

Statistical Classification

≡ Week	TUES. Week 4
≡ Assignment Due	
☑ Assignment Done	<input type="checkbox"/>
📅 Due Date	
☑ Notes Done	<input checked="" type="checkbox"/>

Presentation

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/5bf204cd-196d-4318-8d52-5d3466f73c1a/L1-ProbabilisticClassification.pdf>

Class Notes

By the end of Thursday, we should have the knowledge necessary for our second homework assignment.

Currently, we've used LDA to build a classifier, inspired by PCA (not used often) and KNN, an intuitive algorithm for deciding a class label or assign an observation based on its neighbors in the training set.

- Now, we'll go into the fundamental algorithms in ML using prob. and stats to make decisions.
 - **NOTE:** Review Prob/Stats Week 0 slides!

Statistical Learning

Inference

Our statistical learning will begin with the concept of **inference**, looking at the probability and statistics of our *labeled data* to make predictions on the new data.

- We ask: Given the distribution of *seen data*, what can we *infer* about the new data?
- We typically say the observable data we see is the *evidence/features* $\mathbf{x} = [x_1, x_2, \dots, x_D]$.
 - We want to know what the *probability* is that this object came from class $i \rightarrow$ referred to as the **posterior probability**
 - Overall goal: Posterior probability $\rightarrow P(y = i | \text{feature}_1 = x_1, \text{feature}_2 = x_2, \dots, \text{feature}_D = x_D)$
 - In English, this is read as the probability of class i given the *evidence* (given that we have the feature 1 equaling x_1 and feature 2 equaling x_2 , etc.).
 - Another way to describe this: Given the first feature has value x_1 , the second has x_2 , etc... what is the probability that our class was i ?
- We can use several rules in stats to write this. We can say:
 - Given the first feature equals $x_1 \dots$ feature D equals x_D , what is the probability that a class y equals i ?

$$P(y = i | f_1 = x_1, \dots, f_D = x_D) = \frac{P(y = i, f_1 = x_1, \dots, f_D = x_D)}{P(f_1 = x_1, \dots, f_D = x_D)}$$

- The **numerator** is the probability of our evidence and our class happening *at the same time* → referred to as **joint probability**.
- The **denominator** is the probability of our evidence.
- RECALL: $P(a, b, c)$ is called the **joint probability** and tells us the probability of these things happening at the same time.
- With this posulation of the probability, we can compute the posterior purely in terms of this *joint probability*.
 - Given enough data, we should be able to get the joints easily!

Joint Probability

How to make a joint distribution:

1. Make a truth table listing *all combinations* of values of your variables (if there are M Boolean variances, then the table will have 2^M rows.
 - a. We want to create ALL possible combinations of the evidence.
2. Count how many times in your data each combination occurs.
3. Normalize these counts by the *total data size (total number of observations)* in order to arrive at probabilities.
 - a. NOTE: The sum of these joint probabilities MUST be equal to 1.

Once we have the joint distribution table, it is easy to compute various values such as the probability of a single variable:

- We can look for the entries that contain a variable and the probability of the variable will be the sum of the entries that contain that variable.

$$P(row) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

- NOTE: Again, note that all of the probabilities from the rows should sum to 1.

Example

With these values, we can reasonably compute the probability of any logical expression using the joint distribution table.

gender	hours_worked	wealth		
Female	v0:40.5-	poor	0.253122	<div></div>
		rich	0.0245895	<div></div>
	v1:40.5+	poor	0.0421768	<div></div>
		rich	0.0116293	<div></div>
Male	v0:40.5-	poor	0.331313	<div></div>
		rich	0.0971295	<div></div>
	v1:40.5+	poor	0.134106	<div></div>
		rich	0.105933	<div></div>

One useful law of probability to help calculate these probabilities is the law of total probability:

$$P(Y) = \sum_i P(Y \cap x_i) = \sum_{\text{rows with } Y} P(\text{row})$$

- Examples:

- What is $P(\text{wealth} = \text{Poor})$?
 - $P(\text{wealth} = \text{Poor}) = P(\text{female}, v0 : 40.5-, \text{poor}) + P(\text{female}, v0 : 40.5+, \text{poor}) + P(\text{male}, v0 : 40.5-, \text{poor}) + P(\text{male}, v0 : 40.5+, \text{poor})$
 - Rows with $\text{wealth} = \text{Poor}$
 - $P(\text{wealth} = \text{Poor}) = 0.253122 + 0.0421768 + 0.331313 + 0.134106 = 0.7606178$
- What is $P(\text{wealth} = \text{Poor}, \text{gender} = \text{Male})$?
 - $P(\text{wealth} = \text{Poor}, \text{gender} = \text{Male}) = P(\text{poor}, \text{male}, v0 : 40.5-) + P(\text{poor}, \text{male}, v0 : 40.5+)$
 - Rows with $\text{wealth} = \text{Poor}, \text{gender} = \text{Male}$
 - $P(\text{wealth} = \text{Poor}, \text{gender} = \text{Male}) = 0.331313 + 0.134106 = 0.465419$

We can also easily compute joint/conditional probabilities using the joint distributions:

- Example: $P(\text{gender} = \text{Male} | \text{wealth} = \text{poor}) = \frac{P(\text{gender}=\text{male}, \text{wealth}=\text{poor})}{P(\text{wealth}=\text{poor})} = \frac{P(M,P)}{P(P)}$
 - $P(\text{gender} = \text{Male} | \text{wealth} = \text{poor}) = \frac{0.465419}{0.7606178} = 0.611896014$
 - Prob. of someone being male and poor divided by probability that someone was poor in general.
 - Prob. of someone being male and poor $\rightarrow P(\text{wealth} = \text{Poor}, \text{gender} = \text{Male}) = 0.465419$

- Prob. of someone being poor $\rightarrow P(\text{wealth} = \text{Poor}) = 0.7606178$
- Calculated above
- In general, these conditional probabilities can be calculated as $\frac{P(y, x)}{P(x)}$, where $P(x) = \sum_{i=1}^K P(y, x_i)$

Example: Suppose we wanted to figure out, given gender and hours worked, what the wealth is?

- What is $P(W = \text{rich} | G = \text{female}, H = 40.5) = \frac{0.024}{0.024 + 0.253} = \frac{P(\text{rich}, \text{female}, 40.5)}{P(\text{female}, 40.5)}$

Another example: Given gender and wealth, what were the hours worked?

- What is $P(H = 40.5 | G = \text{male}, W = \text{rich}) = \frac{0.097}{0.1 + 0.09}$

Using Inference for Classification

Inference is a big deal \rightarrow we use it all the time! (e.g.: The lights are out and it's 9pm. What is the likelihood that my spouse is asleep?)

- Question: How can we use this for classification?
 - Given an observation \mathbf{x} with D features, we can compute the posterior for each class.
 - We'll write the posterior shorthand as $P(y = i | \mathbf{x})$ and the joint as $P(y = i, \mathbf{x})$
 - To figure out which class a set of features should belong to, we can just choose the class that *maximizes* the posterior probability:

$$\begin{aligned}\hat{y} &= \underset{i}{\operatorname{argmax}} P(y = i | \mathbf{x}) \\ &= \underset{i}{\operatorname{argmax}} \left(\frac{P(y = i, \mathbf{x})}{P(\mathbf{x})} \right)\end{aligned}$$

- If all we care about is the *largest posterior*, then we don't have to compute the denominator of this above equation, which is the probability of the *evidence*.
 - This value is class-independent \rightarrow there is nothing about y in that computation, in some sense it is a normalizing factor.
 - With this in mind, we ultimately only end up caring about which numerator gives us the highest value.

$$\hat{y} = \underset{i}{\operatorname{argmax}} P(y = i, \mathbf{x})$$

- If after the fact, we're interested in the *true probability*, since we have $P(y = i, \mathbf{x})$ for all classes i , we can get this from all of our numerators by dividing by the sum of the joint probabilities:
 - We let this value be referred to as ρ , which is calculated as $\rho = \sum_{k=1}^K P(y = k, \mathbf{x})$
 - The $P(\mathbf{x})$ will simply be equal to the sum of all of the classes given some set of features.
 - Then, we can compute $P(y = i | \mathbf{x})$ as:

$$P(y = i | \mathbf{x}) = \frac{P(y = i, \mathbf{x})}{\rho}$$

Examples: Given a rich male, let's classify them as having worked more or less than 40.5 hours per week.

- $P(\text{hours} = 40.5+ | \text{gender} = \text{male}, \text{wealth} = \text{rich})$
 $\propto P(\text{hours} = 40.5+, \text{gender} = \text{male}, \text{wealth} = \text{rich})$
- $P(\text{hours} = 40.5- | \text{gender} = \text{male}, \text{wealth} = \text{rich})$
 $\propto P(\text{hours} = 40.5-, \text{gender} = \text{male}, \text{wealth} = \text{rich})$

gender	hours_worked	wealth	probability
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

To calculate a true probability, we would sum these numerator terms up and divide this sum into whatever numerator that we're looking for the probability for.

Bayesian Inference/Classification

Usually, we'll be able to have a reasonable joint distribution table.

- At times, we'll have a mathematical model that provides *generative* information saying that if we know the probability of class y , what the probability of generating a set of features \mathbf{x} ?
 - In this flipped conditional expression, we refer to this as the **likelihood** $\rightarrow P(\mathbf{x}|y)$
 - With this, we can still compute the **posterior** \rightarrow what the probability of y given \mathbf{x} ?
- Bayes' rule gives us a neat formula to switch this *conditional*:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})}$$

- Where $P(y)$ is called the **prior**
- NOTE: Bayes' rule is generally useful when we're given a *generative model* that tells us the probability of class y generating some features \mathbf{x} .
 - Ex: We're told that for some class, it can generate features according to some probability density function, such as a normal/Gaussian distribution.

Probability Density Functions

While we may be given $P(\mathbf{x}|y)$ for discrete features, it is most common to use it if we have continuous features.

- If our features are generated according to some known distribution (like a Gaussian), then the likelihood can be approximated using a **probability density function (PDF)**.
 - A common distribution is the *normal* or *Gaussian distribution* (bell shaped curve).
- Given the means of the features μ and the covariance matrix Σ , the norm PDF is:

$$p(\mathbf{x}|\mu, \Sigma) \propto \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)^T}{2}}$$

Where $|\Sigma|$ is the *determinant* of the covariance matrix.

- This will give us a value proportional to the probability of generating a feature vector \mathbf{x} .

We may be given μ and Σ for a class, or we could *compute it* from the data (subset) that has that class ID.

- Either way, we can let $\mu^{(y)}, \Sigma^{(y)}$ be the mean and covariance of the data from class y , thus:

$$p(\mathbf{x}|y) = \frac{1}{\sqrt{(2\pi)^D |\Sigma^{(y)}|}} e^{-\frac{(\mathbf{x}-\mu^{(y)})\Sigma^{(y)-1}(\mathbf{x}-\mu^{(y)})^T}{2}}$$

- NOTE: At this point, evaluating this probability function is only *proportional* to the actual likelihood.

To arrive at the posterior probability that we'd want, we would multiply these likelihoods by the class prior for *each class* and then compute the overall posteriors by dividing by the sum of their products:

- If we do not divide by the sum of their products, this will give us a probability approximately proportional to the posterior probability.

$$\begin{aligned} P(y = 1|\mathbf{x}) &\propto P(y = 1)p(\mathbf{x}|y = 1) \\ P(y = 0|\mathbf{x}) &\propto P(y = 0)p(\mathbf{x}|y = 0) \end{aligned}$$

- Again, if we only care about what class an observation belongs to, we can simply look at the max of these probabilities.
- By dividing by the sum of the products here, we can get a *true probability* (between 0 and 1).

$$\begin{aligned} P(y = 1|\mathbf{x}) &= \frac{P(y=1)p(\mathbf{x}|y = 1)}{P(y=1)p(\mathbf{x}|y = 1) + P(y=0)p(\mathbf{x}|y = 0)} \\ P(y = 0|\mathbf{x}) &= \frac{P(y=0)p(\mathbf{x}|y = 0)}{P(y=1)p(\mathbf{x}|y = 1) + P(y=0)p(\mathbf{x}|y = 0)} \end{aligned}$$

Notes on General Usefulness of Statistical Classification

The joint distribution can be useful if we have a limited number of values/features and a lot of data to build a robust data distribution table.

Bayes' rule can be useful if we have a convenient way to either obtain or compute a generative model for our data, class-dependent generative model.

Naïve Bayesian Inference/Classification

We know that Bayes' rule computations above can be useful if we obtain or compute a generative model (class-dependent generative model) for our data:

- At times, we have a lot of features (exponential-level) that may have a lot of possible values or super limited amount of data, such that it is relatively difficult to create a robust joint distribution table.
 - If we have a 1,000 features with a 100 different possible values, then with features alone, we have 100^{1000} possible values and we must have enough data to have reliable probabilities for each of the possibilities.
 - We need to enough data to have reliable probabilities for each of the probabilities that we encounter with different combinations of features.
 - This is often NOT feasible.
- If we make a naïve assumption that features are conditionally independent of one another, then the generative likelihood CAN be *approximated* by the product of multiplying each features' generative probability.
 - **Conditional independence** - says that given one feature conditioned on our class, we can't say anything about the other features
 - **NOTE:** Often this is not a reasonable assumption (i.e. weight and height features being related) and will not give us same exact results, but will give reasonable results.
 - With this assumption, we can re-write $P(\mathbf{x}|y)$ as:

$$P(\mathbf{x}|y) \approx \prod_{j=1}^D P(x_j|y)$$

- In combining with Bayes' rule, we can say that if we need to compute the posterior probability, we can use Bayes' rule and this naïve assumption's calculation of the likelihood:

$$P(y|\mathbf{x}) = \frac{P(y)P(\mathbf{x}|y)}{P(\mathbf{x})} \approx \frac{P(y) \prod_{j=1}^D P(x_j|y)}{P(\mathbf{x})}$$

- Since we can't likely compute the evidence $P(\mathbf{x})$, since we know the probabilities over all the classes should sum to 1, we can just divide by the sum of their numerators again:

- If we have K classes, we can let:

$$\rho = \sum_{k=1}^K \left(P(y = k) \prod_{j=1}^D P(x_j|y = k) \right)$$

- And now:

$$P(y = i) = \frac{P(y = i) \prod_{j=1}^D P(x_j|y = i)}{\rho}$$

Log-Exponent Trick

Generally, all the terms in the product of the generative likelihood will be values between 0 and 1.

- If we have many features, multiplying a number of features can have a decent likelihood of **underflow**.
 - A workaround for this issue is a **log-trick**.
 - Trying to maximize the posterior is equivalent trying to maximize the log of the posterior:
 - By log properties, all of these products become summations, avoiding the likelihood of underflow.

$$\log(P(y|\mathbf{x})) \approx \log(P(y)) + \sum_{i=1}^D \log(P(x_i|y)) - \log(P(\mathbf{x}))$$

- If all we're interested in is what is the most likely class, whichever one of the *highest* is our most likely class.

$$\hat{y} = \operatorname{argmax}_i \left(\log(P(y = i)) + \sum_{j=1}^D \log(P(x_j|y = i)) \right)$$

- If we needed the true probability, we can undo the log by exponentiating the value.

$$P(y|\mathbf{x}) = e^{\log(P(y|\mathbf{x}))}$$

- This is referred to as the **log-exponent trick**.
- NOTE: If we are working with logs, we must be careful of $\log(0)$ since this is negative infinity.

Example

We want to determine if an object is a banana, orange, or something else based on its length, sweetness, and color, where these features are all *binary features*.

- Below are tables describing attributes of three types of fruit (over 1000 samples).

Banana				Orange				Other			
Long	Sweet	Yellow	Count	Long	Sweet	Yellow	Count	Long	Sweet	Yellow	Count
F	F	F	50	F	F	F	0	F	F	F	0
F	F	T	50	F	F	T	150	F	F	T	0
F	T	F	0	F	T	F	0	F	T	F	50
F	T	T	0	F	T	T	150	F	T	T	50
T	F	F	0	T	F	F	0	T	F	F	50
T	F	T	50	T	F	T	0	T	F	T	0
T	T	F	0	T	T	F	0	T	T	F	50
T	T	T	350	T	T	T	0	T	T	T	0

- Say we want to know the probability of being a banana if we observe that a fruit is long, not sweet, and yellow:
 - First, let's try to do this using **regular inference**:

Banana				Orange				Other			
Long	Sweet	Yellow	Count	Long	Sweet	Yellow	Count	Long	Sweet	Yellow	Count
F	F	F	50	F	F	F	0	F	F	F	0
F	F	T	50	F	F	T	150	F	F	T	0
F	T	F	0	F	T	F	0	F	T	F	50
F	T	T	0	F	T	T	150	F	T	T	50
T	F	F	0	T	F	F	0	T	F	F	50
T	F	T	50	T	F	T	0	T	F	T	0
T	T	F	0	T	T	F	0	T	T	F	50
T	T	T	350	T	T	T	0	T	T	T	0

$$\blacksquare P(B|L, \sim S, Y) = \frac{P(B, L, \sim S, Y)}{P(L, \sim S, Y)} = \frac{\frac{50}{1000}}{\frac{50}{1000}} = 1$$

◦ We can also try to calculate this using regular Bayes' rule:

$$\blacksquare P(B|L, \sim S, Y) = \frac{P(B)P(L, \sim S, Y|B)}{P(L, \sim S, Y)} = \frac{\frac{500}{1000} \frac{50}{500}}{\frac{50}{1000}} = 1$$

◦ We can also try to calculate this posterior using Naïve Bayes':

Banana			
Long	Sweet	Yellow	Count
F	F	F	50
F	F	T	50
F	T	F	0
F	T	T	0
T	F	F	0
T	F	T	50
T	T	F	0
T	T	T	350

$$\blacksquare P(B|L, \sim S, Y) \approx P(B)P(L|B)P(\neg S|B)P(Y|B) = \frac{50}{1000} \frac{400}{1000} \frac{150}{1000} \frac{450}{1000}$$

Working with Continuous Variables

Remember that each feature can typically fall into one of three categories:

1. Categorical nominal
2. Categorical ordinal
3. Continuous

All the previous material assumed categorical features only.

- What if our data is continuous?

In working with continuous variables, we can work in the native continuous space and use the continuous values to parameterize distributions → allowing us to use Naïve Bayes'

- For our purposes, we assume that these features will follow a normal/Gaussian distribution, allowing us to obtain a value proportional to the probability $P(x_j|y = i)$

- Earlier, we looked at the norm PDF:

$$P(x_j | \mu_{ji}, \sigma_{ji}) \propto \frac{1}{\sigma_{ji} \sqrt{2\pi}} e^{-\frac{(x_j - \mu_{ji})^2}{2\sigma_{ji}^2}}$$

- Note the use of μ_{ji} and σ_{ji} .
- These are the mean and standard deviation of the j^{th} feature of data from class i .
- For each class i and feature j , we should compute the mean of this j^{th} feature and standard of this j^{th} feature, giving us a mean and standard for that class and feature.
 - Given this, we can plug this into our regular univariate normal PDF function and get a generative probability of that value given a mean and standard deviation of that feature from a given class.
 - Then, we can use Naïve Bayes' to calculate the likelihood term that we'll need to calculate the posterior probability that we'd like to use for classification.

Example

Let's say we'd like to classify whether someone is a child/adult based on their height and weight, where we have 4 adults and 12 children.

- Given that our approach is to use Naïve Bayes, we can calculate the class priors given the observable data.
 - Then, we need to create the parameters for our generative PDF functions. Since we're using normal distributions, we need a mean and standard deviation.
 - We'll go through all adults' data and find the mean and standard deviation of the heights, doing the same with the weight feature.
 - We'd go through the same process for calculating these parameters for the childrens' data.
- Class probabilities: $P(y = a)?, P(y = c)?$
- Model for adults:
 - Height as a Gaussian with
 - $\mu_{a,h} = \frac{1}{4} \sum_{i:y_i=a} (x_{i,h})$
 - $\sigma_{a,h}^2 = \frac{1}{4} \sum_{i:y_i=a} (x_{i,h} - \mu_{a,h})^2$
 - Weight as a Gaussian $\mathcal{N}(\mu_{a,w}, \sigma_{a,w})$
- Model for children...
 - Height as a $\mathcal{N}(\mu_{c,h}, \sigma_{c,h})$
 - Weight as a $\mathcal{N}(\mu_{c,w}, \sigma_{c,w})$
- Given a new observation to classify, $x = (w, h)$, and making the assumption that our features are independent, if we want to figure out the probability of someone being an adult or child, we can use Naïve Bayes', $P(y = a|x)$ and $P(y = c|x)$
 - We can compute the posteriors as follows:

$$\begin{aligned} \bullet P(y = a|x) &= \frac{P(y=a)P(x|y = a)}{P(x)} \\ \bullet P(y = c|x) &= \frac{P(y=c)P(x|y = c)}{P(x)} \end{aligned}$$

Then if we make a naïve independence assumption, we arrive at

$$\begin{aligned} \bullet P(y = a|x) &\approx P(y = a)P(w|y = a)P(h|y = a) \\ \bullet P(y = c|x) &\approx P(y = c)P(w|y = c)P(h|y = c) \end{aligned}$$

- Again, if we want a true probability, we'd divide by the sum of these numerators.

Which Statistical Approach to Use?


The approach to use depends on what we have available:

- Ideally, if we have a limited number of features, possible values for those features, and a lot of data, s.t. we can build a robust distribution table, we'd ideally use **regular inference**.
- However, this is often difficult, so we may need to make the independence assumption to use Naïve Bayes.

Resources

A simple explanation of Naive Bayes Classification

I am finding it hard to understand the process of Naive Bayes, and I was wondering if someone could explain it with a simple step by step process in English. I understand it takes comparisons by ti...

 <https://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification/20556654#20556654>



Learning by Implementing: Gaussian Naive Bayes | by Dr. Robert Kübler...

archived 24 May 2021 23:48:46 UTC

 <https://archive.ph/8fZAY#selection-478.0-478.1>

Learning by Implementing: Gaussian Naive Bayes

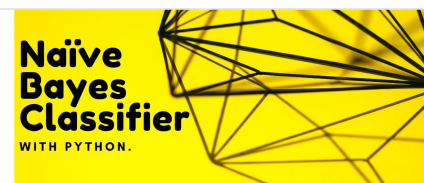
Learn how Gaussian Naive Bayes works and implement it in Python

by Robert Kübler · Jan 5 · 5 min read

Naive Bayes Classifier with Python - AskPython

Naive Bayes Classifier is a probabilistic classifier and is based on Bayes Theorem. In Machine learning, a classification problem represents the selection of the Best Hypothesis given the data. Given a new data point, we try to classify which class label this new data instance belongs to.

 <https://www.askpython.com/python/examples/naive-bayes-classifier>



<https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>


pandas.DataFrame.value_counts — pandas 1.5.3 documentation

 https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.value_counts.html

Using itertools.groupby:

How do I use itertools.groupby()?

IMPORTANT NOTE: You have to sort your data first. The part I didn't get is that in the example construction `groups = []` `uniquekeys = []` for `k, g in groupby(data, keyfunc): groups.append(list(g))` # Store group iterator as a list `uniquekeys.append(k)` `k` is the current grouping key, and `g` is an iterator that you

 <https://stackoverflow.com/questions/773/how-do-i-use-itertools-groupby>



GroupBy - pandas 1.5.3 documentation

GroupBy objects are returned by groupby calls: `pandas.DataFrame.groupby()` , , etc. Indexing, iteration Apply function group-wise and combine the results together. Apply a with arguments to this GroupBy object and return its result.

 <https://pandas.pydata.org/docs/reference/groupby.html>

pandas.DataFrame.groupby — pandas 1.5.3 documentation

 <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html>

Pillow documentation:

Image Module

The module provides a class with the same name which is used to represent a PIL image. The module also provides a number of factory functions, including functions to load images from files, and to create new images.


 <https://pillow.readthedocs.io/en/stable/reference/Image.html>



Confusion matrix plotting:

How can I plot a confusion matrix?

I am using scikit-learn for classification of text documents(22000) to 100 classes. I use scikit-learn's confusion matrix method for computing the confusion matrix.

 <https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix>



seaborn.heatmap — seaborn 0.12.2 documentation


 <https://seaborn.pydata.org/generated/seaborn.heatmap.html>

Review why we shouldn't iterate over rows in a DataFrame:

How to iterate over rows in a DataFrame in Pandas

I have a pandas dataframe, df:

```
c1 c2
0 10 100
```

 <https://stackoverflow.com/a/55557758>




Converting categories to numbers:

Pandas: convert categories to numbers

Suppose I have a dataframe with countries that goes as:

cc | temp

 <https://stackoverflow.com/a/38089089>



TypeError: first argument must be an iterable of pandas objects, you passed an object of type "DataFrame"

I have a big dataframe and I try to split that and after concat that.

I use

 <https://stackoverflow.com/a/50459002>



Convert pandas.DataFrame, Series and list to each other | note.nkmk.me

pandas.DataFrame, pandas.Series and Python's built-in type list can be converted to each other. This article describes the following contents. Convert list to pandas.DataFrame, pandas.Series For data-only list For list containing data and labels (row/column names) For data-only list For list containing d...

 <https://note.nkmk.me/en/python-pandas-list/>

pandas
note.nkmk.me