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.

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

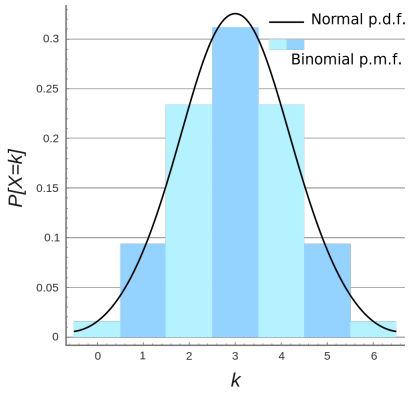
For continuous variables:
$\mathbb{E}[X] = \int x f(x) dx$ $(= \mathbb{E}_\mathbb{P}[X])$

Conditional expectation (discrete case):
$\mathbb{E}[Y|X = x] = \Sigma_y y \mathbb{P}(Y = y|X = x)$

Mass and density 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 "smooth histogram" of the distribution.



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 1: 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 2: Autocorrelation replication

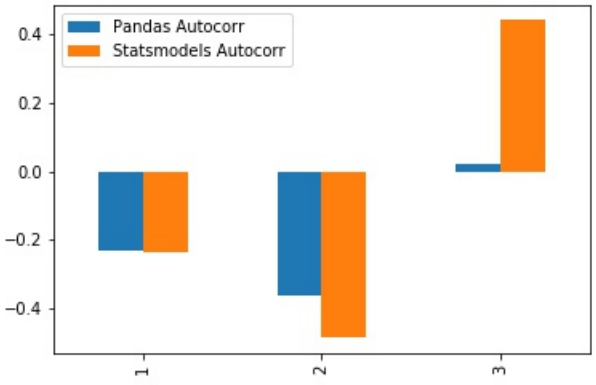
```
import statsmodels.tsa.stattools as sm

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 comparision of both formulas:

Listing 3: Graphical comparison of correlation computations

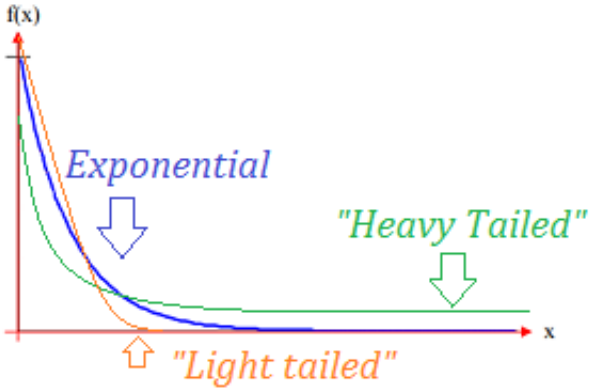
```
import statsmodels.tsa.stattools as sm

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')
```



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



An 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

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:

(*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

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

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 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: 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:

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

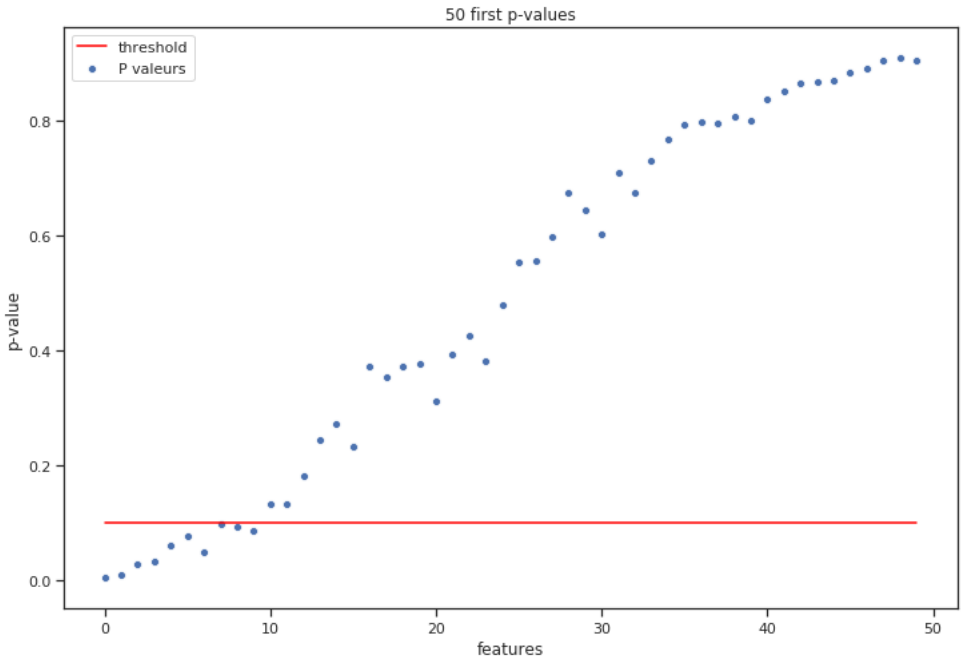
Finally,
$\mathcal{P}_{\theta_j=0}(\frac{k_1}{\Sigma\epsilon^2\sqrt{(X^T X)^{-1}}} \leq T_j \leq \frac{k_2}{\Sigma\epsilon^2\sqrt{(X^T X)^{-1}}}) = \alpha$
Thus, $\frac{k_1}{\Sigma\epsilon^2\sqrt{(X^T X)^{-1}}} = t_{\frac{\alpha}{2}}$ (same for k_2)

Example 3 (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 survival function as above)

Listing 4: 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
```

Likelihood method

This method consists on finding the parameter that maximizes the likelihood of an event. 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

Mahalanobis distance
Mahalanobis distance is a good alternative to Euclidean distance. For any given point x in a set X, the squared Mahalanobis distance is:

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

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.
Note: Euclidean distance is when $\Sigma = Id$.

Predictive models

ROC curve

ROC curve is used essentially for binary classification.
ROC = Receiver Operating Curve

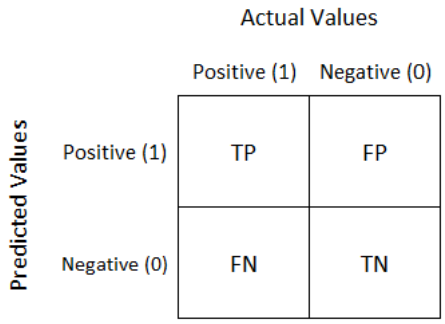
Use of the ROC curve

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

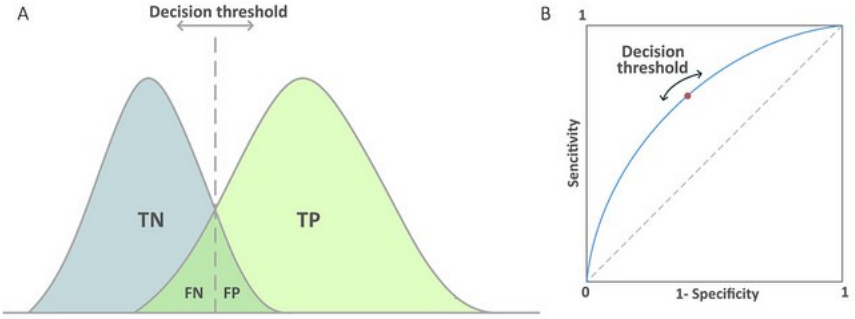
Several models:
We use ROC curve to compare several classifying models in evaluating the area under the curve (AUC) for a range of threshold.

Intuition

After running the prediction of a specific model, we draw the confusion matrix (actual vs predited) with a certain threshold.



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



On the left picture we see the ability of a model to give a clear distinction between the two classes. The curves are drawn from the predictions and the actual results (**how?**)

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.