Confusion Matrix, Accuracy, Precision, Recall, F1 Score:

L1 : <https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd>

Choosing the right model:

L1: <https://t.co/PEBxnSsdts>

L2: <https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html>

Optimizing hyperparameters in random forest classification:

L1: <https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6>

Random Forest Classification:

L1: <https://towardsdatascience.com/random-forest-classification-678e551462f5>

Random Forest Regressor:

L1: <https://towardsdatascience.com/random-forest-regression-5f605132d19d>

Standard Scaler:

Label Encoding:

<https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/>

One Hot encoding:

L1: <https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/>

**Bootstrapping:**

Imagine you have a big jar of jellybeans and you want to know how many jellybeans of each color are in the jar. Instead of counting all the jellybeans, you can take a small scoop of jellybeans from the jar many times. Each time you count how many jellybeans of each color you got in the scoop. Then, you add up all the times you scooped each color and find the average. This gives you a good idea of how many jellybeans of each color are in the whole jar without counting all of them.

Let’s say you have a jar with 100 jellybeans in it. You don’t know how many jellybeans of each color are in the jar. So, you decide to use bootstrapping to find out.

First, you take a small scoop of 10 jellybeans from the jar. Let’s say you get 3 red, 2 blue, 2 green, 1 yellow, 1 orange and 1 purple jellybean in the scoop. You write down these numbers.

Then, you put the jellybeans back in the jar and mix them up. You take another scoop of 10 jellybeans. This time you get 4 red, 1 blue, 2 green, 2 yellow and 1 purple jellybean in the scoop. You write down these numbers too.

You repeat this process a few more times and get the following results:

Scoop 1: 3 red, 2 blue, 2 green, 1 yellow, 1 orange and 1 purple Scoop 2: 4 red, 1 blue, 2 green, 2 yellow and 1 purple Scoop 3: 2 red, 3 blue, 1 green, 2 yellow and 2 purple Scoop 4: 3 red, 2 blue, 3 green and 2 purple

Now you add up all the times you scooped each color and find the average:

Red: (3 + 4 + 2 +3) /4 = **3** Blue: (2 +1 +3 +2) /4 = **2** Green: (2 +2 +1 +3) /4 = **2** Yellow: (1 +2 +2) /4 = **1.25** Orange: (1) /4 = **0.25** Purple: (1 +1 +2 +2) /4 = **1.5**

So based on your bootstrapping results you can estimate that there are about **30 red**, **20 blue**, **20 green**, **12.5 yellow**, **2.5 orange** and **15 purple** jellybeans in the jar.

I multiplied by 10 because each scoop had 10 jellybeans and the jar had 100 jellybeans in total. So, if we estimate that there are 3 red jellybeans in a scoop of 10 jellybeans, then we can estimate that there are 3 \* 10 = **30 red jellybeans** in the whole jar of 100 jellybeans. Similarly, for the other colors.

**Random Forest Regressor:**

A RandomForestRegressor is like a group of friends trying to guess how many candies are in a jar. Each friend makes a guess based on what they know. Some friends might be really good at guessing because they have seen lots of candy jars before. Other friends might not be as good at guessing but they still try their best.

Once all the friends have made their guesses, they get together and talk about their guesses. They decide to take the average of all their guesses to come up with a final guess. This final guess is usually better than any one friend’s guess because it takes into account what all the friends know.

In the same way, a RandomForestRegressor model uses many decision trees (the friends) to make predictions (guesses) about how much something will be (like the number of candies in a jar). Each decision tree makes a prediction based on the data it has seen. Then, the model takes the average of all the predictions to come up with a final prediction.

**Random Forest Classifier:**

A Random Forest Classifier is like a group of friends trying to guess what kind of fruit is in a bag without looking inside. Each friend makes a guess based on what they know. For example, one friend might shake the bag and guess it’s an apple because it sounds like one. Another friend might smell the bag and guess it’s a banana because it smells like one.

Once all the friends have made their guesses, they get together and talk about their guesses. They decide to pick the fruit that most friends guessed as their final guess. This final guess is usually better than any one friend’s guess because it takes into account what all the friends know.

In the same way, a Random Forest Classifier model uses many decision trees (the friends) to make predictions (guesses) about what something is (like the fruit in the bag). Each decision tree makes a prediction based on the data it has seen. Then, the model picks the prediction that most decision trees made as its final prediction.

**Meaning of parameter:**

* n\_estimators: This is like telling the machine how many friends to ask for help when making a guess. The more friends it asks, the better its guess might be.
* criterion: This is like telling the machine what kind of math to use when making a guess. It can use different kinds of math, like subtracting or taking the average, to figure out how good its guess is.
* max\_depth: This is like telling the machine how many questions it can ask when making a guess. The more questions it asks, the better its guess might be.
* max\_features: This is like telling the machine how many candies to look at when making a guess. The more candies it looks at, the better its guess might be.
* bootstrap: This is like telling the machine whether or not to use the bootstrapping method we talked about earlier when making a guess.
* max\_samples: If the machine is using bootstrapping, this is like telling it how many candies to scoop out of the jar each time it makes a guess.

Column Transformer:

L1: <https://towardsdatascience.com/improve-your-data-preprocessing-with-columntransformer-and-pipelines-b6ff7edd2f77>

Fit and Fit\_Transform method:

L1: <https://towardsdatascience.com/what-and-why-behind-fit-transform-vs-transform-in-scikit-learn-78f915cf96fe>

L2: [Sklearn fit() vs transform() vs fit\_transform() – What’s the Difference? – Be on the Right Side of Change (finxter.com)](https://blog.finxter.com/sklearn-fit-vs-transform-vs-fit_transform-whats-the-difference/)

Tidbit:

1. If you have structured data, used ensemble methods

2. If you have unstructured data, use deep learning or transfer learning

Different names for:

* X = features, features variables, data
* y = labels, targets, target variables

Difference between structured and unstructured data

In machine learning, structured data refers to data that is organized into a specific format or schema, such as a table with rows and columns. Structured data can be easily searched, analyzed, and organized because it follows a predefined structure. Examples of structured data include dates, names, addresses, and credit card numbers. Structured data is used in machine learning algorithms because its specific and organized architecture eases manipulation and querying of data¹.

On the other hand, unstructured data refers to data that does not have a specific format or structure. It can include various types of data such as text, images, audio, and video. Unstructured data is more difficult to search, analyze, and organize because it does not follow a predefined structure. Unstructured data is used in natural language processing (NLP) and text mining¹.

In summary, the main difference between structured and unstructured data in machine learning is the level of organization and the ease of manipulation. Structured data is highly organized and easily used by machine learning algorithms while unstructured data is less organized and more difficult to use.

Source: Conversation with Bing, 27/6/2023

1. Structured vs Unstructured Data – What's the Difference?. <https://www.g2.com/articles/structured-vs-unstructured-data>.
2. Structured vs. Unstructured Data: What’s the Difference? | IBM. <https://www.ibm.com/cloud/blog/structured-vs-unstructured-data>.
3. Difference between Structured data and Unstructured data. <https://www.javatpoint.com/structured-data-vs-unstructured-data>.
4. Structured vs. Unstructured Data: What’s the Difference?. <https://www.ibm.com/blog/structured-vs-unstructured-data-whats-the-difference/>.
5. Structured vs Unstructured Data: What’s the Difference? - MonkeyLearn. <https://monkeylearn.com/blog/structured-data-vs-unstructured-data/>.

Overfitting and Underfitting

Overfitting and underfitting are two common challenges that can arise when training machine learning models.

Overfitting occurs when a model is trained too well on the training data, to the point where it starts to memorize the training data instead of learning to generalize to new data. This can result in a model that performs very well on the training data but performs poorly on new, unseen data. In other words, the model is too complex and fits the training data too closely.

Underfitting, on the other hand, occurs when a model is not complex enough to capture the underlying patterns in the data. This can result in a model that performs poorly on both the training data and new, unseen data. In other words, the model is too simple and does not fit the data well enough.

In summary, overfitting and underfitting represent two different types of errors that can occur when training machine learning models. Overfitting occurs when a model is too complex and memorizes the training data, while underfitting occurs when a model is too simple and fails to capture the underlying patterns in the data.

Source: Conversation with Bing, 27/6/2023

1. ML | Underfitting and Overfitting - GeeksforGeeks. <https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>.
2. Overfitting and Underfitting in Machine Learning - Javatpoint. <https://www.javatpoint.com/overfitting-and-underfitting-in-machine-learning>.
3. Overfitting vs. Underfitting: What Is the Difference?. <https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/>.
4. Overfitting in Machine Learning: What It Is and How to Prevent It. <https://elitedatascience.com/overfitting-in-machine-learning>.

Gini Impurity :

Gini impurity is a measure of how mixed a set of items is. It is used in decision trees to help decide which feature to split on at each step in the tree.

Imagine you have a bag of marbles and you want to know how mixed up the colors are. If all the marbles are the same color, then the bag is not mixed up at all and the Gini impurity is 0. But if there are many different colors of marbles in the bag, then it is very mixed up and the Gini impurity is higher.

In a decision tree, we use Gini impurity to help us decide which feature to split on. We want to choose the feature that will make the groups of items as unmixed as possible. For example, if we are trying to decide whether an animal is a cat or a dog, we might split on the feature "Does it bark?". If all the animals that bark are dogs and all the animals that don't bark are cats, then this would be a good split because it makes the groups very unmixed.

I hope that helps explain what Gini impurity is! Do you have any other questions?

Source: Conversation with Bing, 27/6/2023

(1) Gini Impurity – LearnDataSci. https://www.learndatasci.com/glossary/gini-impurity/.

(2) ML | Gini Impurity and Entropy in Decision Tree - GeeksforGeeks. <https://www.geeksforgeeks.org/gini-impurity-and-entropy-in-decision-tree-ml/>.

(3) Understanding the maths behind the Gini impurity method for decision .... <https://analyticsindiamag.com/understanding-the-maths-behind-the-gini-impurity-method-for-decision-tree-split/>.

(4) Explaining the Gini Impurity with Examples in Python. <https://insidelearningmachines.com/gini_impurity/>.

Examples:

Sure! Let’s say we have a group of 30 fruits, 10 of which are apples, 10 of which are oranges, and 10 of which are bananas. We want to use a decision tree to classify these fruits based on their features.

One feature we could use is “Is it round?”. If we split the group based on this feature, we would end up with two groups: one group of fruits that are round (apples and oranges) and one group of fruits that are not round (bananas).

We can calculate the Gini impurity for each group to see how mixed up they are. For the group of fruits that are round, the probability of randomly picking an apple is 10/20 = 0.5, the probability of randomly picking an orange is 10/20 = 0.5, and the probability of randomly picking a banana is 0/20 = 0. The Gini impurity for this group is calculated as:

1 - (0.5^2 + 0.5^2 + 0^2) = 0.5

Copy

For the group of fruits that are not round, the probability of randomly picking an apple is 0/10 = 0, the probability of randomly picking an orange is 0/10 = 0, and the probability of randomly picking a banana is 10/10 = 1. The Gini impurity for this group is calculated as:

1 - (0^2 + 0^2 + 1^2) = 0

Copy

Since the first group has a Gini impurity of 0.5, this means that it is somewhat mixed up. The second group has a Gini impurity of 0, which means it is not mixed up at all.

We could also try splitting on another feature, such as “Is it yellow?”. If we split the group based on this feature, we would end up with two groups: one group of fruits that are yellow (bananas) and one group of fruits that are not yellow (apples and oranges).

For the group of fruits that are yellow, the probability of randomly picking an apple is 0/10 = 0, the probability of randomly picking an orange is 0/10 = 0, and the probability of randomly picking a banana is 10/10 = 1. The Gini impurity for this group is calculated as:

1 - (0^2 + 0^2 + 1^2) = 0

Copy

For the group of fruits that are not yellow, the probability of randomly picking an apple is 10/20 = 0.5, the probability of randomly picking an orange is 10/20 = 0.5, and the probability of randomly picking a banana is 0/20 = 0. The Gini impurity for this group is calculated as:

1 - (0.5^2 + 0.5^2 + 0^2) = 0.5

Copy

In this case, both splits result in one group with a Gini impurity of 0 and one group with a Gini impurity of 0.5.

predict and predict\_proba in machine learning

predict is a method that takes in new data and returns a prediction for the class or value of that data. For example, if we have a machine learning model that can predict whether an email is spam or not, we can use the predict method to make a prediction for a new email. The predict method will return either “spam” or “not spam” based on the model’s prediction.

predict\_proba, on the other hand, returns the probability estimates for each class or value. In the case of our spam email example, predict\_proba would return the probability that the email is spam and the probability that it is not spam. These probabilities can be useful when we want to understand how confident the model is in its prediction.

In summary, predict returns a prediction for the class or value of new data, while predict\_proba returns the probability estimates for each class or value.

what does the score method do in machine learning

The score method in machine learning is used to evaluate the performance of a model. It takes in a set of test data and the true labels or values for that data and returns a score that represents how well the model performed on the test data.

The specific way that the score method calculates the performance of the model depends on the type of model and the type of problem it is being used to solve. For example, in a classification problem, the score method might return the accuracy of the model, which is the percentage of test data that the model correctly classified. In a regression problem, the score method might return the R-squared value, which measures how well the model fits the test data.

In summary, the score method is used to evaluate the performance of a machine learning model on a set of test data.

Explain me in detail the difference between score method , scoring parameter and cross\_val\_score

Let’s take a closer look at the difference between the score method, the scoring parameter, and the cross\_val\_score function in machine learning.

The score method is a method that is available on many machine learning models in scikit-learn. It is used to evaluate the performance of the model on a set of test data. The specific way that the score method calculates the performance of the model depends on the type of model and the type of problem it is being used to solve.

The scoring parameter is an argument that can be passed to many model evaluation functions in scikit-learn, including cross\_val\_score. It specifies the metric that should be used to evaluate the performance of the model. For example, if we want to use accuracy as our evaluation metric, we can set scoring='accuracy'.

The cross\_val\_score function is a function in scikit-learn that can be used to evaluate the performance of a model using cross-validation. Cross-validation is a technique for evaluating the performance of a model by splitting the data into multiple subsets and training and testing the model on each subset. The cross\_val\_score function takes in a model, data, and labels, and returns an array of scores representing the performance of the model on each subset of the data.

In summary, the score method is used to evaluate the performance of a machine learning model on a set of test data. The scoring parameter specifies the metric that should be used to evaluate the performance of the model. The cross\_val\_score function is used to evaluate the performance of a model using cross-validation.

Pipelining

cross\_val\_score(clf, X, y, cv=5)

Explain me this code line by line

cross\_val\_score(clf, X, y, cv=5)

This line of code is calling the cross\_val\_score function from scikit-learn. This function is used to evaluate the performance of a machine learning model using cross-validation.

The first argument to the function, clf, is the machine learning model that we want to evaluate. This could be any type of model that has a fit and predict method, such as a decision tree or a support vector machine.

The second and third arguments, X and y, are the data and labels that we want to use to evaluate the model. X is a two-dimensional array where each row represents a data point and each column represents a feature. y is a one-dimensional array where each element represents the label or value for the corresponding data point in X.

The fourth argument, cv=5, specifies the number of folds to use in the cross-validation process. In this case, we are using 5-fold cross-validation, which means that the data will be split into 5 subsets and the model will be trained and tested on each subset.

In summary, this line of code is calling the cross\_val\_score function to evaluate the performance of a machine learning model using 5-fold cross-validation on the given data and labels.

Classification model evaluation metrics

Accuracy

Area under ROC curve

Confusion matrix

Classification report

* **Accuracy**: Accuracy is the most common metric for evaluating classification models. It is calculated as the number of correct predictions divided by the total number of predictions. In other words, it measures how often the model correctly predicts the class of a data point. The formula for accuracy is:

Accuracy = (True Positives + True Negatives) / (True Positives + False Positives + True Negatives + False Negatives)

For example, let’s say we have a model that can predict whether an email is spam or not. We test the model on 100 emails and it correctly identifies 90 of them as spam or not spam. The accuracy of the model would be 90/100 = 0.9 or 90%.

* **Area under ROC curve**: The ROC (Receiver Operating Characteristic) curve is a plot that shows the performance of a binary classification model at different thresholds. The area under the ROC curve (AUC) is a measure of how well the model can distinguish between the two classes. An AUC of 1.0 means that the model can perfectly distinguish between the two classes, while an AUC of 0.5 means that the model is no better than random guessing.

For example, let’s say we have a model that can predict whether a person has a certain disease or not based on their symptoms. We can plot the ROC curve for this model by changing the threshold for what counts as a positive prediction and calculating the true positive rate and false positive rate at each threshold. The AUC is then calculated as the area under this curve.

* **Confusion matrix**: A confusion matrix is a table that shows the number of true positives, false positives, true negatives, and false negatives for a binary classification model. It can be used to calculate many other evaluation metrics, such as precision and recall.

For example, let’s say we have a model that can predict whether an email is spam or not. We test the model on 100 emails and get the following results:

* + True positives (spam emails correctly identified as spam): 20
  + False positives (non-spam emails incorrectly identified as spam): 10
  + True negatives (non-spam emails correctly identified as non-spam): 60
  + False negatives (spam emails incorrectly identified as non-spam): 10

The confusion matrix for this model would look like this:

|--------------|--------------|

| True Positives | False Positives |

|--------------|--------------|

| False Negatives | True Negatives |

|--------------|--------------|

Copy

* **Classification report**: A classification report is a summary of several evaluation metrics for a classification model. It typically includes metrics such as precision, recall, and f1-score for each class, as well as overall averages.

For example, let’s say we have a model that can predict whether an email is spam or not. We test the model on 100 emails and get the following results:

* + Precision (how many of the emails identified as spam were actually spam): 0.67
  + Recall (how many of the actual spam emails were identified as spam): 0.67
  + F1-score (a measure that combines precision and recall): 0.67

The classification report for this model would look something like this:

|--------------|--------------|--------------|--------------|

| Class | Precision | Recall | F1-score |

|--------------|--------------|--------------|--------------|

| Spam | 0.67 | 0.67 | 0.67 |

| Not Spam | ... | ... | ... |

|--------------|--------------|--------------|--------------|

AUC ROC Curve

<https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

Explain me sensitivity and specificity

* **Sensitivity**, also known as the true positive rate or recall, measures the proportion of actual positives that are correctly identified by the model. In other words, it measures how well the model can correctly identify positive instances. The formula for sensitivity is:

Sensitivity = True Positives / (True Positives + False Negatives)

For example, let’s say we have a model that can predict whether a person has a certain disease or not. We test the model on 100 people and it correctly identifies 80 of them as having the disease. The sensitivity of the model would be 80/100 = 0.8 or 80%.

* **Specificity**, also known as the true negative rate, measures the proportion of actual negatives that are correctly identified by the model. In other words, it measures how well the model can correctly identify negative instances. The formula for specificity is:

Specificity = True Negatives / (True Negatives + False Positives)

For example, let’s say we have a model that can predict whether a person has a certain disease or not. We test the model on 100 people and it correctly identifies 90 of them as not having the disease. The specificity of the model would be 90/100 = 0.9 or 90%.

Regression model evaluation metrics

R^2 (pronounced r-squared) or coefficient of determination

Mean absolute error (MAE)

Mean squared error (MSE)

Explain me each one in detail with example

* **R-squared**: R-squared, also known as the coefficient of determination, is a measure of how well a regression model fits the data. It is calculated as the proportion of the variance in the dependent variable that is predictable from the independent variables. The formula for R-squared is:

R-squared = 1 - (Sum of Squared Residuals / Total Sum of Squares)

For example, let’s say we have a model that can predict the price of a house based on its size and location. We test the model on some data and calculate the R-squared value to be 0.8. This means that 80% of the variance in the house prices can be explained by the size and location of the houses.

* **Mean absolute error (MAE)**: Mean absolute error is a measure of how close the predictions of a regression model are to the true values. It is calculated as the average absolute difference between the predicted values and the true values. The formula for MAE is:

MAE = (1/n) \* sum(|y\_true - y\_pred|)

For example, let’s say we have a model that can predict the price of a house based on its size and location. We test the model on 10 houses and calculate the MAE to be $50,000. This means that, on average, the model’s predictions are $50,000 away from the true house prices.

* **Mean squared error (MSE)**: Mean squared error is another measure of how close the predictions of a regression model are to the true values. It is calculated as the average squared difference between the predicted values and the true values. The formula for MSE is:

MSE = (1/n) \* sum((y\_true - y\_pred)^2)

For example, let’s say we have a model that can predict the price of a house based on its size and location. We test the model on 10 houses and calculate the MSE to be $5,000,000,000. This means that, on average, the squared difference between the model’s predictions and the true house prices is $5,000,000,000.

sklearn.metrics.ConfusionMatrixDisplay.from\_estimator(estimator, X, y) - this takes a fitted estimator (like our clf model), features (X) and labels (y), it then uses the trained estimator to make predictions on X and compares the predictions to y by displaying a confusion matrix.

sklearn.metrics.ConfusionMatrixDisplay.from\_predictions(y\_true, y\_pred) - this takes truth labels and predicted labels and compares them by displaying a confusion matrix.

explain me the difference between these two

They can be used to create and display a confusion matrix for a binary classification model. Here’s the difference between these two methods:

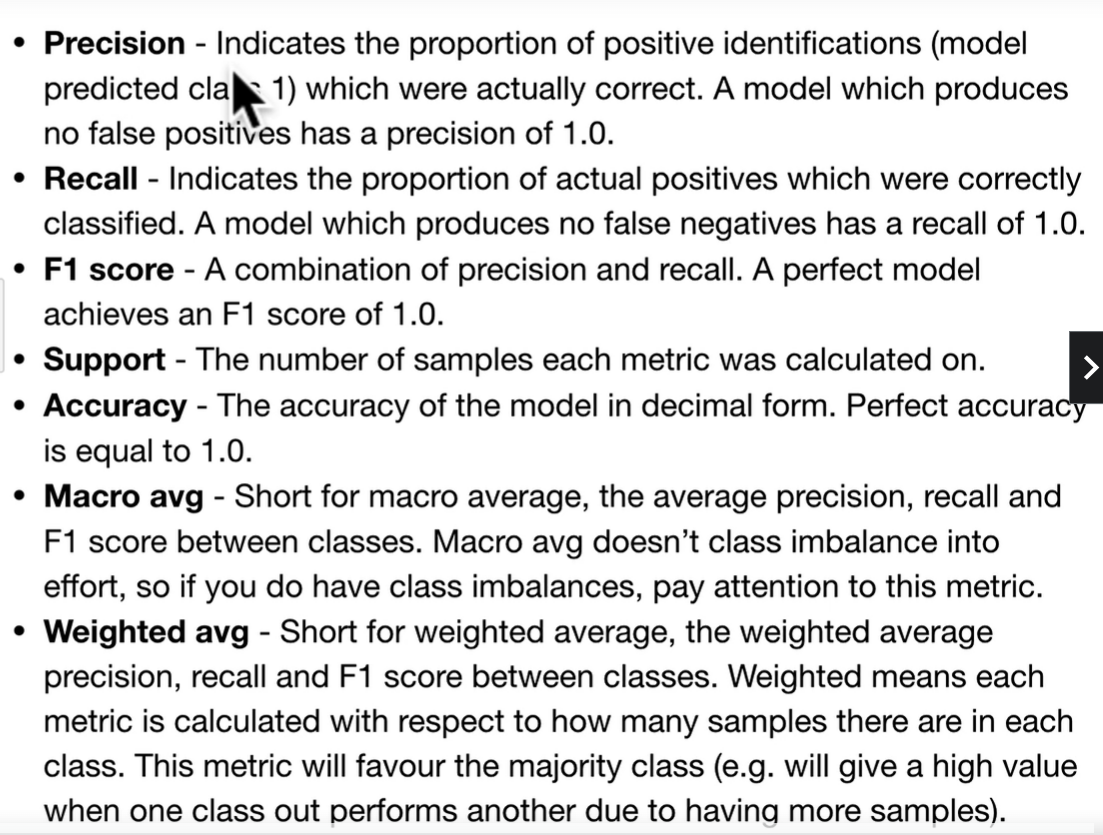
sklearn.metrics.ConfusionMatrixDisplay.from\_estimator(estimator, X, y) takes a fitted estimator (like our clf model), features (X) and labels (y). It then uses the trained estimator to make predictions on X and compares the predictions to y by displaying a confusion matrix. In other words, this method takes a trained model and some test data and creates a confusion matrix based on the model’s predictions on the test data.

sklearn.metrics.ConfusionMatrixDisplay.from\_predictions(y\_true, y\_pred) takes truth labels (y\_true) and predicted labels (y\_pred) and compares them by displaying a confusion matrix. In other words, this method takes the true labels for some test data and the predicted labels for that data (which could have been generated by any model or method) and creates a confusion matrix based on these labels.

The main difference between these two methods is that from\_estimator takes a trained model and some test data and generates the predicted labels using the model, while from\_predictions takes the true labels and predicted labels directly as arguments.

Difference between pickle and joblib

<https://medium.com/nlplanet/is-it-better-to-save-models-using-joblib-or-pickle-776722b5a095#:~:text=TLDR%3A%20joblib%20is%20faster%20in,)%2C%20joblib%20should%20be%20faster>.



I want to understand the logspace function in numpy

The numpy.logspace function in the NumPy library returns an array of numbers spaced evenly on a log scale. In linear space, the sequence starts at base \*\* start and ends at base \*\* stop. The function takes several parameters, including start, stop, num, endpoint, base, dtype, and axis. [The default base is 10.0, and the default number of samples to generate is 501](https://numpy.org/doc/stable/reference/generated/numpy.logspace.html).

Here’s an example of how to use the function:

import numpy as np

# Create an array of 4 numbers spaced evenly on a log scale between 10^2 and 10^3

arr = np.logspace(2.0, 3.0, num=4)

print(arr)

# Output: [ 100. , 215.443469 , 464.15888336, 1000. ]

Logistic Regression

L1: <https://medium.com/analytics-vidhya/understanding-logistic-regression-b3c672deac04>

L2: <https://medium.com/codex/machine-learning-logistic-regression-with-python-5ed4ded9d146>

HyperParameter tuning

L1: <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>

KNN

<https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/>