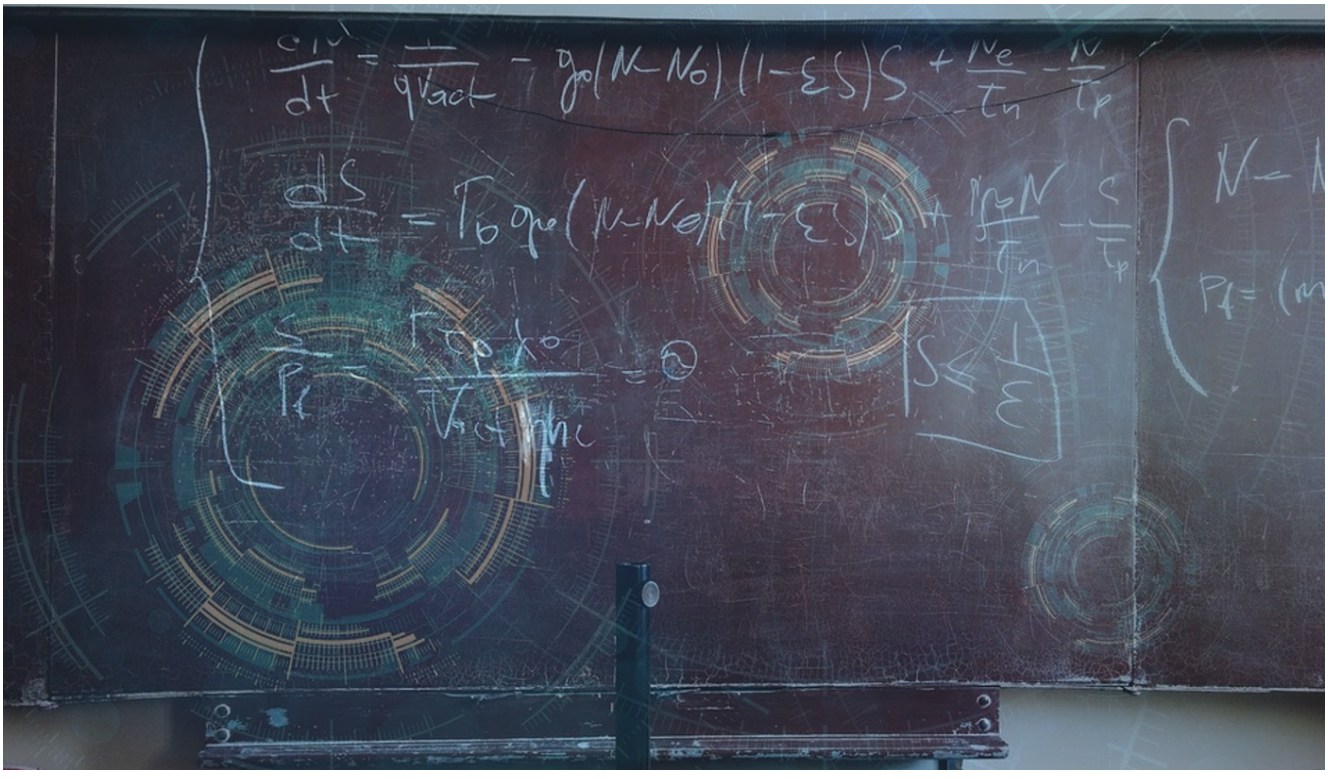
Your Ultimate Data Science Statistics & Mathematics Cheat Sheet

Machine Learning Metrics, Statistical Indicators, & More

Andre Ye



Pixabay

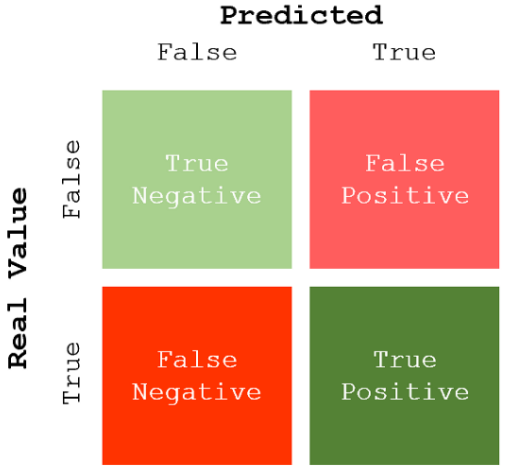
> *All images created by author unless stated otherwise.*

In data science, having a solid understanding of the statistics and mathematics of your data is essential to applying and interpreting machine learning methods appropriately and effectively.

- **Classifier Metrics.** Confusion matrix, sensitivity, recall, specificity, precision, F1 score. What they are, when to use them, how to implement them.
- **Regressor Metrics.** MAE, MSE, RMSE, MSLE, R^2. What they are, when to use it, how to implement it.
- **Statistical Indicators.** Correlation coefficient, covariance, variance, standard deviation. How to use and interpret them.
- **Types of Distributions.** The three most common distributions, how to identify them.

Classifier Metrics

Classifier metrics are metrics used to evaluate the performance of machine learning classifiers — models that put each training example into one of several discrete categories.



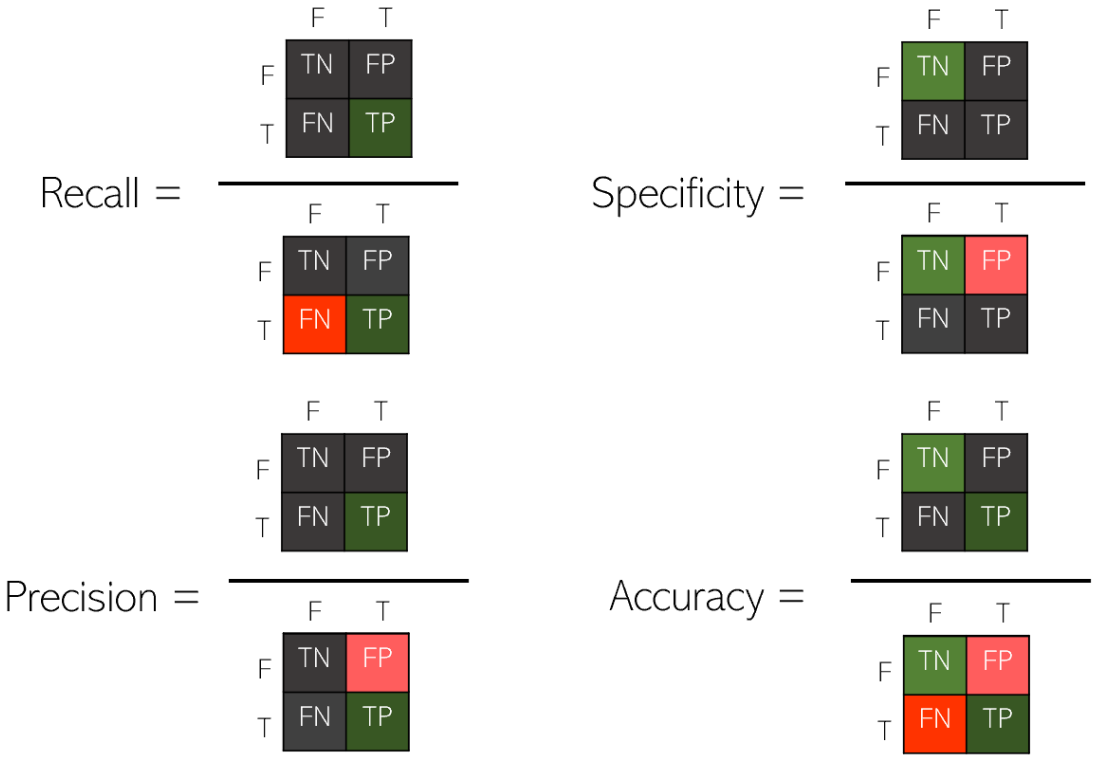
Confusion Matrix is a matrix used to indicate a classifier's predictions on labels. It contains four cells, each corresponding to one combination of a predicted true or false and an actual true or false. Many classifier

metrics are based on the confusion matrix, so it's helpful to keep an image of it stored in your mind.

Sensitivity/Recall is the number of positives that were accurately predicted. This is calculated as `TP/(TP+FN)` (note that false negatives are positives). Sensitivity is a good metric to use in contexts where correctly predicting positives is important, like medical diagnoses. In some cases, false positives can be dangerous, but it is generally agreed upon that false negatives (e.g. a diagnosis of 'no cancer' in someone who does have cancer) are more deadly. By having model maximize sensitivity, its ability to prioritize correctly classify positives is targeted.

Specificity is the number of negatives that were accurately predicted, calculated as `TN/(TN+FP)` (note that false positives are actually negatives). Like sensitivity, specificity is a helpful metric in the scenario that accurately classifying negatives is more important than classifying positives.

Precision can be thought of as the opposite of sensitivity or recall in that while sensitivity measures the proportion of actually true observations that were predicted as true, precision measures the proportion of predicted true observations that actually were true. This is measured as `TP/(TP+FP)`. Precision and recall together provide a rounded view of a model's performance.



F1 Score combines precision and recall through the harmonic mean. The exact formula for it is `(2 × precision × recall) / (precision + recall)`. The harmonic mean is used since it penalizes more extreme values, opposed to the mean, which is naïve in that it weights all errors the same.

Detection/Accuracy Rate is the number of items correctly classified, calculated as the sum of True Positives and True Negatives divided by the sum of all four confusion matrix quadrants. This accuracy rate weights positives and negatives equally, instead of prioritizing one over the other.

Using F1 Score vs Accuracy: The F1 score should be used when not making mistakes is more important (False Positives and False Negatives being penalized more heavily), whereas accuracy should be used when the

model's goal is to optimize performance. Both metrics are used based on context, and perform differently depending on the data. Generally, however, the F1-score is better for imbalanced classes (for example, cancer diagnoses, when there are vastly more negatives than positives) whereas accuracy is better for more balanced classes.

Implementing metrics follows the following format.

```
#import
from sklearn.metrics import metric_name

#create instance
metric_name(y_test, model.predict(X_test))
```

real target

predicted target

Discussed metric names in `sklearn` are:

- Confusion Matrix: `confusion_matrix`
- Sensitivity/Recall: `recall_score`
- Precision: `precision_score`
- F1 Score: `f1_score`
- Accuracy: `accuracy_score`
- Balanced Accuracy (for unevenly distributed classes): `balanced_accuracy_score`

Regressor Metrics

Regression metrics are used to measure how well a model

that puts a training example on a continuous scale, such as determining the price of a house.

Mean Absolute Error (MAE) is perhaps the most common and interpretable regression metric. MAE calculates the difference between each data point's predicted *y*-value and the real *y*-value, then averages every difference (the difference being calculated as the absolute value of one value minus the other).

Median Absolute Error is another metric of evaluating the average error. While it has the benefit of moving the error distribution lower by focusing on the middle error value, it also tends to ignore extremely high or low errors that are factored into the mean absolute error.

Mean Square Error (MSE) is another commonly used regression metric that 'punishes' higher errors more. For example, an error (difference) of 2 would be weighted as 4, whereas an error of 5 would be weighted as 25, meaning that MSE finds the difference between the two errors as 21, whereas MAE weights the difference at its face value — 3. MSE calculates the square of each data point's predicted *y*-value and real *y*-value, then averages the squares.

Root Mean Square Error (RMSE) is used to give a level of interpretability that mean square error lacks. By square-rooting the MSE, we achieve a metric similar to MAE in that it is on a similar scale, while still weighting

higher errors at higher levels.

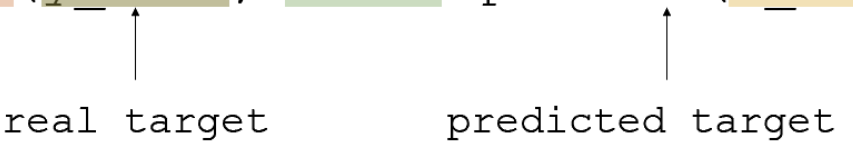
Mean Squared Logarithmic Error (MSLE) is another common variation of the mean absolute error. Because of the logarithmic nature of the error, MSLE only cares about the percent differences. This means that MSLE will treat small differences between small values (for example, 4 and 3) the same as large differences on a large scale (for example, 1200 and 900).

R^2 is a commonly used metric (where *r* is known as the correlation coefficient) which measures how much a regression model represents the proportion of the variance for a dependent variable which can be explained by the independent variables. In short, it is a good metric of how well the data fits the regression model.

Implementing metrics follows the following format.

```
#import
from sklearn.metrics import metric_name

#create instance
metric_name(y_test, model.predict(X_test))
```



Discussed metric names in `sklearn` are:

- Mean Absolute Error: `mean_absolute_error`
- Median Absolute Error: `median_absolute_error`

- Mean Squared Error: `mean_squared_error`
- Root Mean Squared Error: `root_mean_squared_error`
- Mean Squared Logarithmic Error: `mean_squared_log_error`
- R^2: `r2_score`

Statistical Indicators

Correlation Coefficient

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

Covariance

$$Cov_{xy} = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{(n-1)} = \frac{\Sigma xy - n\bar{x}\bar{y}}{(n-1)}$$

Variance

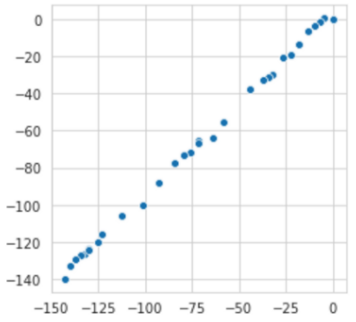
$$\sigma^2 = \frac{\sum(\chi-\mu)^2}{N}$$

Standard Deviation

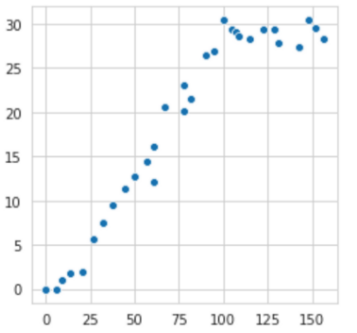
$$\sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$$

4 main data science statistical measures.

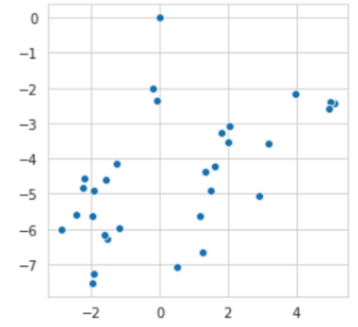
Correlation is a statistical measure of how well two variables fluctuate together. Positive correlations mean that two variables fluctuate together (a positive change in one is a positive change to another), whereas negative correlations mean that two variable change opposite one another (a positive change in one is a negative change in another). The correlation coefficient, from +1 to -1, is also known as *R*.



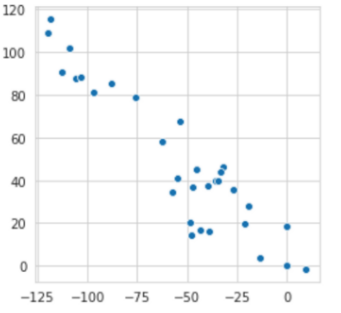
R = 0.99



R = 0.93



R = 0.54

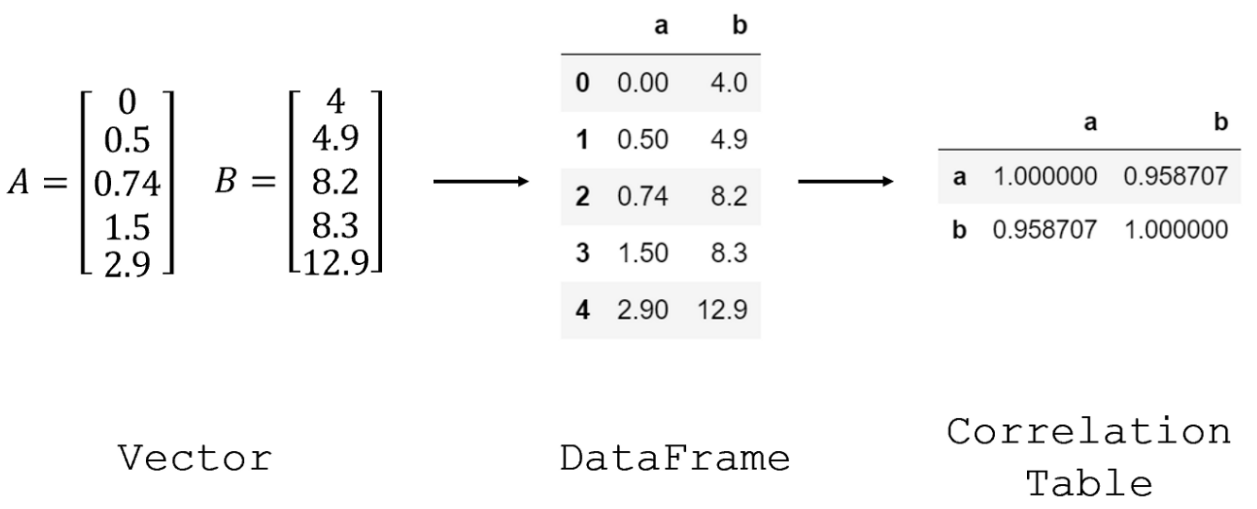


R = -0.92

The correlation coefficient can be accessed using the `.corr()` function through Pandas DataFrames. Consider the following two sequences:

```
seq1 = [0,0.5,0.74,1.5,2.9]
seq2 = [4,4.9,8.2,8.3,12.9]
```

With the constructor `table = pd.DataFrame({'a':seq1,'b':seq2})`, a DataFrame with the two sequences is created. Calling `table.corr()` yields a correlation table.



For example, in the correlation table, sequences *A* and *B* have a correlation of 0.95. The correlation table is symmetric and equal to 1 when a sequence is compared against itself.

Covariance is, similarly to correlation, a measure of the property of a function of retaining its form when the variables are linearly transformed. Unlike correlation, however, covariance can take on any number while correlation is limited between a range. Because of this, correlation is more useful for determining the strength of the relationship between two variables. Because covariance has units (unlike correlation) and is affected by changes in the center or scale, it is less widely used as a stand-alone statistic. However, covariance is used in many statistics formulas, and is a useful figure to know.

This can be done in Python with `numpy.cov(a,b)[0][1]`, where `a` and `b` are the sequences to be compared.

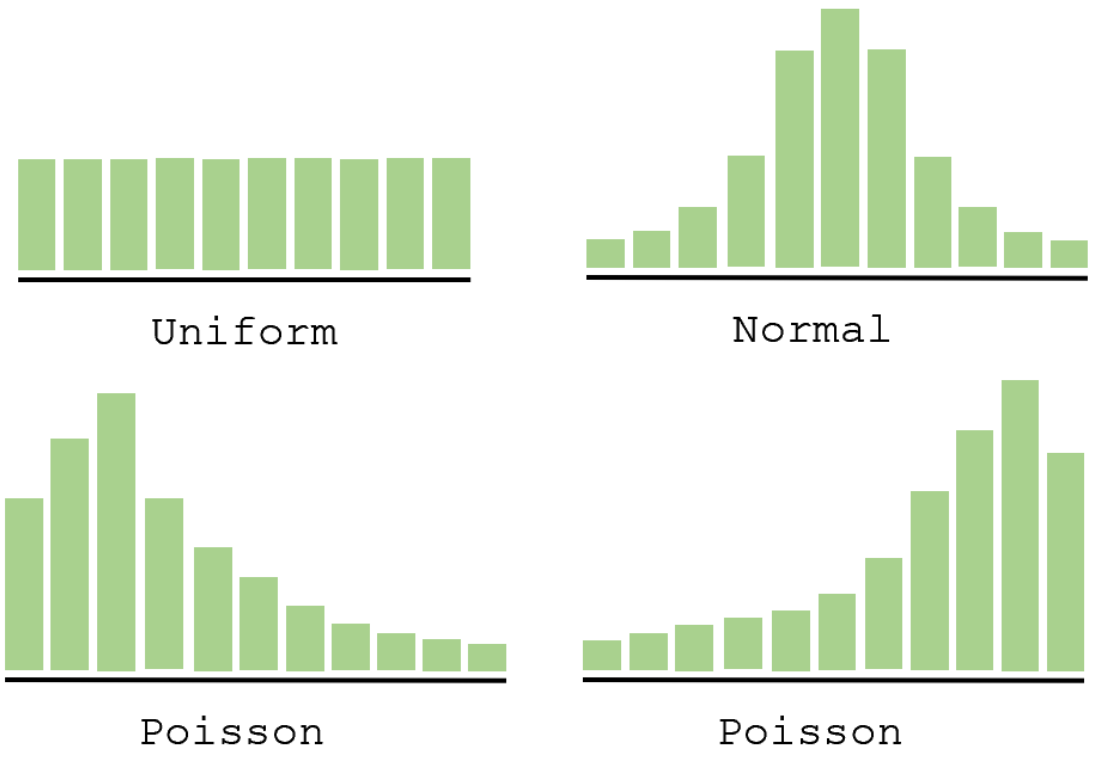
Variance is a measure of expectation of the squared

deviation of a random variable from its mean. It informally measures how far a set of numbers are spread out from their mean. Variance can be measured in the statistics library (`import statistics`) with `statistics.variance(list)`.

Standard Deviation is the square root of the variance, and is a more scaled statistic for how spread out a distribution is. Standard deviation can be measured in the statistics library with `statistics.stdev(list)`.

Types of Distributions

Knowing your distributions are very important when dealing with data analysis and understanding which statistical and machine learning methods to use.



While there are several types of mathematically

specialized distributions, most of them can fit into these three distributions.

Uniform Distribution is the easiest distribution — it is completely flat, or truly random. For example, which number of dots a die landed on (from 1 to 6) recorded for each of 6000 throws would yields a flat distribution, with approximately 1000 throws per number of dots. Uniform distributions have useful properties — for example, read how the Allied forces saved countless lives in World War II using statistical attributes of uniform distributions.

How Data Science Gave the Allied Forces an Edge in World War II

The German Tank Problem with computer simulations

Normal Distribution is a very common distribution that resembles a curve (one name for the distribution is the 'Bell Curve'). Besides its common use in data science, it is where most things in the universe can be described by, like IQ or salary. It is characterized by the following features:

- 68% of the data is within one standard deviation of the mean.
- 95% of the data is within two standard deviations of the mean.
- 99.7% of the data is within three standard deviations

of the mean.

Many machine learning algorithms require a normal distribution among the data. For example, logistic regression requires the residuals be normally distributed. This can be visualized and recognized with a `residplot()`. Information and examples of the usage of this and other statistical models can be found here.

Poisson Distribution can be thought of as a generalization of the normal distribution; a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event. With changing values of λ, the Poisson distribution shifts the mean left or right, with the capability of creating left-skewed or right-skewed data.

Thanks for reading!

If you found this cheat sheet helpful, feel free to upvote and bookmark the page for easy reference. If you enjoyed this cheat sheet, you may be interested in applying your statistics knowledge in other cheat-sheets. If you would like to see additional topics discussed in this cheat-sheet, feel free to let me know in the responses!

Your Ultimate Python Visualization Cheat-Sheet

This cheat-sheet contains the elements of a plot you will most commonly need in a clear and organized fashion, with...

Your Ultimate Data Manipulation & Cleaning Cheat Sheet

Parsing Dates, Imputing, Anomaly Detection, & More

Your Ultimate Data Mining & Machine Learning Cheat Sheet

Feature Importance, Decomposition, Transformation, & More