Need a Confusion Matrix?

- We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:
  1. **Classification Accuracy**
  2. **Misclassification rate**
  3. **Precision**
  4. **Recall**
  5. **F-measure**
  6. **Null Error rate**
  7. **ROC Curve**

Confusion Matrix (Error matrix.)

- For a binary classification problem, we would have a 2 x 2 matrix, as shown below, with 4 values:



		Labels returned by the classifier	
		pos	neg
True labels:	pos	N_{TP}	N_{FN}
	neg	N_{FP}	N_{TN}

The number of correct classifications :

$$N_{TP} + N_{TN}$$

The number of errors made is: $N_{FP} + N_{FN}$

Total size of the example set $|T| = N_{FP} + N_{FN} + N_{TP} + N_{TN}.$

Confusion Matrix (Error matrix.)

- **Error Rate :**
- A classifier’s error rate, E, is the frequency of errors made by the classifier over a given set of examples.
- It is calculated by dividing the number of errors, by the total number of examples

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

Confusion Matrix (Error matrix.)

Classification Accuracy :

- It is the frequency of correct classifications made by the classifier over a given set of examples.
- Classification accuracy is calculated by dividing the number
- of correct classifications by the total number of examples

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

- Note that $Acc = 1 - E$

Confusion Matrix (Error matrix.)

Rejecting an Example

- It is suggested that the classifier should sometimes be allowed **to refuse to classify an example** if the evidence supporting the winning class is **not strong enough.**
- **The motivation is quite simple:** in some domains, the penalty for misclassification can be much higher than the penalty for not making any classification at all.
- **E.g.:** An incorrect medical diagnosis is often more expensive than no diagnosis at all. A wrong diagnosis may result in choosing a treatment that does more harm than good.

Confusion Matrix (Error matrix.)

Rejecting an Example

- ***Insufficient evidence*** is easy to define :
- For example : In a **7-NN classifier**, four neighbors **favor the pos class**, and the remaining **three favor the neg class.** The situation seems **“too close to call”**
- So the you may **define a threshold** for the minimum difference between the number of votes favoring the winning class and the number of votes favoring the runner-up class.
- The **classifier may refuse to classify if the value is below threshold.**

Confusion Matrix (Error matrix.)

Rejecting an Example

- In **Bayesian classifiers**, it may reject to classify if the difference between the probabilities of the two most strongly supported classes falls short of a user-specified minimum.
- Something similar can be done also in neural networks.
- In other classifiers, such as **decision trees**, implementation of the rejection mechanism **is not so straightforward**

Confusion Matrix (Error matrix.)

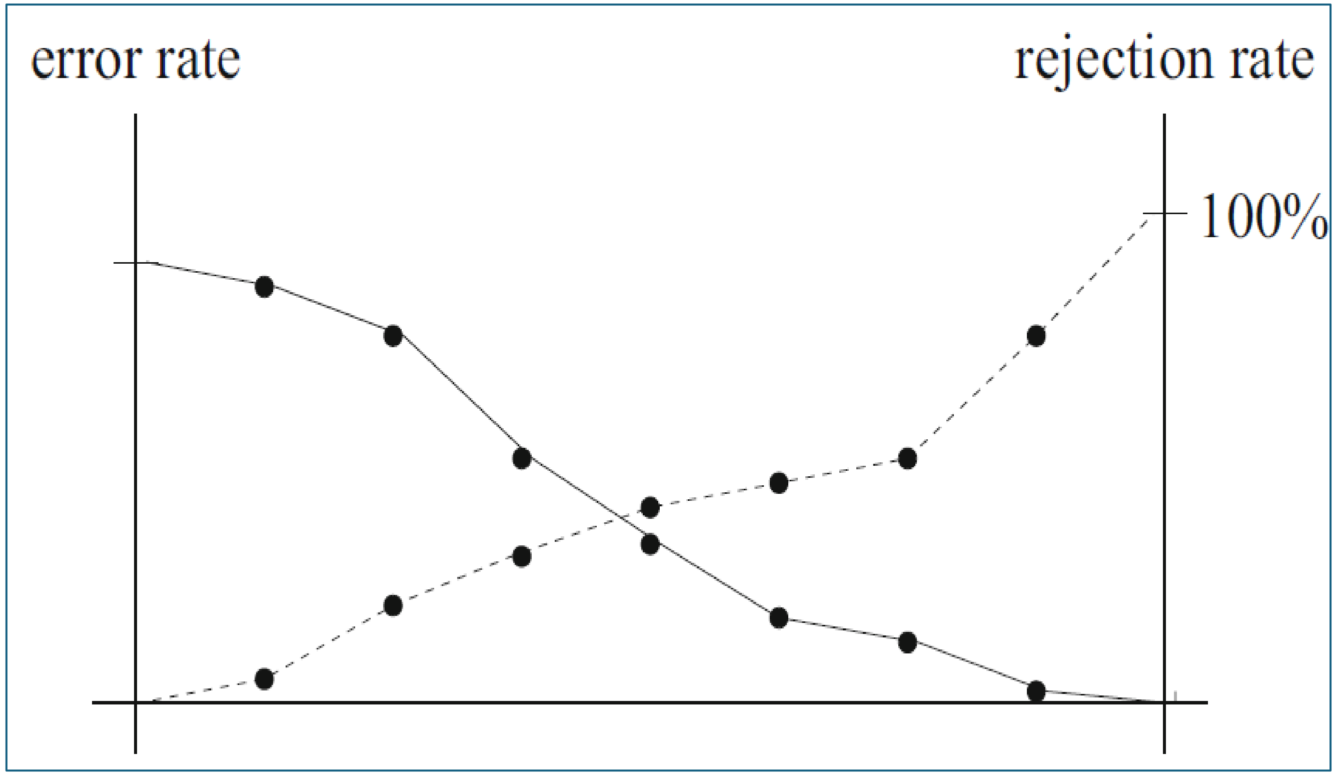
Advantages and Disadvantages of a Rejection to Classify

- The classifier that occasionally refuses to classify is of course less likely to go wrong.
- Indeed, the more examples are rejected, **the lower the error rate.**
- But if very low rate is achieved only because of the refusal to classify almost all examples, the classifier becomes impractical.

Confusion Matrix (Error matrix.)

Advantages and Disadvantages of a Rejection to Classify

- So there is a need to trade off between low error rate and rejection rate



The lesson is clear. Occasional rejection of unclear examples makes a lot of sense, but the principle should be handled with care.

Precision and Recall

What is the problem with Accuracy?

- Let’s think about a hypothetical classification problem: **Predicting how many people will be infected with a contagious virus in times before they show the symptoms and isolate them from the healthy population.**
- The two values for our target variable would be **Sick and Not Sick.**
- Let’s assume that our dataset is an **imbalanced dataset**. Assume that there are 947 data points for the **negative class (no sick)** and 3 data points for the **positive class (sick)**.

Precision and Recall

What is the problem with Accuracy in this case?

- We want to measure **how many positive cases we can predict correctly** to arrest the spread of the contagious virus.

- If **TP = 30, TN = 930, FP = 30, FN = 10 , then**

$$\boxed{Accuracy = \frac{30 + 930}{30 + 30 + 930 + 10} = 0.96}$$

- **So 96% of the times the classifier** correctly predicts. However, **93%** of the times it correctly predicts negative cases and only **3%** of the times it correctly predicts positive cases. That it is doing exactly opposite of what we expected. Therefore, this is an **incorrect metric for our model.**

Precision and Recall

- The majority of realistic applications are in some degree marked by the phenomenon of imbalanced classes.
- For example, patients suffering from a specific medical disorder are in the entire population relatively rare.
- In domains of this kind, error rate and classification accuracy will hardly tell us anything reasonable about the classifier's practical utility.
- We need criteria that focus on a class which, while important, is represented by only a few examples. This is where we come across the dual concept of **Precision and Recall.**

Precision and Recall

- Precision is the **percentage of true positives**, N_{TP}, among all examples that the classifier has labeled as positive: $N_{TP} + N_{FP}$.
- The value is thus obtained by the following formula:

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

- Put another way, pr lity that the classifier is right when labeling an example as positive.

Precision and Recall

- **Recall means** the probability that **a positive example will be correctly recognized as such** (by the classifier).
- The value is therefore obtained by dividing the number of true positives, N_{TP}, by the number of positives in the given set: $N_{TP} + N_{FN}$.

$$\boxed{Re = \frac{N_{TP}}{N_{TP} + N_{FN}}}$$

- Note that **precision is the frequency of true positives among all examples deemed positive** by the classifier, recall is **the frequency of the same true positives among all positive examples in the set**.

Precision and Recall

- Suppose a classifier has been induced. Evaluation on a testing set gave the results summarized in this table:

		Labels returned by the classifier	
		pos	neg
True labels:	pos	20	50
	neg	30	900

$$\boxed{\text{precision} = \frac{20}{50} = 0.40; \quad \text{recall} = \frac{20}{70} = 0.29; \quad \text{accuracy} = \frac{920}{1000} = 0.92}$$

Precision and Recall

- Just notice that the given dataset is highly imbalanced.
- The induced classifier, while exhibiting an **impressive classification accuracy**, **suffers from poor precision and recall**.
- Specifically, precision of 0.40 means that of the **50 examples predicted as positive by the classifier**, only **20 are indeed positive.** the remaining **30 being nothing but false positives.**
- With recall, **things are even worse**: out of the 70 positive examples in the testing set, **only 20** are correctly identified as such by the classifier.

Precision and Recall

- Suppose the classifier's parameters were modified with the intention to improve its behavior on positive examples. After the modification, evaluation on a testing set gave the results summarized in the table below.

		Labels returned by the classifier	
		pos	neg
True labels:	pos	30	70
	neg	20	880

You can see that precision has considerably improved, while recall remained virtually unaffected. Note that classification accuracy has not improved, either.

$$\text{precision} = \frac{30}{50} = 0.60; \quad \text{recall} = \frac{30}{100} = 0.30; \quad \text{accuracy} = \frac{910}{1000} = 0.91$$

Precision and Recall

- **When High Precision Matters?**
  - For instance, when you purchase something from an e-commerce web site, their recommender system often reacts **“Customers who have bought X buy also Y.”**
  - Here precision is more important than recall because many of the recommended merchandise should make sense that they are related to the products bought.
  - Recall is not so important here because it does not matter that small percentage of all items that the customers may like are recommended.

Precision and Recall

When High Recall Matters?

- For instance, patient suffering from X but not diagnosed as such represents a false negative, something the doctor wants to avoid.
- Thus, for this case we want maximize Recall by minimizing false negatives in the Recall formula.
- Consequently, small value of N_{FN} implies a high value of recall.

ROC Curves

- It stands for **Receiver Operating Characteristic Curves**
- If the user has an idea as to which of these two quantities is more important, tweaking certain parameters can modify the values of N_{FP} and N_{FN}, improving recall or precision required.
- E.g. in k-NN **4 +ve vs 3-ve** neighbor case, classifier can be instructed still to return the negative label unless some very strong evidence supports the positive label.
- In this way, the number of **false positives** can be reduced, even though this often increases **false negatives**

ROC Curves

- Even stronger reduction in N_{FP} (and an increase in N_{FN}) can be achieved by requesting that any example be deemed negative unless at least five (or six) of the seven nearest neighbors are positive.
- The behavior of the classifier under different values of its parameters can be visualized by the **so-called ROC curve**, a graph where the **horizontal axis represents error rate on the negative examples (FPR – False Positive Rate)** , and the **vertical axis represents classification accuracy on the positive examples (TPR - True Positive Rate)**.

ROC Curves

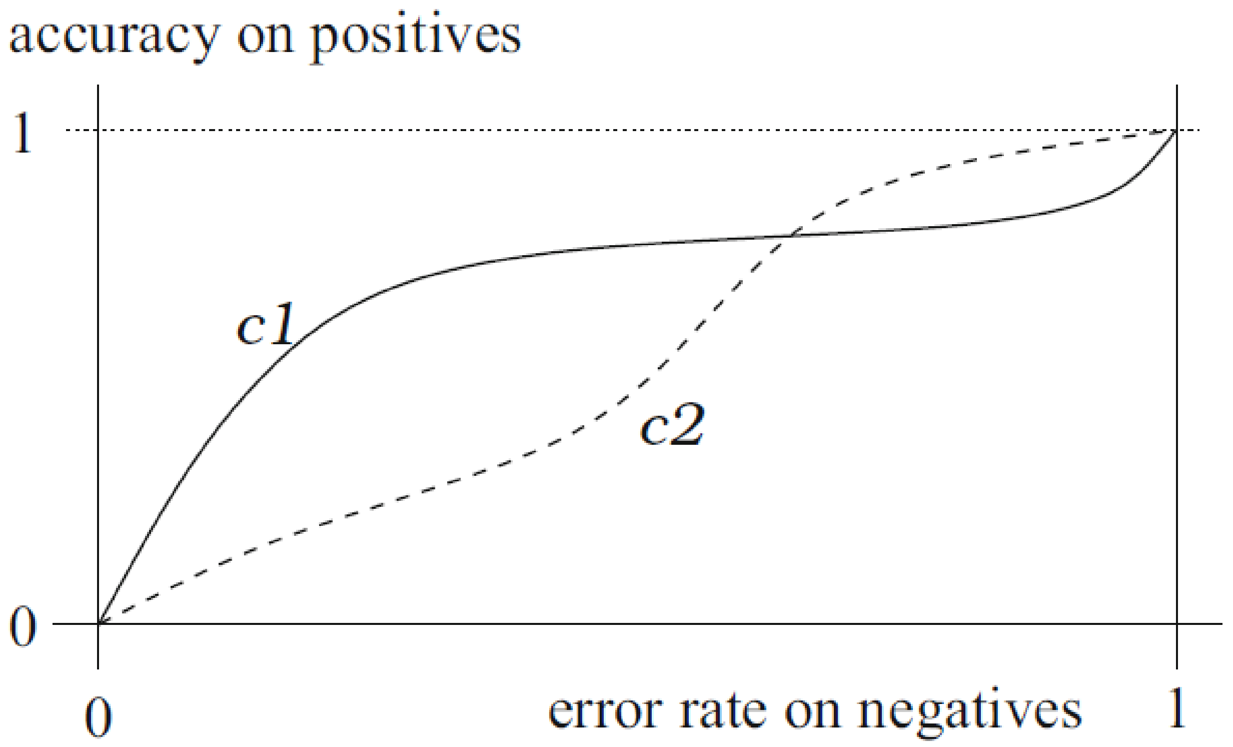


Fig. 11.2 Example of ROC curves for two classifiers, *c*1 and *c*2. The parameters of the classifiers can be used to modify the numbers of false positives and false negatives

Ideally, we would like to reach the upper-left corner that represents zero error rate on the negatives and 100% accuracy on the positives

ROC Curves

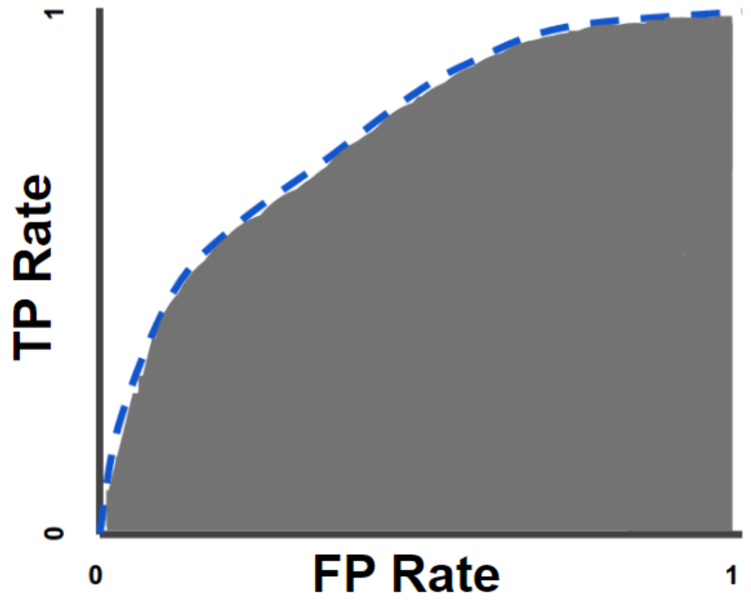
- By just looking at the graph is that c1 outperforms c2 on the positives in the region with low error rate on negatives.
- As the error rate on the negative examples increases, c2 outperforms c1 on the positive examples.
- Again, whether this is good or bad can only be decided by the user.

ROC Curves

- By just looking at the graph is that c1 outperforms c2 on the positives in the region with low error rate on negatives.
- As the error rate on the negative examples increases, c2 outperforms c1 on the positive examples.
- Again, whether this is good or bad can only be decided by the user.

ROC Curves

- **AUC** stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).



AUC provides an aggregate measure of performance across all possible classification thresholds. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

Other Ways to Measure Performance

- Other criteria are sometimes used, each reflecting a somewhat different aspect of the classifier's behavior.

Combining Precision and Recall in F_β:

- Attempts have been made somehow to combine Precision and Recall, in one quantity in the following formula:

$$F_\beta = \frac{(\beta^2 + 1) \times Pr \times Re}{\beta^2 \times Pr + Re}$$

Other Ways to Measure Performance

- The parameter, $\beta \epsilon [0, \infty]$, enables the user to weigh the relative importance of the two criteria.
- If β > 1, then more weight is given to recall.
- If β < 1, then more weight is apportioned to precision.
- It would be easy to show that $\boldsymbol{F_{\beta}}$ converges to recall when $\beta \rightarrow \infty$, and to precision when β = 0.

Other Ways to Measure Performance

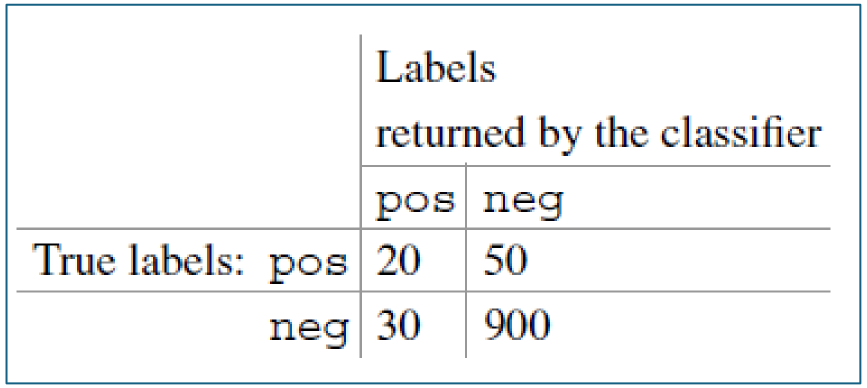
- Quite often, the engineer does not really know which of the two, precision or recall, is more important, and by how much. In that event, she prefers to work with the neutral value of the parameter, β = 1. It is called F_1 score

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re}$$

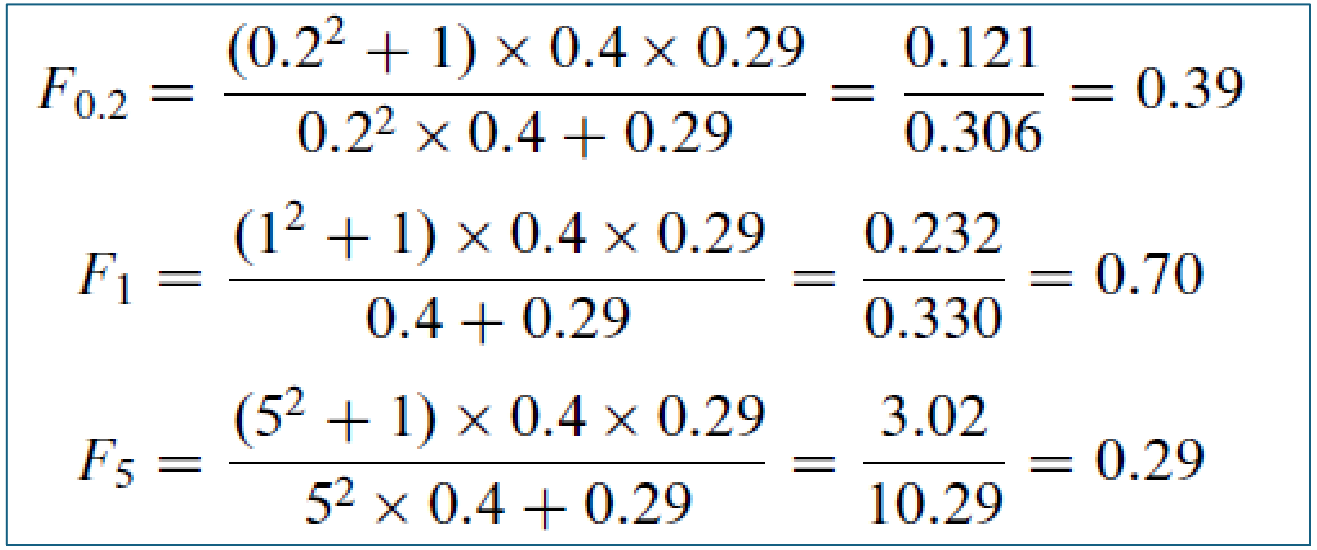
Other Ways to Measure Performance

- **A Numeric Example:** Suppose that an evaluation of a classifier on a testing set resulted in the values summarized in the following table. For these, the table established the values of precision and recall, respectively, as $\boldsymbol{Pr = 0.40}$ **and** $\boldsymbol{Re = 0.29}$. Using these numbers, we will calculate $\boldsymbol{F_\beta}$ for the following concrete settings of the parameter:

  β = 0.2, β = 1, and β = 5.



		Labels returned by the classifier	
		pos	neg
True labels:	pos	20	50
	neg	30	900



$$F_{0.2} = \frac{(0.2^2+1)\times 0.4\times 0.29}{0.2^2\times 0.4+0.29} = \frac{0.121}{0.306} = 0.39$$

$$F_1 = \frac{(1^2+1)\times 0.4\times 0.29}{0.4+0.29} = \frac{0.232}{0.330} = 0.70$$

$$F_5 = \frac{(5^2+1)\times 0.4\times 0.29}{5^2\times 0.4+0.29} = \frac{3.02}{10.29} = 0.29$$

Other Ways to Measure Performance

- **Sensitivity and Specificity: T**hese quantities are nothing but recall measured on the positive and negative examples, respectively.

Sensitivity is *recall* measured on positive examples:

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

Specificity is *recall* measured on negative examples:

$$Sp = \frac{N_{TN}}{N_{TN} + N_{FP}}$$

Other Ways to Measure Performance

- **Gmean:** When inducing a classifier in a domain with imbalanced class representation, the engineer sometimes wants to achieve similar performance on both classes**, pos and neg**. In this event, the geometric mean, **gmean**, of the two classification accuracies (on the positive examples and on the negative examples) is used:

$$\text{gmean} = \sqrt{acc_{\text{pos}} \times acc_{\text{neg}}} = \sqrt{\frac{N_{TP}}{N_{TP} + N_{FN}} \times \frac{N_{TN}}{N_{TN} + N_{FP}}}$$

- $= \sqrt{sensitivity \times specificity}$

Other Ways to Measure Performance

- Note that gmean is actually the square root of the product of two numbers: **recall on positive examples and recall on negative examples**—or, in other words, the product of sensitivity and specificity.

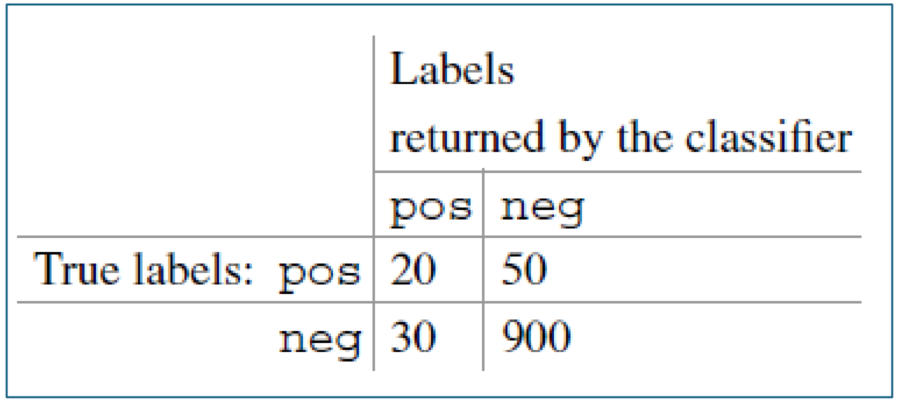
- **gmean** is indeed smaller when the two numbers are different; and the more different they are, the lower the value of gmean.

$$\sqrt{0.75 \times 0.75} = 0.75$$

$$\sqrt{0.55 \times 0.95} = 0.72$$

Other Ways to Measure Performance

- Consider the confusion matric given below. The values of **sensitivity, specificity, and gmean** are calculated as follows:



		Labels returned by the classifier	
		pos	neg
True labels:	pos	20	50
	neg	30	900

$$Se = \frac{20}{50 + 20} = 0.29$$

$$Sp = \frac{900}{900 + 30} = 0.97$$

$$\texttt{gmean} = \sqrt{\frac{20}{50 + 20} \times \frac{900}{900 + 30}} = \sqrt{0.29 \times 0.97} = 0.53$$

Other Ways to Measure Performance

- Consider the confusion matric given below. The values of **sensitivity, specificity, and gmean** are calculated as follows:

Regression Metrics

Mean Squared Error (MSE)

- **It is the average of residual (error) sum of squares (RSS)**

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{RSS}{N}$$

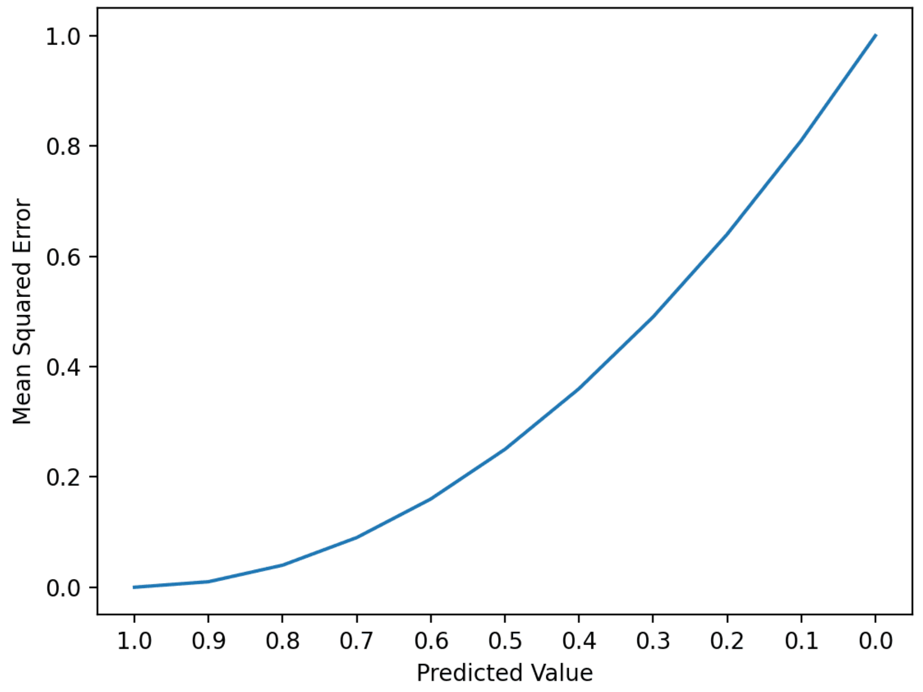
y_i → Actual value of i^{th}example

$\widehat{y}_i$ →predicted value of i^{th} example

N →Number of examples (Dataset size)

Mean Squared Error (MSE)

- **Assume that actual (expected) value is 1**. We can create a plot to get a feeling for how the change in prediction error impacts the squared error.



A line plot is created showing the curved or super-linear increase in the squared error value as the difference between the expected and predicted value is increased

Mean Squared Error (MSE)

Note :

1. MSE will never be negative since the errors are squared.
2. MSE increases exponentially with an increase in error. A good model will have an MSE value closer to zero.
3. MSE uses the mean (instead of the sum) to keep the metric **independent of the dataset size.**
4. MSE puts a significantly **heavier penalty on large errors**. This is helpful in weeding out outliers. So, it is highly sensitive to outliers.
5. MSE is an example of a scale-dependent metric, that is, the error is expressed in the units **of the underlying data.** Therefore, it cannot be used to compare the performance between different datasets.

Root Mean Squared Error (RMSE)

- **RMSE is simply the square root of the MSE**

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$$

y_i → Actual value of i^{th} example

$\widehat{y}_i$ →predicted value of i^{th} example

N →Number of examples (Dataset size)

Root Mean Squared Error (RMSE)

Note :

- The units of the RMSE are the same as the original units of the target value that is being predicted. For example, if your target variable has the units “dollars,” then the RMSE error score will also have the unit “dollars” and not “squared dollars” like the MSE.
- RMSE provides a clear understanding of the model’s performance, with lower values indicating better predictive accuracy.
- Like MSE, RMSE is dependent on the scale of the data. It increases in magnitude if the scale of the error increases.

Mean Absolute Error (MAE)

- It computes by averaging the absolute differences between predicted and actual values across the dataset. A lower MAE indicates superior model accuracy.

$$MAE = \frac{1}{N}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

y_i → Actual value of i^{th} example

$\widehat{y}_i$ →predicted value of i^{th} example

N →Number of examples (Dataset size)

Mean Absolute Error (MAE)

- The metric is expressed at the same scale as the target variable, making it easier to interpret.
- It is useful if the training data has outliers as MAE does not penalize high errors caused by outliers.
- Absolute value disregards the direction of the errors, so **underforecasting = overforecasting.**
- MAE is also **scale-dependent,** so you cannot compare it between different datasets.
- As the formula contains absolute values, MAE is not easily differentiable.
- Sometimes the large errors coming from the outliers end up being treated as the same as low errors.

Mean Absolute Error (MAE)

- The metric is expressed at the same scale as the target variable, making it easier to interpret.
- It is useful if the training data has outliers as MAE does not penalize high errors caused by outliers.
- Absolute value disregards the direction of the errors, so **underforecasting = overforecasting.**
- MAE is also **scale-dependent,** so you cannot compare it between different datasets.
- As the formula contains absolute values, MAE is not easily differentiable.
- Sometimes the large errors coming from the outliers end up being treated as the same as low errors.

Mean Absolute Percentage Error (MAPE)

- MAPE is calculated by dividing the absolute difference between the actual and predicted values by the actual value. This absolute percentage is averaged across the dataset.

$$MAPE = \frac{1}{N}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

y_i → Actual value of i^{th} example

$\widehat{y}_i$ →predicted value of i^{th} example

N →Number of examples (Dataset size)

Mean Absolute Percentage Error (MAPE)

- MAPE is asymmetric(biased) and puts a heavier penalty on negative errors (when predictions are higher than actuals) than on positive ones. This is caused by the fact that the percentage error cannot exceed 100% for forecasts that are too low. Meanwhile, there is no upper limit for forecasts that are too high. As a result, optimizing for MAPE will favor models **that underforecast rather than overforecast.**
- Therefore , due to the division operation, MAPE’s sensitivity to alterations in actual values leads to varying loss for the same error. (Thus MAPE is a relative metric)

Mean Absolute Percentage Error (MAPE)

- MAPE is expressed as a percentage, which makes it a scale-independent metric. It can be used to compare predictions on different scales. (MAPE can exceed 100%)
- So all errors are normalized on a common scale and it is easy to understand.
- MAPE is undefined when the actuals are zero (division by zero). Additionally, it can take extreme values when the actuals are very close to zero.

Root Mean Squared Log Error(RMSLE)

- Taking the log of the RMSE metric slows down the scale of error.

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1)-\log(a_i+1))^2}$$

Where:

n is the total number of observations in the data set

p_i is your prediction of target, and

a_i is the actual target for i

$\log(x)$ is the natural logarithm of x. that is $log_e(x)$

Root Mean Squared Log Error(RMSLE)

- It is a very simple metric that is used by most of the datasets hosted for Machine Learning competitions
- It is an extension on Mean Squared Error (MSE) that is mainly used when predictions have large deviations
- In the case of RMSLE the outliers are drastically scaled down therefore nullifying their effect.

  Consider, p_i = 1000, a_i = 9000

  Calculated RMSLE: 0.1053

  Calculated RMSE : 1000

Value of the **RMSE explodes in magnitude** as soon as it encounters an outlier. **RMSLE error is not affected much**

Root Mean Squared Log Error(RMSLE)

- Also, more penalty is incurred when the predicted Value is less than the Actual Value. On the other hand, Less penalty is incurred when the predicted value is more than the actual value.

Case 1: Underestimation of Actual Value

p_i = 1000 a_i = 600

RMSE Calculated: 400 RMSLE Calculated: 0.510

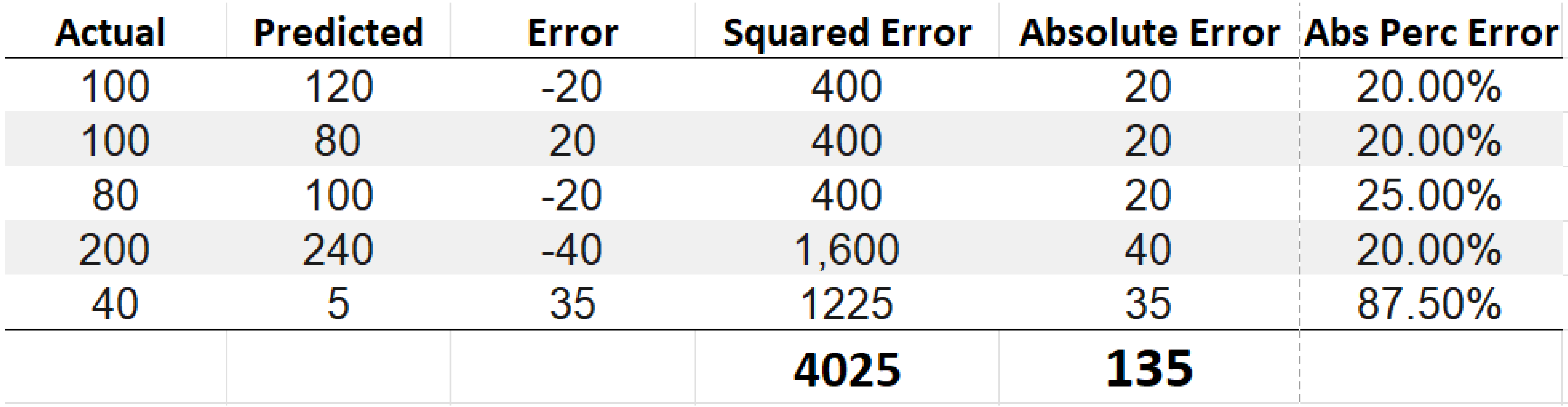
Case2: Overestimation of Actual Value

p_i = 1000 a_i = 1400

RMSE Calculated: 400 RMSLE Calculated: 0.33

An Example

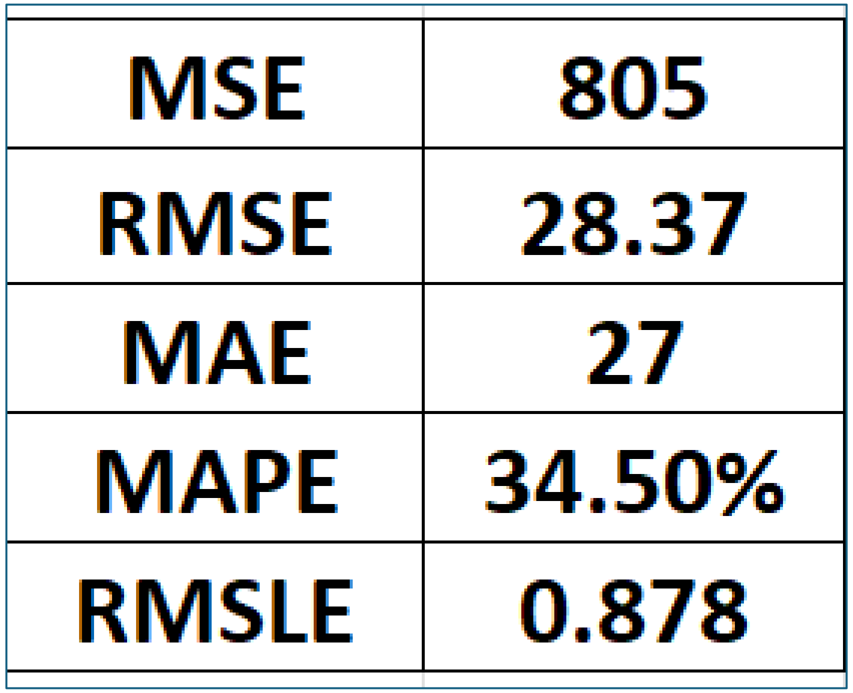
The following example contains five observations. Table 1 shows the actual values, predictions.



Actual	Predicted	Error	Squared Error	Absolute Error	Abs Perc Error
100	120	-20	400	20	20.00%
100	80	20	400	20	20.00%
80	100	-20	400	20	25.00%
200	240	-40	1,600	40	20.00%
40	5	35	1225	35	87.50%
			4025	**135**	

An Example

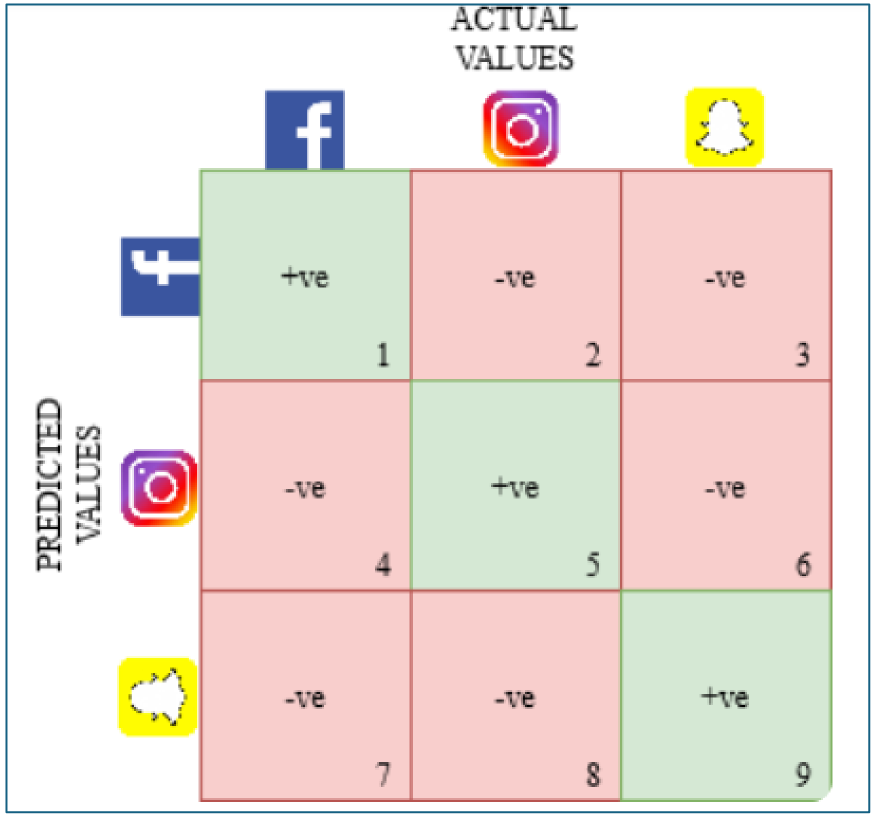
Example of calculating the performance metrics on five observations of the previous table



MSE	805
RMSE	28.37
MAE	27
MAPE	34.50%
RMSLE	0.878

Confusion Matrix for Multiclass Classification

- Let consider 3 class classification problem,. we have to predict whether a person **loves Facebook, Instagram, or Snapchat**. The confusion matrix would be a 3 x 3 matrix like this:



Facebook	Instagram	Snapchat
TP = $Cell_1$	TP = $Cell_5$	TP = $Cell_9$
FP = $Cell_2 + Cell_3$	FP = $Cell_4 + Cell_6$	FP = $Cell_7 + Cell_8$
TN = $Cell_5 + Cell_6 + Cell_8 + Cell_9$	TN = $Cell_1 + Cell_3 + Cell_7 + Cell_9$	TN = $Cell_1 + Cell_2 + Cell_4 + Cell_5$
FN = $Cell_4 + Cell_7$	FN = $Cell_2 + Cell_8$	FN = $Cell_3 + Cell_6$

Confusion Matrix for Multiclass Classification

- **Example :**

	F	I	S
F	30	10	5
I	3	20	5
S	2	10	15

Confusion Matrix for Multiclass Classification

- **Example :**

Facebook

TP = 30

FP = 10 + 5 = 15

TN = 20 + 5 + 10 + 15 = 50

FN = 3 +2 = 5

Instagram

TP = 20

FP = 3 +5 = 8

TN = 30 + 5 + 2 + 15 = 52

FN = 10 + 10 = 20

Snapchat :

TP = 15

FP = 2 +10 = 12

TN = 30 + 10 + 3 + 20 = 63

FN = 5 + 5 = 10