Probabilistic tools

Probabilistic Theory

Objectivism: the probability of an event is determined in a unique manner.

Subjectivism: the probability of an event is not determined in a unique manner.
Bayesianism is a probabilistic theory part of the subjectivism. It states that a probability varies depending on new information (Bayes theorem).
In Bayesian inference, random variables are X. θ is not random and not known. The objective is to estimate θ using *a-posteriori* probabilities.

Combinatorics

Permutations

A permutation (also called *arrangement*) is the number of sequences we can make in selecting elements from a set.
Conditions:
- selected elements are ordered
- no element occurs more than once
- it is not necessary to select all elements from the set

$$P_k^n = n(n-1)(n-2)...(n-k+1) = \frac{n!}{(n-k)!} \text{ where } k \leq n$$

Example: we have 4 numbered balls. If we select 2 of them, how many different ordered sequences can we do?

Combinations

Combinations are similar to permutations except that order doesn't matter. We thus adjust the above formula in removing the number of possible permutations in the selected sequence:

$$C_k^n = \binom{n}{k} = \frac{P_k^n}{P_k^k} = \frac{\frac{n!}{(n-k)!}}{\frac{k!}{0!}} = \frac{n!}{(n-k)!k!} \text{ where } k \leq n$$

Example (Time's up): we have 15 names. If I select 5 names, what is the total number of possible combinations? (Answer: 3003!!)

Expectation

Generic definition:
X random variable defined on $(\Omega, \mathcal{F}, \mathbb{P})$:
$\mathbb{E}[X] = \int_\Omega X(\omega) d\mathbb{P}(\omega)$

Using measure theory results, we find the specific cases for discrete and continuous variables.

For discrete variables :
$\mathbb{E}[X] = \Sigma_i x_i \mathbb{P}(X = x_i)$ $(= \mathbb{E}_{\mathbb{P}}[X])$

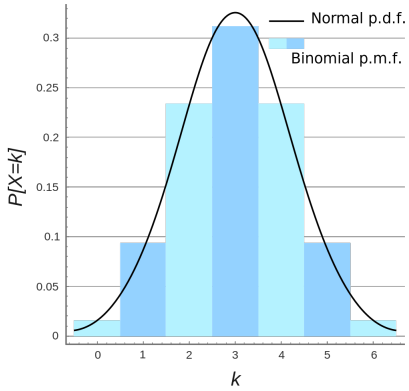
For continuous variables:
$\mathbb{E}[X] = \int x f(x) dx$ $(= \mathbb{E}_{\mathbb{P}}[X])$

Conditional expectation (discrete case):
$\mathbb{E}[Y|X = x] = \Sigma_y y \mathbb{P}(Y = y|X = x)$
It can also be written as a linear regression:
$\mathbb{E}[Y|X = x] = \beta_0 + \beta_1 X$

Distribution functions

Mass function
The probability mass function (p.m.f.) is the histogram of the distribution, that is:
- x-axis: values
- y-axis: frequency

Density function
The probability density function (p.d.f.) is the "smoothed histogram" of the distribution.

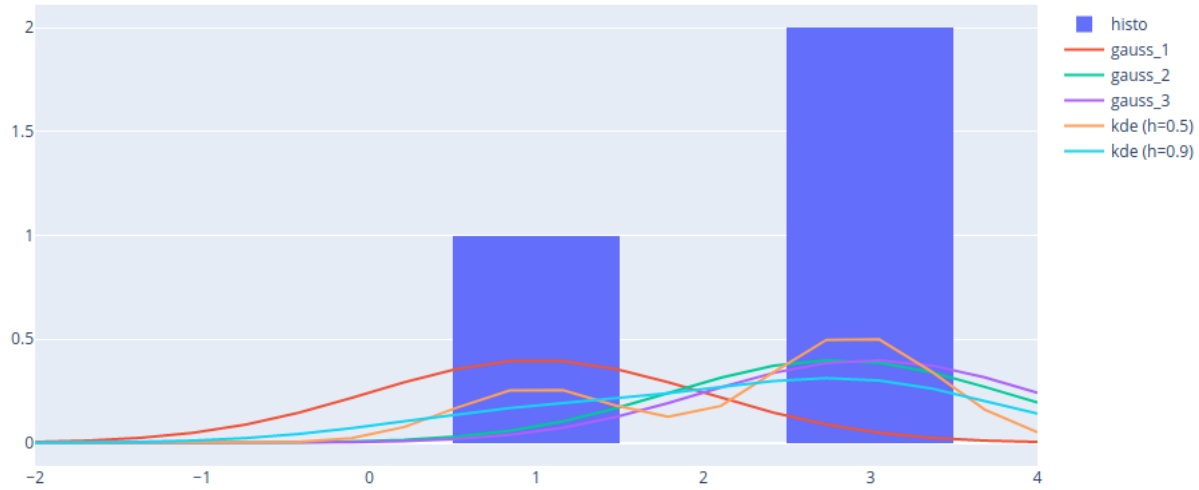


The major drawback of histograms is that they are not continuous, thus all points in the same range have the same estimated density. This can be adjusted in changing the bandwidth parameter (bins).
One of the common methods to solve this problem is the **Kernel Density Estimation** (KDE) method, also called the Parzen-Rosenblatt method. This method aims at estimating the density using the following formula:

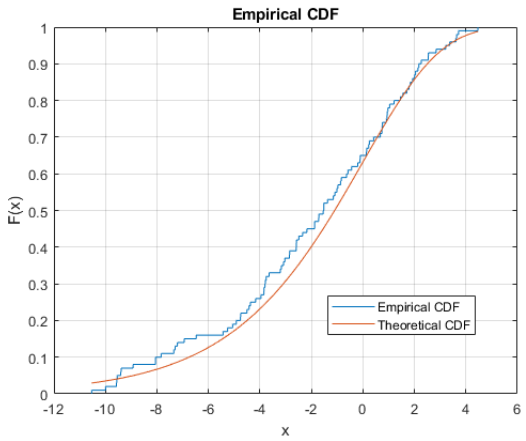
$$\widehat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

x is any value that we want to plot the function on.
x_i are the values from the distribution to be approximated.
h is the bandwidth (or smoothing parameter).

Since Kernel functions usually have inputs in $\mathbb{R}^2$, $K\left(\frac{x-x_i}{h}\right)$ can also be written $K(\frac{x}{h}, \frac{x_i}{h})$.
We note that the KDE estimation is an average of kernels. Thus, if the choosen kernel is continuous, the estimated density is continuous.
Using the Gaussian kernel $K(x,y) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{||x-y||_2}{2\sigma^2}\right)$, we note that the estimated density is an average of generated gaussian distributions (kernel) that are centered in x_i. In the graph below (extracted from KDE notebook), the distribution to be estimated is made of values $[1, 2.8, 3]$. Here the KDE method aims at building an average of three Gaussian distributions (*gauss_ 1*, *gauss_ 2*, *gauss_ 3*) centered respectively in 1, 2.8 and 3.



Cumulative distribution function
The cumulative distribution function (c.d.f) is given by $F_X(x) = \mathbb{P}(X < x)$.
The empirical distribution function is its estimation: $\widehat{F}_n(x) = \frac{1}{n}\{\text{number of elements} < x\}$



Listing 1: Python CDF easy implementation

```
plt.plot(np.sort(data_array), np.linspace(0, 1, len(data_array), endpoint=False))
```

Correlation

Pearson coefficient:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

np.cov(a,b) gives a **matrix** with covariances and **unbiased** variances (on the diagonal).
Several computation equivalences are shown below:

Listing 2: Pearson coefficient replication

```
a = pd.Series([5, 2, 6])
b = pd.Series([18, 2, 5])

print(a.corr(b) # biased standard deviation estimators !!
      == np.corrcoef(a,b)[0,1]
      == (np.cov(a,b)[0,1] / np.sqrt(np.cov(a,b)[0,0]*np.cov(a,b)[1,1]))
      == np.cov(a,b)[0,1] / (np.std(a,ddof=1)*np.std(b,ddof=1))
      != np.cov(a,b)[0,1] / (np.std(a)*np.std(b)))

# prints True
```

Note: when we compute those statistics numerically, we use **empirical** values.
Thus, $\mathbb{V}[X] = \mathbb{E}[X - \mathbb{E}[X]]$ is computed as $var_n(x) = \frac{1}{n}\Sigma(x_i - \overline{x})^2$

Autocorrelation (1):

$$R_k = \frac{\mathbb{E}[(X_i - \mu_X)(X_{i+k} - \mu_X)]}{\sigma_X^2}$$

X_i is the dataset without the last k values
X_{i+k} is the dataset without the first k values
μ_X is the mean on **the whole** dataset X
σ_X^2 is the variance **the whole** dataset X

Autocorrelation (2):

$$R_k = \frac{\mathbb{E}[(X_i - \mu_{X_i})(X_{i+k} - \mu_{X_{i+k}})]}{\sigma_{X_i}\sigma_{X_{i+k}}}$$

X_i is the dataset without the last k values
X_{i+k} is the dataset without the first k values
μ_{X_i} is the mean on dataset X_i
σ_{X_i} is the standard deviation on dataset X_i

statsmodels.tsa.stattools.acf uses formula (1).
np.autocorr uses formula (2).
Below is the summary of equivalences:

Listing 3: Autocorrelation replication

```
import statsmodels.tsa.stattools as sm

s = pd.Series([5, 2, 6, 18, 2, 5])

a = pd.Series([5, 2, 6])
b = pd.Series([18, 2, 5])

# Formula (1)
print(s.autocorr(3) # unbiased standard deviation estimators !!
      == a.corr(b)
      ==  np.cov(a,b)[0,1]/(np.std(a,ddof=1)*np.std(b,ddof=1)))

# prints True

# Formula (2)
def acf_by_hand(x, lag):
    y1 = np.array(x[:(len(x)-lag)])
    y2 = np.array(x[lag:])
    sum_product = np.sum((y1-np.mean(x))*(y2-np.mean(x)))
    return sum_product / (len(x) * np.var(x))

print(round(acf_by_hand(s,3),6)
       == round(sm.acf(s)[3],6)) # biased covariance and standard deviation estimators !!

# prints True
```

Below a graphical comparison of both formulas:

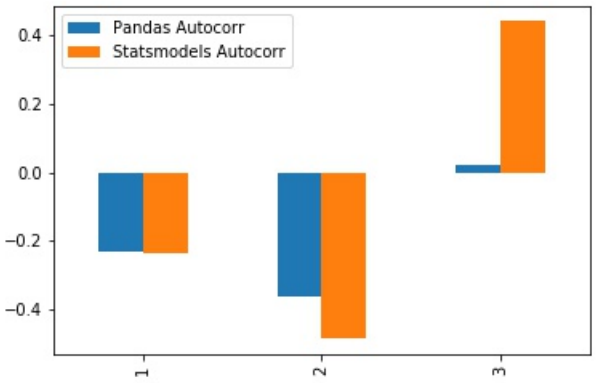
Listing 4: Graphical comparison of correlation computations

```
import statsmodels.tsa.stattools as sm

s = pd.Series([5, 2, 6, 18, 2, 5])
a = pd.Series([5, 2, 6])
b = pd.Series([18, 2, 5])

corr_statsmodel =  sm.acf(s)[1:4]
corr_pandas = [s.autocorr(i) for i in range(1,4)]

test_df = pd.DataFrame([corr_statsmodel, corr_pandas]).T
test_df.columns = ['Pandas Autocorr', 'Statsmodels Autocorr']
test_df.index += 1
test_df.plot(kind='bar')
```



Partial autocorrelation

Based on article understanding-partial-auto-correlation (towardsdatascience)

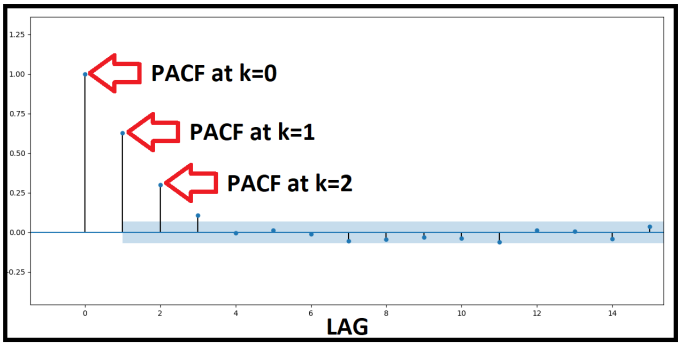
$$PR_k = \frac{cov(X_t|X_{t-1}...X_{t-k+1}, X_{t-k}|X_{t-1}...X_{t-k+1})}{\sigma_{X_t|X_{t-1}...X_{t-k+1}} \sigma_{X_{t-k}|X_{t-1}...X_{t-k+1}}}$$

$X_t|X_{t-1}...X_{t-k+1}$ is the residual of regression $X_t = \beta_0 + \beta_1 X_{t-1} + ... + \beta_k X_{t-k+1}$
$X_{t-k}|X_{t-1}...X_{t-k+1}$ is the residual of regression $X_{t-k} = \beta_0 + \beta_1 X_{t-1} + ... + \beta_k X_{t-k+1}$

Thus, one can write:

$$PR_k = \rho_{\epsilon_t, \epsilon_{t-k}}$$

We use partial autocorrelation in order to define the order p in which we can compute an AR(p) model.



Based on this graph, we can use an AR(2) or even AR(3) ($k = 3$ is just outside the 95% confidence interval.

Time series

Differential equations

A differential equation is an equation with the following characteristics:
- variables = functions
- it expresses the relationship of functions (variables) with their derivatives

Case of *linear and constant coefficient* differential equations:

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + ... + a_1 y' + a_0 y = 0 \text{ (E)}$$

(n): n-th derivative

In order to solve such equations, we use *characteristic equations*. Let $y(x) = e^{rx}$
(E) => $a_n r^n e^{rx} + a_{n-1} r^{(n-1)} e^{rx} + ... + a_1 r e^{rx} + a_0 e^{rx} = 0$
Since $e^{rx} \neq 0$
(E) => $a_n r^n + a_{n-1} r^{(n-1)} + ... + a_1 r + a_0 = 0$
We thus end up with a polynomial function.
In order to find the general solution of (E), we can find the solution of the characteristic equation and deduce the general solution (using exponential).

Autoregressive processes

Autoregressive processes are a specifc case of *differential equations*.

$y_{t+k} = \beta_1 y_{t+k-1} + \beta_2 y_{t+k-2} + ... + \beta_k y_t$

Characteristic equation:
$r^k - \beta_1 r^{k-1} - ... - \beta_{k-1} r - \beta_k = 0$

Stationary processes

A stationary process has the same moment (expectation, variance, etc.) in every single point.
In practice, we check the stationarity with only the first two moments (expectation and variance).

Intuition behind the importance of stationary processes in regressions:
When performing regressions, it is important to make sure the error term is stationary. If non stationary, there's probably a trend that is not caught by the explanatory variables used. This can lead to *spurious regressions*.

To make sure a process is stationary, we have to check the existence of a *unit root*.

Why existence of unit root leads to non-stationary process?
Toy example:
Let us consider a 1st order autoregressive process $y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t$

Let $\beta_0 = 0$. The characteristic equation is:
$r - \beta_1 = 0$
The solution is $r = \beta_1$
The problem has thus a unit root when $\beta_1 = 1$
Since $y_t = \beta_0 + \beta_1 y_{t-1} + \epsilon_t$ we can write:
$y_1 = y_0 + \epsilon_0$
$y_2 = y_1 + \epsilon_1 = y_0 + \epsilon_0 + \epsilon_1$
$y_3 = y_0 + \epsilon_0 + \epsilon_1 + \epsilon_2$
Thus, $y_t = y_0 + \Sigma_{j=0}^{t} \epsilon_j$
The variance is $\mathbb{V}[y_t] = t\sigma^2$ (we assume a constant variance for ϵ)
Consequently, the variance is increasing with time so the process is **not stationary**.

To detect stationarity, we can perform a unit root test such as *Augmented Dicky Fuller test.*

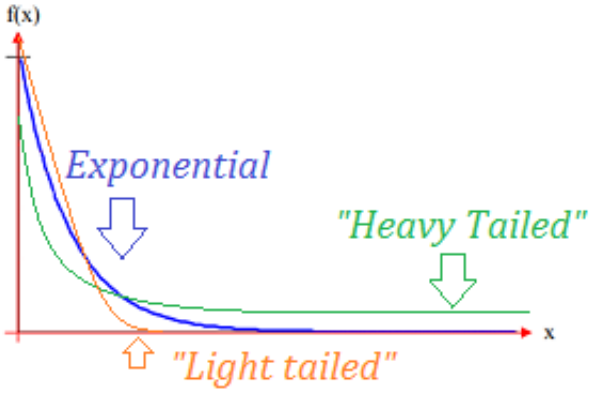
Non stationarity can be corrected in several ways :
- time regression : performing a regression on time and working with the error term
Example : if y_t in non stationary
$y_t = \beta_0 + \beta_1 t + \epsilon_t$ -> ϵ_t will not depend on time anymore
- finite differences : removing previous term to each observation $y_t = y_t - y_{t-1}$ -> this will have the effect to remove the trend
- moving average NxN
Example : using (double) centered moving average 5x5

Listing 5: Centered moving average (double)

```
cpi_roll = cpi.rolling(window=5).mean() # cell at index 4 is the mean of the 5 previous ones (i
cpi_mm = cpi - cpi_roll
cpi_roll_2 = cpi_mm.rolling(window=5).mean()
cpi_mm_2 = cpi_mm - cpi_roll_2
```

Heavy-Tailed Distribution

A distribution is heavy-tailed when there are more chances to get large values. Consequently, the variance is higher and will make the mean misleading as many outliers have high values. Below are p.d.f. (light-tailed and heavy-tailed):



A real-life example of heavy-tailed distribution is the income in the US.

Central Limit Theorem

Let $(X_n)_{n\geq 1}$ be a real and independent sequence with same law such that $\mu = \mathbb{E}[X_1]$ and $\mathbb{V}[X_1] = \sigma^2$ are defined ($\mathbb{V}[X_1] \leq +\infty$). Noting $\bar{X}_n = \frac{1}{n}(X_1 + ... + X_n)$, we have:

$$\sqrt{n}\frac{(\bar{X}_n - \mu)}{\sigma} \sim_{n\to\infty} \mathcal{N}(0,1)$$

Spectral Theorem

Let M be a symmetric matrix with real coefficients. Then it exists U orthogonal and D diagonal with real coefficients such that $M = UDU^T$.

Inferential statistics

Parametric Tests

A test is *parametric* if its goal is to test parameters of a known/unknown distribution.

Procedure:
1) find the test to perform
2) find the right estimator to use
3) deduce the reject region
4) compute the test statistic
5) retrieve quantiles of known distributions

Example 1 **(Gaussian-test)**:

(*inspired from example in Saporta p.325*)

$X_1, ..., X_n$ $(iid) \sim \mathbb{P}_\theta$

We want to analyze the mean. $m = a$?

1) find the test to perform

$$\begin{cases} \mathcal{H}_0 : m = a \\ \mathcal{H}_1 : m > a \end{cases}$$

2) find the right estimator to use

Since we are testing the mean, we choose the empirical mean as **estimator** $\widehat{\theta} = \frac{1}{n}\sum X_i$

3) deduce the reject region

We fix k for a rejection level α. The rejection region is:
$Z = \{\widehat{\theta} \geq k\}$

We look for k defined as such:
$\mathbb{P}_{\theta \in \Theta_0}(\widehat{\theta} \geq k) = \alpha$ => under $\mathcal{H}_0$, we reject the hypothesis when our estimator $\widehat{\theta}$ is above k
Intuitively, we want to keep our hypothesis if it's verified in most of the cases => under our hypothesis, there is a low probability that we are in the rejection region.
Thus, if in real life we have a result that makes the hypothesis unverified, we reject the hypothesis. However, we have a risk of α that our hypothesis was correct and that we ended up in the rejection region by mistake.

4) compute the test statistic

We center and reduce the estimator in order to get the Gaussian law and thus end up with known quantiles:
$\mathbb{P}_{\theta=a}(T \geq \frac{\sqrt{n}(k-a)}{\sqrt{\sigma^2}}) = \alpha$ with $T \sim_{n\to\infty} \mathcal{N}(0,1)$
T is the test statistic (a test statistic is a random variable for which we know the law under $\mathcal{H}_0$)

5) retrieve quantiles of known distributions

Finally, $\frac{\sqrt{n}(k-a)}{\sqrt{\sigma^2}} = q_\alpha$ => we can find k telling us when rejecting $\mathcal{H}_0$

Why not looking at the average directly?
=> the average can be influenced by the outliers and thus doesn't take into consideration extreme events.

How about the median?
=> the median doesn't take into account the distribution / tendency of the values.

α is also called the p-value. The lower the p-value is, the less error we make in rejecting our hypothesis so the more significant the rejection is.
p-value is the lowest error probability we want to make when rejecting our hypothesis.

When performing OLS, our hypothesis is $\theta_{x1} = 0$ so we don't reject it if the pvalue column is higher than our threshold. In the below OLS result, pvalues are displayed in column $P > |t|$. All variables are significant.

```
==============================================================================
Dep. Variable:                      y   R-squared:                       0.106
Model:                            OLS   Adj. R-squared:                  0.104
Method:                 Least Squares   F-statistic:                     62.11
Date:                Thu, 12 Mar 2020   Prob (F-statistic):           1.89e-14
Time:                        11:18:36   Log-Likelihood:                -383.98
No. Observations:                 526   AIC:                             772.0
Df Residuals:                     524   BIC:                             780.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.5010      0.027     55.870      0.000       1.448       1.554
x1             0.0240      0.003      7.881      0.000       0.018       0.030
==============================================================================
Omnibus:                        8.882   Durbin-Watson:                   1.776
Prob(Omnibus):                  0.012   Jarque-Bera (JB):               11.058
Skew:                           0.185   Prob(JB):                      0.00397
Kurtosis:                       3.606   Cond. No.                         10.9
==============================================================================
```

Example 2 (**T-test**): when the variance is not known.

Say we want to test whether a coefficient is zero:
1) find the test to perform

$$\begin{cases} \mathcal{H}_0 : \theta_j = 0 \\ \mathcal{H}_1 : \theta_j \neq 0 \end{cases}$$

2) find the right estimator to use

$$\widehat{\theta_j} = (X^T X)^{-1} X^T Y$$

3) deduce the reject region

$$Z = \{k_1 \leq \widehat{\theta_j} \leq k_2\}$$

4) compute the test statistic

$T_j = \frac{\widehat{\theta_j} - \theta_j}{\sigma_{\theta_j}} = \frac{\widehat{\theta_j}}{\sigma_{\theta_j}} \sim \mathcal{N}(0,1)$ with $\sigma_{\theta_j} = \sigma\sqrt{(X^T X)^{-1}}$ (recall that $\sigma = \sigma_\epsilon$)

Since we don't know σ, we can use the Cochrane theorem to remove this value:

$$T_j = \frac{\frac{\widehat{\theta_j}}{\sigma\sqrt{(X^T X)^{-1}}} \sim \mathcal{N}(0,1)}{\sqrt{\frac{\widehat{\sigma}^2(n-p-1)}{\sigma^2} \sim \mathcal{X}_{n-p-1}}} \sim \mathcal{T}(n-p-1) \text{ with } \widehat{\sigma}^2 = \frac{1}{n-p-1}\Sigma\epsilon^2$$

$$T_j = \frac{\widehat{\theta_j}}{\Sigma\epsilon^2\sqrt{(X^T X)^{-1}}}$$

5) retrieve quantiles of known distributions

Finally,

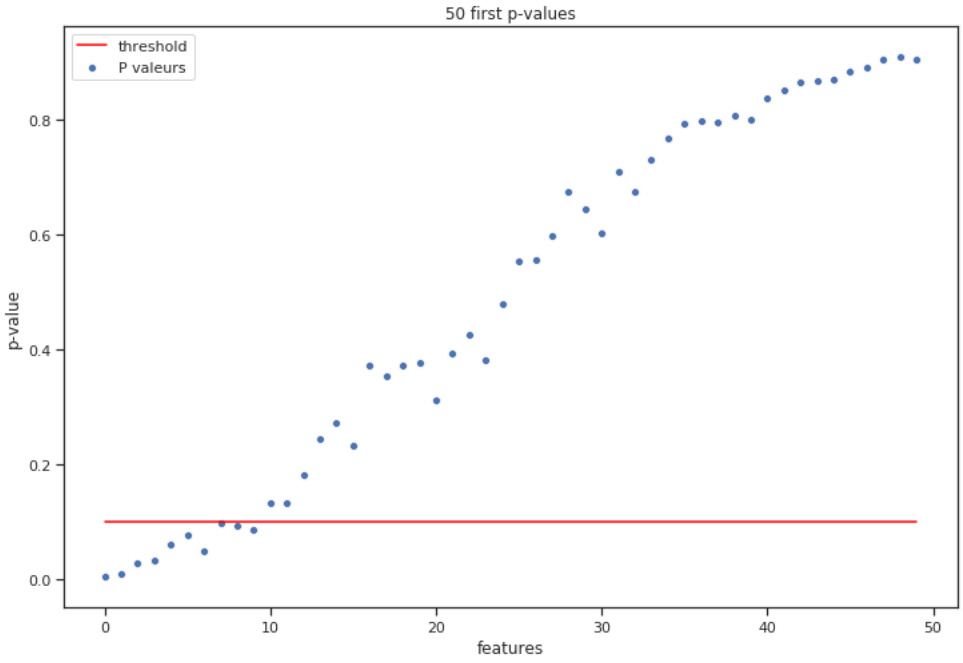
$\mathcal{P}_{\theta_j=0}(\frac{k_1}{\Sigma\epsilon^2\sqrt{(X^TX)^{-1}}} \leq T_j \leq \frac{k_2}{\Sigma\epsilon^2\sqrt{(X^TX)^{-1}}}) = \alpha$
Thus, $\frac{k_1}{\Sigma\epsilon^2\sqrt{(X^TX)^{-1}}} = t_{\frac{\alpha}{2}}$ (same for k_2)

Example 3 (**T-test** with forward selection):

Concept:
Regress all variables one by one on the most significant variable's residual, remove the most significant variable after each full round

Algorithm 1 Forward selection

$sel\_variables \leftarrow \emptyset$
for $i = 1$ to $nb\_variables$ **do**
  $resid\_mem \leftarrow \emptyset$
  $T\_stats \leftarrow \emptyset$
  for $j = 1$ to $rem\_variables$ **do**
    $Y = X_j\theta$
    $resid\_mem \leftarrow resid\_mem + \{res\}$ // adding residuals from previous regression
    $T\_stats \leftarrow T\_stats + \{T_j\}$ // T_j is computed as seen in example 2
  end for
  $k \leftarrow argmax(T\_stats)$
  $Y = resid\_mem(k)$
  $rem\_variable \leftarrow rem\_variable - \{k\}$
  $sel\_variables \leftarrow sel\_variables + \{k\}$
end for



(x-axis is the order in which we selected variables; see notebook *ACP_ForwardSelection_ Ridge_ Lasso.ipynb*)
We can then select only the most significant variables based on p-values on variables from list *sel_variables*
Note: since $pval = 2 * (1 - cdf(T)) = 2 * \frac{1-(1-\alpha)}{2}$, choosing the biggest T-stat is equivalent to choose the smallest p-value

Example 4 (**F-test**):

When several variables are correlated (often the case in practice), the student test is not efficient enough since it does not take the correlation into account. F-test allows to test **global** significativity.
Let's say we have 4 variables and we want to check the significativity of 2 of them.

$$\begin{cases} \mathcal{H}_0 : \theta_1 = \theta_2 = 0 \\ \mathcal{H}_1 : \theta_1, \theta_2 \neq 0 \end{cases}$$

$SSR = sum\ squared\ residuals = \Sigma(\widehat{y_i} - y_i)^2$

$$F = \frac{(SSR_C - SSR_{NC})/(p_{NC} - p_C)}{(SSR_{NC})/(n - p_{NC})} \sim \mathcal{F}(p_{NC} - p_C, n - p_{NC})$$

NC: not constraint model
C: constraint model

Method:
- OLS on not constraint model => computation of SSR_{NC}
- OLS on constraint model => computation of SSR_C

- Computation of the Fisher stat => computation of p-value (using complementary cumulative distribution function as above)

Listing 6: F-test

```
# Non constraint model
X0=np.column_stack((educ, exper, tenure, const))
model=sm.OLS(y,X0)
results = model.fit()
u=results.resid
SSR0=u.T@u

# Constraint model
X=np.column_stack((const, educ, tenure))
model=sm.OLS(y,X)
results = model.fit()
u=results.resid
SSR1=u.T@u

# Computation of Fisher stat
n=np.shape(X0)[0]
F=((SSR1-SSR0)/1)/(SSR0/(n-4))
f.sf(F,1,n-4) # p-value
```

Non-Parametric Tests

Example 1 **(Kolmogorov-Smirnov test)**:

- Test whether a sample follow a known law

F is the cumulative ditribution function and $\widehat{F_n}$ its empirical estimation.

The statistic test is $\widehat{F_n}(x) - F(x)$.

We have,

$\sqrt{n} \max_{1 \leq i \leq k} |\widehat{F_n}(x_i) - F(x_i)| \underset{n \to +\infty}{\to} \max_{0 \leq i \leq k} |W_i|$ where W_i is a *Brownian motion* or *Wiener process.*

We also have,

$\sqrt{n} \max_{0 \leq x \leq 1} |\widehat{F_n}(x) - x)| \underset{n \to +\infty}{\to} \max_{0 \leq x \leq 1} |B(x)|$ where B is a *Brownian bridge.*

Proofs (Empirical-Process Theory)

A Brownian bridge has the following property:

$\mathbb{P}(\sup_{t \in [0,1]} |B_t| \geq b) = 2\Sigma_{n \geq 1}(-1)^{n-1} e^{-2n^2 b^2}$.
This allowed statisticians to draw a quantile table, we can thus easily know the critical region.

- Test whether two samples follow the same law

In that case, the statistic is the distance $D_{n,m} = sup_x|\widehat{F}_{1,n}(x) - \widehat{F}_{2,m}(x)|$.

Associated test hypothesis are:

$$\begin{cases} \mathcal{H}_0 : \widehat{F}_{1,n}(x) = \widehat{F}_{2,m}(x) \\ \mathcal{H}_1 : \widehat{F}_{1,n}(x) \neq \widehat{F}_{2,m}(x) \end{cases}$$

We reject the null hypothesis for level α if $D_{n,m} > \frac{1}{\sqrt{n}}\sqrt{-ln(\frac{\alpha}{2})\frac{1+\frac{n}{m}}{2}}$.

Scipy

Test statistic computation

Listing 7: Kolmogorov-Smirnov test statistic

```
cdf1 = np.searchsorted(data1, data_all, side='right') / n1
cdf2 = np.searchsorted(data2, data_all, side='right') / n2
cddiffs = cdf1 - cdf2
T = np.max(cddiffs)
```

Critical probability computation

The critical probability is computed differently depending on sample size. If sample size is small, an exact computation is done. If sample size is large, an asymptotic computation is done. In both cases, the critical probability is computed using combinatorics and largely inspired by J. L. Hodges, Jr..

Example 2 **(Wilcoxon-Mann-Whitney test** or **Mann-Whitney U test)**:

- Test whether two samples follow the same law

$$\begin{cases} \mathcal{H}_0 : \widehat{F}_{1,n}(x) = \widehat{F}_{2,m}(x) \\ \mathcal{H}_1 : \widehat{F}_{1,n}(x) \neq \widehat{F}_{2,m}(x) \end{cases}$$

The test statistic is $U = \Sigma rank_1 - \frac{n_1(n_1+1)}{2}$ where $rank_1$ are the ranks of each element from the first dataset in the second dataset.

Intuition behind the test
The test is equivalent of ranking all elements from the two datasets; if the resulting dataset is well mixed, the p-value is likely to be high (similar distributions).
The test can also be interpreted as a comparison between the two medians; if they are very different, distributions are likely to be different.

Example 3 **(Fisher exact test)**:

- Test whether proportions are representative

$\mathcal{H}_0$: proportions are representative

This test is to be used for the analysis of contingency tables.

	Men	Women	*Row Total*
Studying	***a***	***b***	*a + b*
Non-studying	***c***	***d***	*c + d*
Column Total	*a + c*	*b + d*	*a + b + c + d (=n)*

Fisher showed that knowing the total numbers (Row Total and Column Total), the probability to have a certain combination follows a *hypergeometric* distribution.

$$p = \frac{C_a^{a+b} C_c^{c+d}}{C_{a+c}^n}$$

The test is said *exact* since there is no asymptotic behavior in the formula.

$p < \alpha$ means that, based on total numbers, this specific combination is unlikely to happen.

Example 4 **(Chi-2 test)**:

- Test whether several samples follow the same law = samples are distributed in the same proportions among the different categorical variables.

	Feature 1	**Feature 2**		**Feature r**	Total
Sample 1	n_{11}	n_{12}		n_{1r}	...
Sample 2	n_{21}	n_{22}		n_{2r}	...
. . .			. . .		
Sample k	n_{k1}	n_{k2}		n_{kr}	
Total	...	...		...	n

Note: the table is read as such: "In sample 1, n_{11} individuals have feature 1, n_{12} individuals have feature 2, ...". Thus, the columns are derived from categorical variables in building a contingency table (displays the frequencies of variables).

$\mathcal{H}_0$: All samples have the same probabilities to have feature 1, 2, 3, ... Those probabilities are $p_1, p_2, ..., p_r$.

The test statistic is: $X^2 = \Sigma_{i=1}^k \Sigma_{j=1}^r \frac{(n_{ij} - n_{i.} p_j)^2}{n_{i.} p_j}$ where $n_{i.}$ is the sum of row i.
The test statistic is also often written as such: $X^2 = \Sigma_{i=1}^k \Sigma_{j=1}^r \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$ where O_{ij} are the observed numbers in the above table and E_{ij} are the expected numbers (numbers we want to have).

Pearson showed that $X^2 \sim \mathcal{X}^2_{kr-k}$. We can thus use the quantile table to deduce whether we reject $\mathcal{H}_0$.

Fairness metrics

How can we make sure the model doesn't discriminate any group? To answer this question, we first have to define a **sensitive attribute**. This attribute is the criteria we want to assess the fairness with respect to.
In the case of a bank, we can think about a model predicting whether a client will default. We can take **client gender** as a sensitive attribute. Then the question becomes: are we sure men and women are equally likely to be granted a loan?

Statistical Parity

Statistical Parity, also called Demographic Parity or Group Fairness, aims at comparing proportions of predictions conditioned on the sensitive attribute.

$$\mathbb{P}(\hat{Y} = 1|A = a) = \mathbb{P}(\hat{Y} = 1|A = b)$$

Where A is the sensitive attribute (e.g. gender). From a statistical perspective, this metric consists of comparing two distributions of $\hat{Y}$: one conditioned on $A = a$ and one conditioned on $A = b$. More details can be found on the metric in this article.

The next two metrics are thoroughly explained in this article and this article. Their main advantages over Statistical Parity is that **they account for correlation between the target value and the sensitive attribute**.

Equal Opportunity

$$\mathbb{P}(\hat{Y} = 1|A = a, Y = 1) = \mathbb{P}(\hat{Y} = 1|A = b, Y = 1)$$

Here we focus only on positive outcome. In other words, we measure how fair the model is when it successfully predicts that clients will default. This definition is also equivalent to comparing True Positive Rates.

Predictive Equality

$$\mathbb{P}(\hat{Y} = 1|A = a, Y = 0) = \mathbb{P}(\hat{Y} = 1|A = b, Y = 0)$$

Here we focus only on negative outcome. In other words, we measure how fair the model is when it wrongly predicts that clients will default. This definition is also equivalent to comparing False Positive Rates.

Likelihood method

This method consists on finding the parameter that maximizes the likelihood of an event. The event here is to observe some data. It is usually done when we know the type of law of a random variable (uniform, gaussian etc.) and we are looking for the parameter that maximizes the likelihood ($\approx$ probability) that an event occurs.

$L(\theta; x_1, ..., x_n) = \prod_{i=1}^{n} f(x_i; \theta)$ which is the product of densities across all samples.
In discrete form: $L(\theta; x_1, ..., x_n) = \prod_{i=1}^{n} \mathbb{P}(X = x_i; \theta)$

Note (wording clarification): $L(\theta|X) = \mathbb{P}(X|\theta)$
$\mathbb{P}(X|\theta)$: the probability of observing an event with fixed model parameters.
$L(\theta|X)$: the likelihood of the parameters taking certain values given that we observe an event.

Intuitively, we want to find the θ that maximizes a certain event, that is, **obtaining some data** X (which is why we have $X|\theta$).
We often use the log in order to get rid of power coefficients appearing with the product.
likelihood equation: $\frac{d}{d\theta} ln(L(x_1, ..., x_n; \theta)) = 0$

Note: in machine learning, we use likelihood maximization in unsupervised learning when we want to estimate parameters of a distribution sample (generative models).

Exploratory statistics

Distance Metrics

In statistic, the generic distance metric is expressed as follow:

$$d(x, y) = (x - y)^T M (x - y)$$

where M is a symmetric positive definite matrix.

Note: the distance is a number (1 dimension).

Euclidean distance

This is equivalent to the generic definition with $M = Id$.

Euclidean distance is also called the 2-norm: $\Sigma_{i=1}^{n}(x_i - y_i)^2$

Mahalanobis distance

This is equivalent to the generic definition with $M = \Sigma^{-1}$.

It is also common to define the squared distance between a vector x and its mean vector μ_x:

$$D^2 = (x - \mu_x)^T \Sigma^{-1} (x - \mu_x)$$

Advantage : it takes into account the data standard deviation and correlation. The more the data is dispersed, the lower the distance is. Indeed, using the inverse matrix is like if we divided the distance from the mean $(x - \mu_X)$ by the standard deviation.

Principal component Analysis

The PCA's objective is to get an approximation of data in a **low** dimensional space.

Inertia

Inertia $I_M = \Sigma_{i=1}^{n} p_i ||x_i - g||_M^2$

where $g^T = (\bar{x}^{(1)}, ..., \bar{x}^{(p)})$ also called the *gravity center*.

–> The inertia is thus the weighted average of the squared distance of each observation with the gravity center.

–> p_i is the weight given to each observation. Most of the times, $p_i = \frac{1}{n}$ (every observation contributes equally to the analysis)

–> the distance $||.||$ depends on the choosen metric M

If the data are centered:

$$I_M = \Sigma_{i=1}^{n} p_i x_i^T M x_i$$

Since $I_M \in \mathbb{R}$:

$$I_M = Tr(\Sigma_{i=1}^{n} p_i x_i^T M x_i)$$

Thanks to the trace properties:

$$I_M = Tr(\Sigma_{i=1}^{n} M x_i p_i x_i^T)$$

With $V = Cov(X)$:

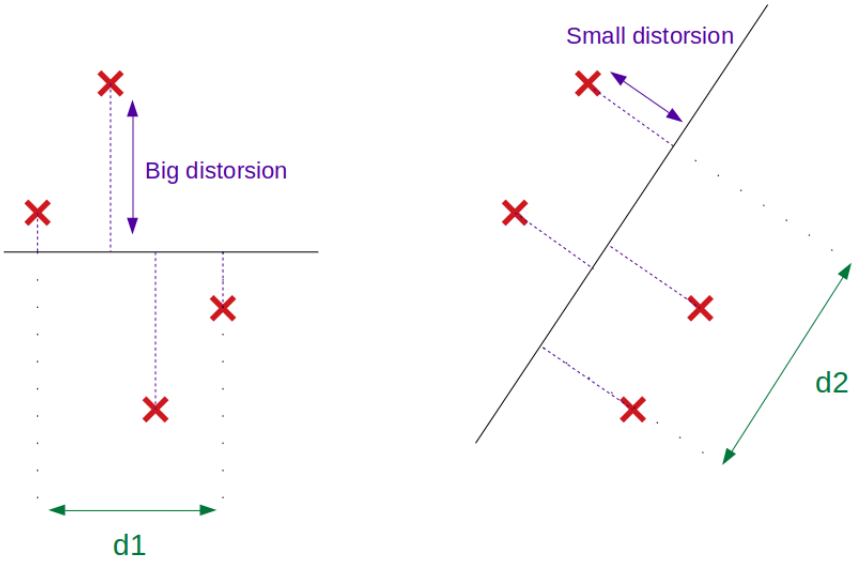
$$I_M = Tr(MV)$$

Projection

In order to represent the data in a low dimensional space, we use projections.

The projection should distort the initial space the less as possible, that is:
=> reduce the projection distances as much as possible
=> maximize the average of squared distances between projected points
=> maximize inertia of the projected points

In the below figure, maximizing the inertia leads to choosing the projection on the right since d2 > d1.



Let P a projector. $V = Cov(X) = X^T DX$ (with D the weight matrix). The covariance matrix of the projected points is:

$$V_P = (PX)^T D(PX) = (XP^T)^T D(XP^T) = PX^T VXP^T = PVP^T$$

Note: a projector P is such that $P^2 = P$ and $PM = MP^T$

Optimisation

As seen previously, the objective is to maximize the inertia. Combining the previous 2 paragraphs, we can express the inertia of projected points:

$$Ip_M = Tr(V_p M) = Tr(PVP^T M)$$

$$\begin{aligned} Tr(PVP^T M) &= Tr(PVMP) \text{ since } PM = MP^T \\ &= Tr(VMP^2) \text{ since } Tr(AB) = Tr(BA) \\ &= Tr(VMP) \text{ since } P^2 = P \end{aligned}$$

Thus the optimisation problem is:

$$\max\ Ip_M = \max\ Tr(VMP)$$

The objective is to find the line (in black on above figure) going through g and maximizing the inertia. Let a be a point on this line. We have the following equation:

$$P = a(a'Ma)^{-1}a'M$$

(Indeed we have $P^2 = P$ and $PM = MP^T$)

$$
\begin{aligned}
Tr(VMP) &= Tr(VMa(a'Ma)^{-1}a'M) \\
&= \frac{1}{a'Ma}Tr(VMaa'M) \\
&= \frac{Tr(a'MVMa)}{a'Ma} \\
&= \frac{a'MVMa}{a'Ma} \text{ since } a'MVMa \text{ is a scalar}
\end{aligned}
$$

In order to obtain the maximum, we use first order optimal conditions:

$$\frac{d}{da}(\frac{a'MVMa}{a'Ma}) = 0$$

With $\frac{d}{da}(\frac{a'MVMa}{a'Ma}) = \frac{(a'Ma)2MVMa - (a'MVMa)2Ma}{(a'Ma)^2}$, previous equation becomes:

$MVMa = (\frac{a'MVMa}{a'Ma})Ma$

Since $\frac{a'MVMa}{a'Ma}$ is a scalar, let's replace it by λ:

$$VMa = \lambda a$$

Based on eigenvalue definition, λ is thus the eigenvalue of VM.

We can thus rewrite the optimization problem:

$$\max\ Ip_M = \max\ Tr(VMP) = \max\ \lambda$$

This final result leads to the theorem:

The lower dimensional space is given by the eigenvectors associated with the biggest eigenvalues.

Predictive models

Linear regression

$$Y = X\theta + \epsilon$$

Hypothesis: $\begin{cases} \mathbb{E}[\epsilon] = 0 \\ \mathbb{V}[\epsilon] = \sigma \end{cases}$

Bias

$$Bias = \mathbb{E}[\widehat{\theta} - \theta^*]$$

$$\begin{aligned}\mathbb{E}[\widehat{\theta}] &= \mathbb{E}[(X^TX)^{-1}X^TY] \\ &= \mathbb{E}[(X^TX)^{-1}X^T(X\theta^* + \epsilon)] \\ &= \theta^* + (X^TX)^{-1}X^T\mathbb{E}[\epsilon] \\ &= \theta^*\end{aligned}$$

The estimator is **not biased**.

Variance-covariance

$$\begin{aligned}Cov(\widehat{\theta}) &= \mathbb{V}[(X^TX)^{-1}X^TY] \\ &= \mathbb{V}[(X^TX)^{-1}X^T(X\theta^* + \epsilon)] \\ &= 0 + ((X^TX)^{-1}X^T)^T\mathbb{V}[\epsilon](X^TX)^{-1}X^T \\ &= (X^TX)^{-1}\sigma^2 \qquad \text{since } X^TX \text{ is symmetric}\end{aligned}$$

Note: the variance-covariance is a matrix. We define here the variance as a number.

$$\mathbb{V}[\widehat{\theta}] = \mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])^2]$$

We know that $||u||_2 = \Sigma_k u_k^2 = Tr(uu^T)$.

Thus:

$$\begin{aligned}\mathbb{V}[\widehat{\theta}] &= \mathbb{E}[Tr((\widehat{\theta} - \mathbb{E}[\widehat{\theta}])(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])^T)] \\ &= Tr(\mathbb{E}[(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])(\widehat{\theta} - \mathbb{E}[\widehat{\theta}])^T)] \quad \text{since the trace is a number} \\ &= Tr(Cov(\widehat{\theta})) \\ &= Tr((X^TX)^{-1}\sigma^2) \\ &= \sigma^2 Tr((UDU^T)^{-1}) \quad \text{thanks to the spectral theorem (we assume inversible matrices)} \\ &= \sigma^2 Tr((UU^T)^{-1}D^{-1}) \quad \text{thanks to the trace properties} \\ &= \sigma^2 Tr(D^{-1}) \quad \text{since } U \text{ is orthogonal} \\ &= \sigma^2 Tr(\begin{bmatrix} \frac{1}{\lambda_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\lambda_p} \end{bmatrix}) \quad \text{with } \lambda_i \text{ the eigenvalues} \\ &= \sigma^2 \Sigma_{k=1}^p \frac{1}{\lambda_k}\end{aligned}$$

We can see that the variance becomes **unstable** when eigenvalues are small, which is the case when variables are collinear.

Performance metrics for classification

1) Accuracy

The accuracy is the percentage of correct predictions:

$$accuracy = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{\hat{y}_i = y_i\}$$

where $\hat{y}$ is the prediction and y the true value.

Note: for imbalanced data, the accuracy can be very misleading because a classifier can achieve a high score just by predicting the majority class with probability 1.

In case of binary classification, the accuracy can be written as such: $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

2) True positive rate

The TPR measures the proportions of positives that are correctly identified.

$$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$$

Note: this metric is typically used for medical tasks as it is crucial to minimize the count of false-negatives.

3) False positive rate

The FPR measures the proportions of negatives that are correctly identified.

$$FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$$

4) ROC (Receiver Operating Characteristic) curve

The ROC curve combines the TPR and FPR.

Use of the ROC

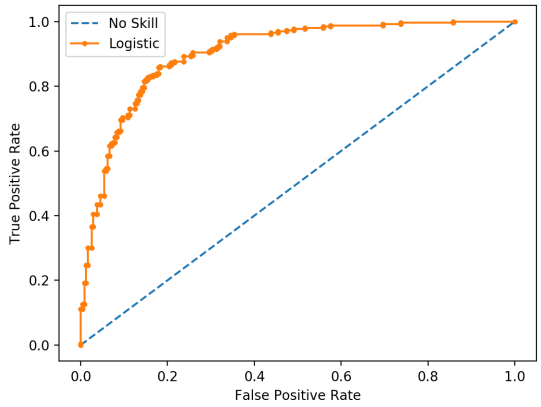
If one model is used:
We use the ROC curve to evaluate the performance of one classifying model that we can obtain when varying a threshold.

If several models are used:
We use the ROC curve to compare several classifiers in evaluating the area under the curve (AUC) for a range of thresholds.

Intuition

After running the prediction of a specific model, we draw the confusion matrix (see below) with a certain threshold.

We then modify the threshold and draw another confusion matrix.
The ROC curve summarizes all of the confusion matrices that each threshold produced.



Implementation

1. Get probability predictions
2. Sort the probabilities (prediction)
3. Sort the validation (actual) according to previous sort
4. Loop on the sorted validation. At each iteration:
- increment TP or FP
- compute the TPR and FPR.
5. Plot (FPR, TPR)

See *https://docs.eyesopen.com/toolkits/cookbook/python/plotting/roc.html* for an implementation example, or data challenge Face_Recognition.

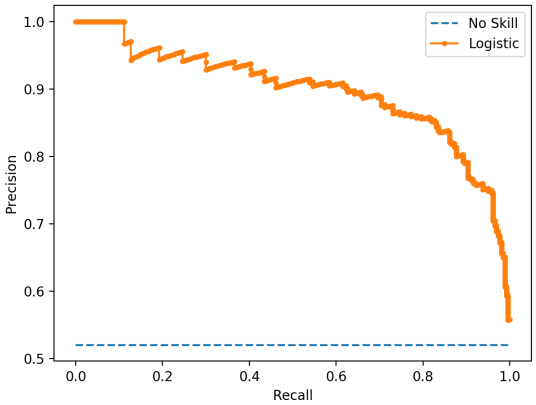
5) Precision

$$precision = \frac{TP}{TP + FP}$$

Note: this metric should be used when false-negative is not too much a concern (e.g. YouTube recommendations). It can also be used when the data are imbalanced (see details below).

6) PR (Precision Recall) curve

The PR curve combines TPR (recall) and $precision$.



The PR curve is better adapted than the ROC curve in the case of imbalanced data:

ROC curve uses $FPR = \frac{FP}{N}$ –> N can be either very large or very small if classes are imbalanced. PR curve uses Precision $= \frac{TP}{TP+FP}$ –> the precision considers only the positive values coming from the model.

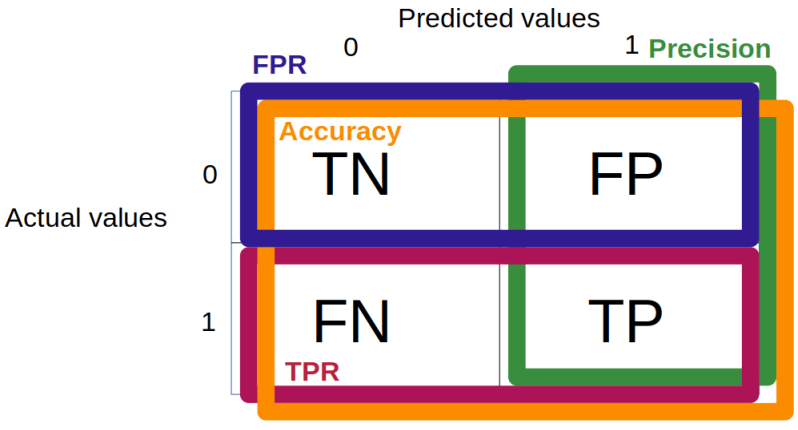
7) F1-score

F1-score is a single metric that combines TPR (recall) and precision.

$$f1\text{-}score = 2\frac{precision * TPR}{precision + TPR}$$

Note: it is widely used for imbalanced datasets since it involves the precision.

Metric summary



Performance metrics using likelihood

AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are based on a trade-off between model accuracy and model complexity.
Those criteria have foundations in *information theory*: it measures the **loss of information**. The lower the metrics, the less information a model losses thus the higher the quality of that model.

$$AIC = -2 \ln L(\hat{\theta}) + 2k$$

$$BIC = -2 \ln L(\hat{\theta}) + ln(n)k$$

$L(\hat{\theta})$ = the maximized value of the likelihood function of the model, i.e. $L(\hat{\theta}) = \mathbb{P}(x|\hat{\theta})$ where $\hat{\theta}$ are the parameter values that maximize the likelihood function and x the observed data.
k is the number of parameters in the model.
n is the sample size.

The first term is the **risk of underfitting**: we want the highest probability to observe a behavior under specific parameters. It measures how well adapted is our model for our data. The more parameters we have, the better the model is. We also call it the *goodness of fit*.

The second term **risk of overfitting**: we want to penalize models with high number of parameters because the more we increase this number the more the model becomes complex.

The main difference between the two metrics is that BIC penalizes more overfitting because of the n term. Consequently, for large sample sizes, BIC will favor models with few parameters compared with AIC.