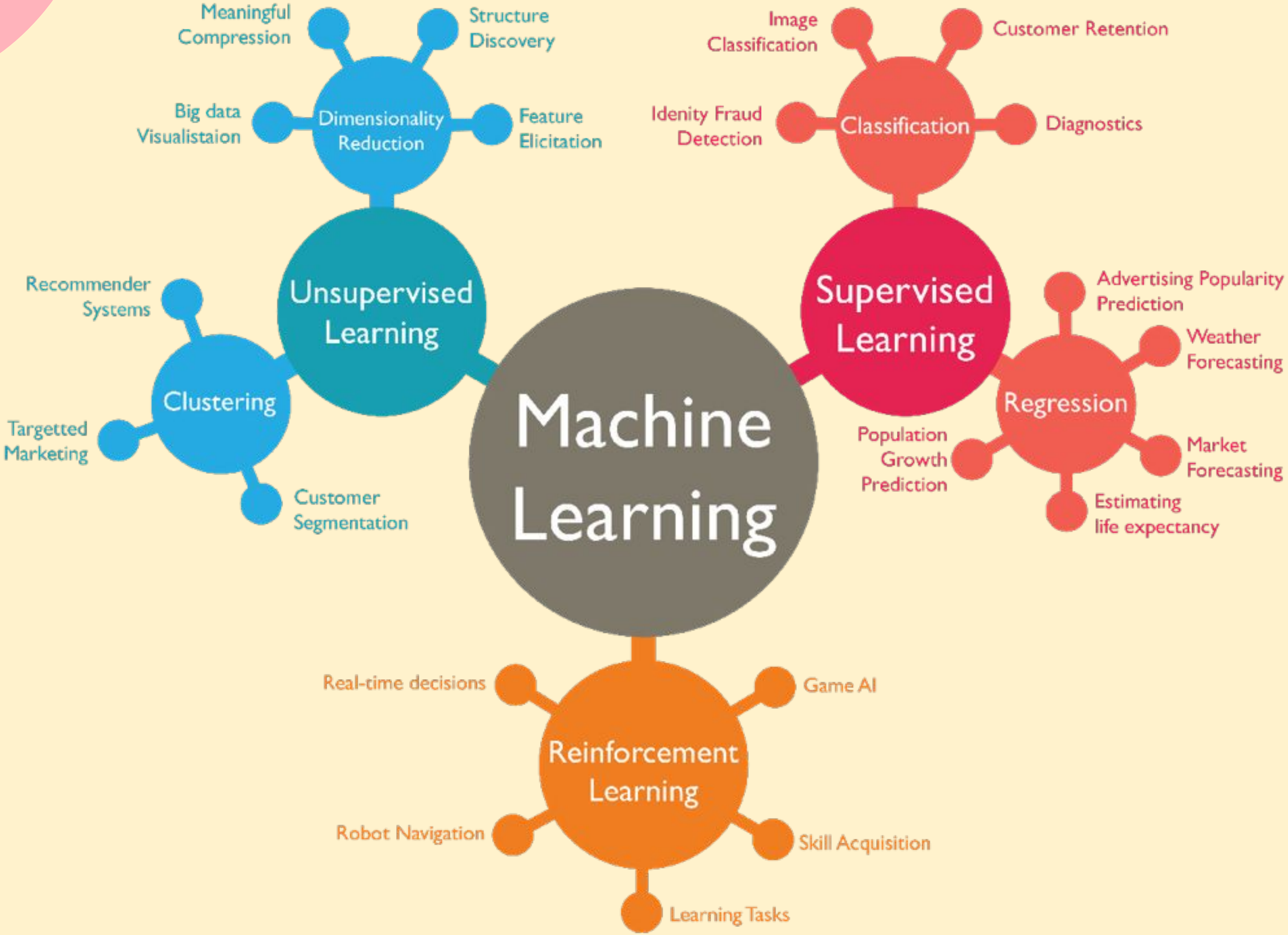
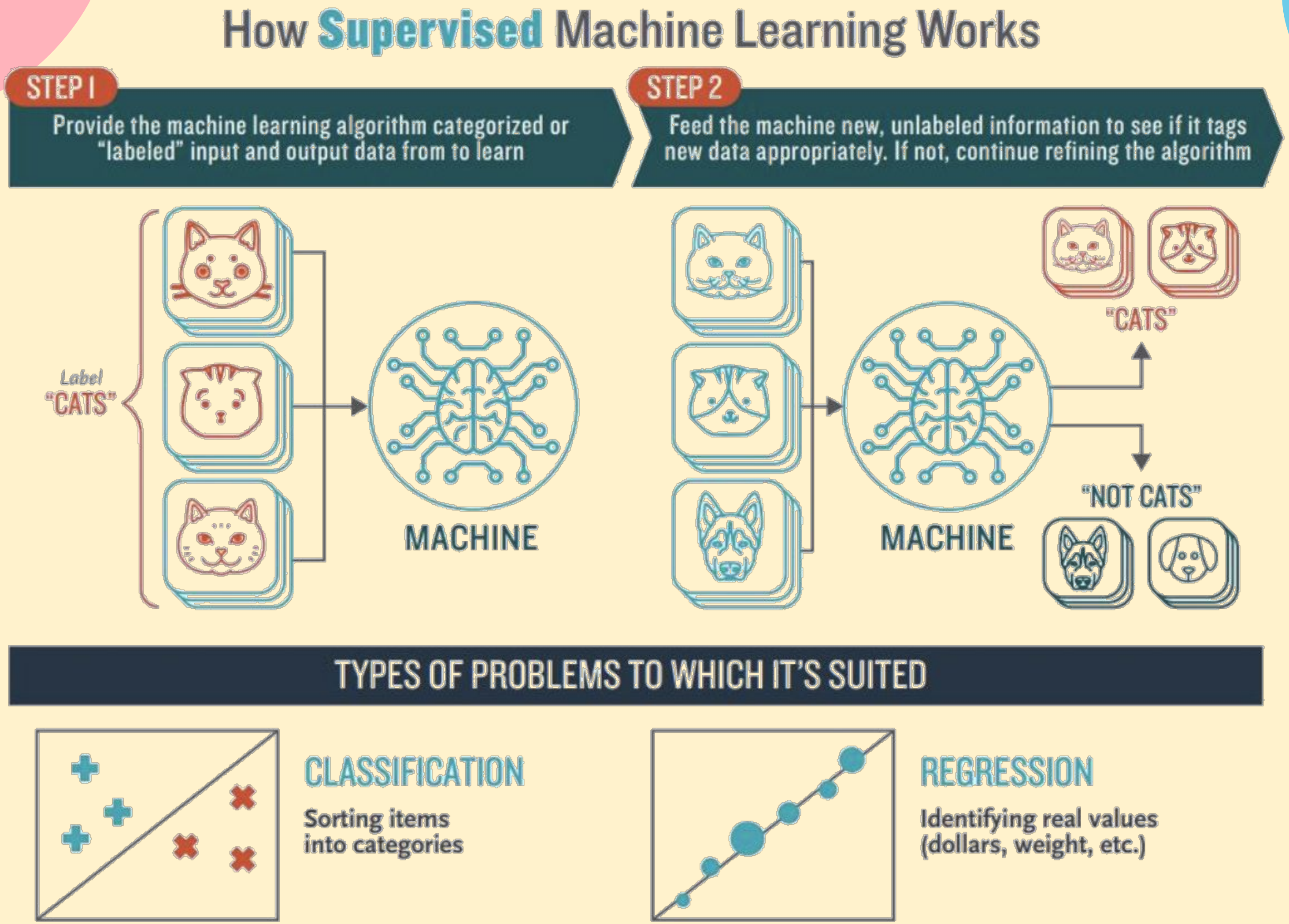
Capítol 1.2: Machine Learning

Aina Palacios

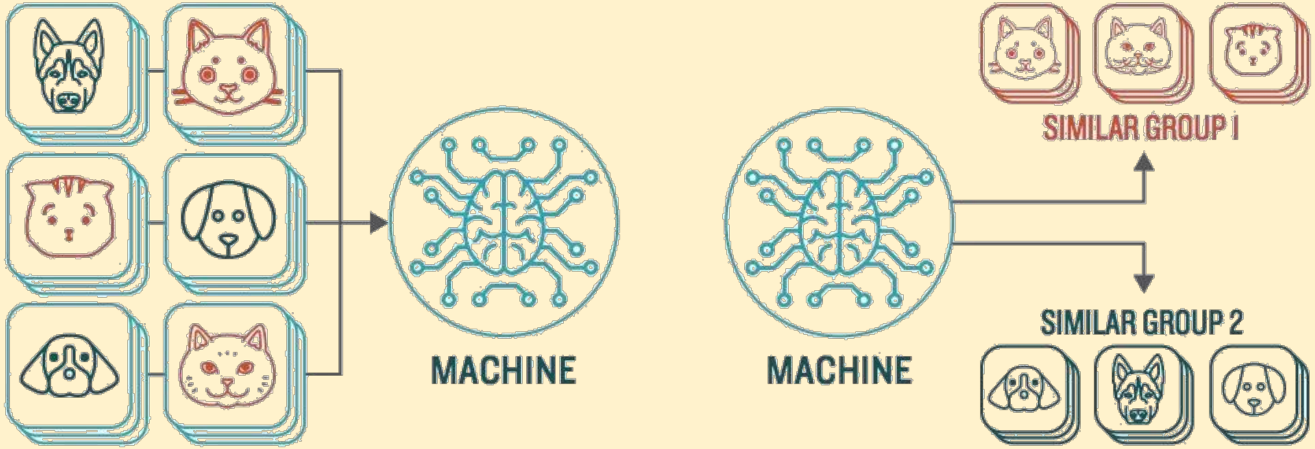

Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks


How Supervised Machine Learning Works
STEP 1
Provide the machine learning algorithm categorized or "labeled" input and output data from to learn
STEP 2
Feed the machine new, unlabeled information to see if it tags new data appropriately. If not, continue refining the algorithm
Label "CATS"
MACHINE
MACHINE
"CATS"
"NOT CATS"
TYPES OF PROBLEMS TO WHICH IT'S SUITED
CLASSIFICATION
Sorting items into categories
REGRESSION
Identifying real values (dollars, weight, etc.)

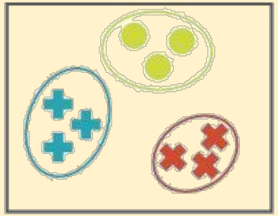
How Unsupervised Machine Learning Works

STEP 1
Provide the machine learning algorithm uncategorized, unlabeled input data to see what patterns it finds

STEP 2
Observe and learn from the patterns the machine identifies



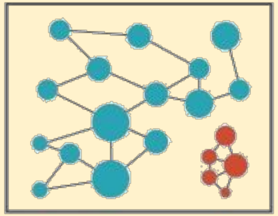
TYPES OF PROBLEMS TO WHICH IT'S SUITED



CLUSTERING

Identifying similarities in groups

For Example: Are there patterns in the data to indicate certain patients will respond better to this treatment than others?



ANOMALY DETECTION

Identifying abnormalities in data

For Example: Is a hacker intruding in our network?

Entorn de treball

Pandas

Per poder treballar la base de dades

Llibreria Sklearn

Per poder aplicar Machine Learning així com preprocessats

Com aplicar Machine Learning

1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. Procés enginyeria
4. Determinar train i test
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. Modificar paràmetres

Com aplicar Machine Learning

1. **Determinar objectiu i estudi de les dades**
2. Preprocessat
3. Procés enginyeria
4. Determinar train i test
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. Modificar paràmetres

1. Determinar Objectiu i estudi de les dades

Primerament, hem de determinar quin és l'objectiu que estem perseguint. A vegades hem de fixar-nos que les mostres recollides contenen informació a priori i posterior de l'esdeveniment:

- Per exemple, si busquem si un paquet arribarà tard al seu destí i tenim una base de dades amb tot el recorregut del paquet, la informació a posteriori de l'entrega, així com els motius de retard, són dades que no podem utilitzar, ja que no són indicadors prioris del retard de l'entrega.

El nostre objectiu pot ser classificar, predir una regressió o aplicar una tècnica de clústering.

Per poder entendre millor el que estem fent, és important fer un estudi de totes les característiques de la meva mostra. D'aquesta manera, podem aplicar **preprocessats** de les mostres per ajudar als algoritmes a treballar millor.

Anàlisi descriptiu i gràfic

Com aplicar Machine Learning

1. Determinar objectiu i estudi de les dades
2. **Preprocessat**
3. Procés enginyeria
4. Determinar train i test
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. Modificar paràmetres

2. Preprocessat

És important netejar les dades abans d’aplicar cap algoritme sobre elles

1. **Dades no existents**: Nans, Unknown, Null, ext.
    a. Si són numèriques podem posar a 0, fer interpolació, mitjana,...
    b. Si són categòriques podem posar un identificador
    c. Si no es poden substituir, hauriem d’eliminar les dades
2. **Dades anòmales**: Hi ha dades que poden no tenir sentit, aplicar el coneixement previ. Un outlier no és una dada anòmala.

https://scikit-learn.org/stable/modules/preprocessing.html

Com aplicar Machine Learning

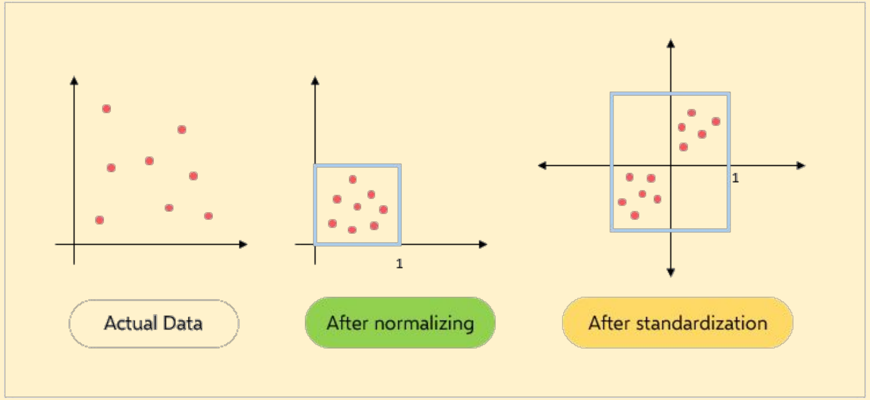
1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. **Procés enginyeria**
4. Determinar train i test
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. Modificar paràmetres

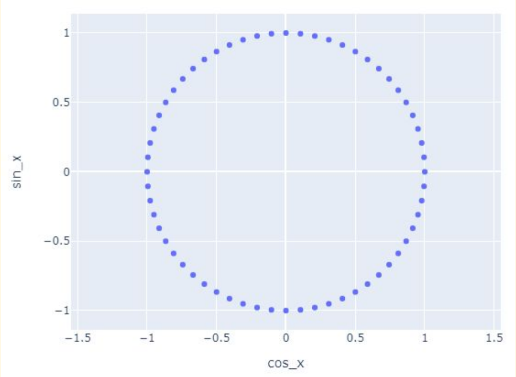
3. Procés enginyeria

No aplicar procés sobre el target o invertir el procés

3. **Procés d'enginyeria**: Existeixen diferents transformacions que ajuden al model a predir millor.

   a. Si són gaussianes -> Estandardització
   b. Si la seva distribució no és normal -> Normalització
   c. Si conté outliers -> RobustScaler o altres
   d. Si són categòriques -> Dummys o enumeració
   e. Altres: Polimorfisme, transformacions, cícliques,...





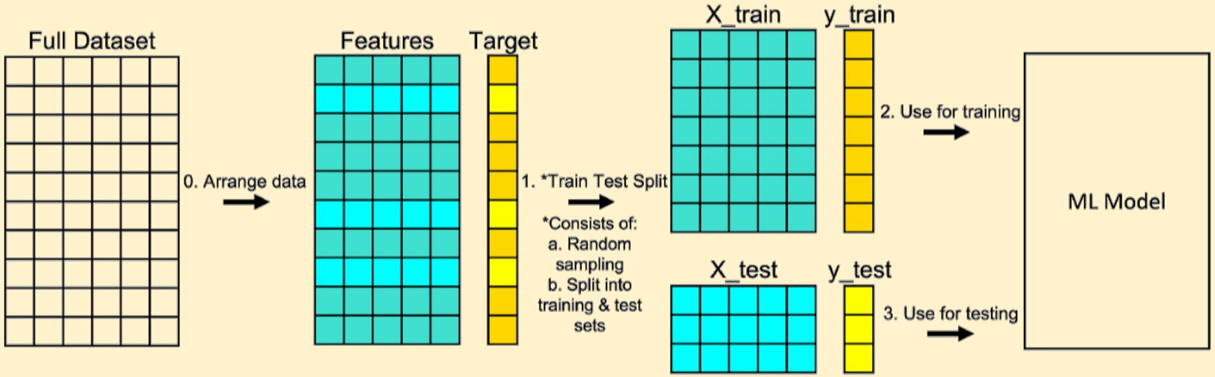
https://scikit-learn.org/stable/modules/preprocessing.html

Com aplicar Machine Learning

1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. Procés enginyeria
4. **Determinar train i test**
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. Modificar paràmetres

4. Determinar train i test

Si treballem amb Supervised Machine Learning, necessitem dividir les dades entre train i test per tal de poder determinar el millor **model** per aconseguir el nostre objectiu.



El test l'utilitzarem per veure com de bé el model funciona. Normalment representa un 30% o 20% de les dades.

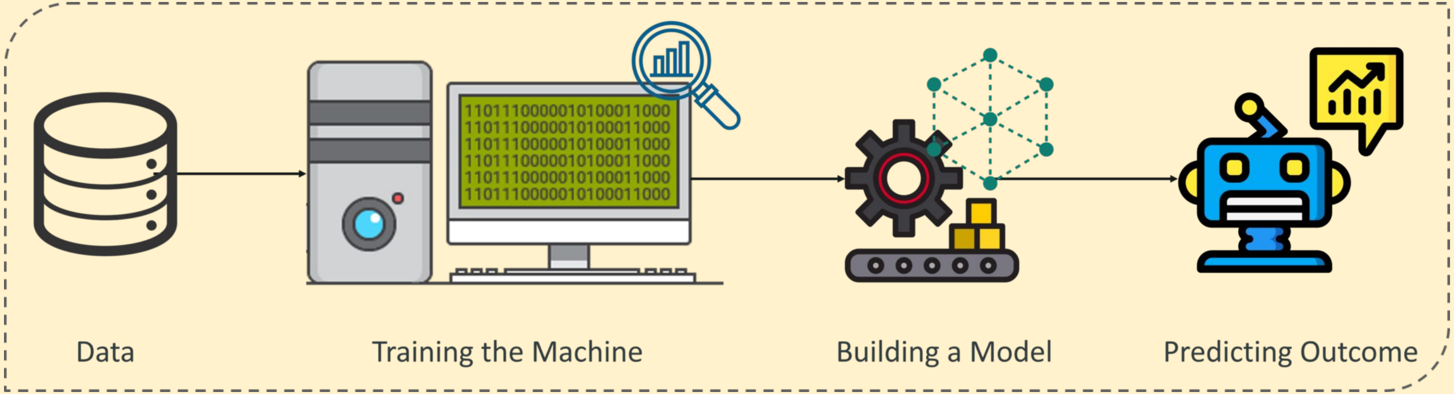
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Com aplicar Machine Learning

1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. Procés enginyeria
4. Determinar train i test
5. **Escollir i entrenar el model**
6. Mètriques i validació creuada
7. Modificar paràmetres

Machine Learning

- Enllaç a curs de Machine Learning
- Enllaç Machine Learning amb Sklearn
- Cheeting sheet Machine Learning amb Sklearn



Classification

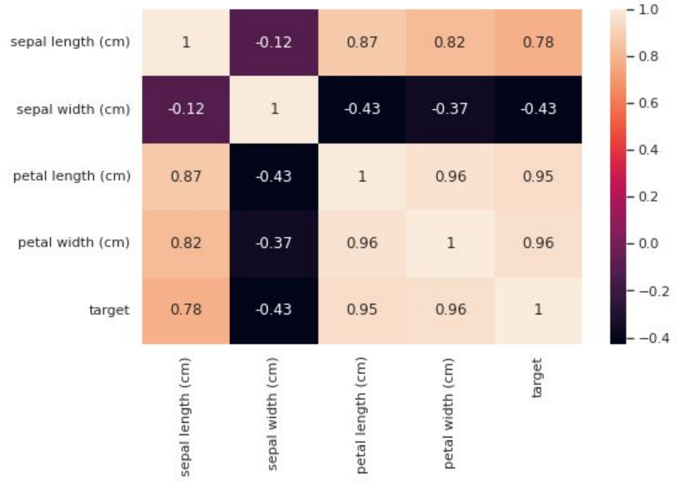
```python
from sklearn import neighbors, datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
```

```python
import pandas as panda
import numpy as np

df= pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                 columns= iris['feature_names'] + ['target'])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

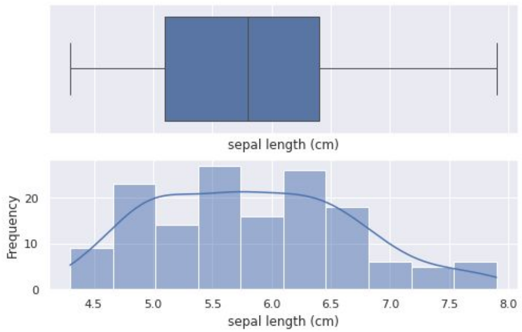
```python
corr = df.corr()
sns.heatmap(corr, annot=True)
```

<AxesSubplot:>

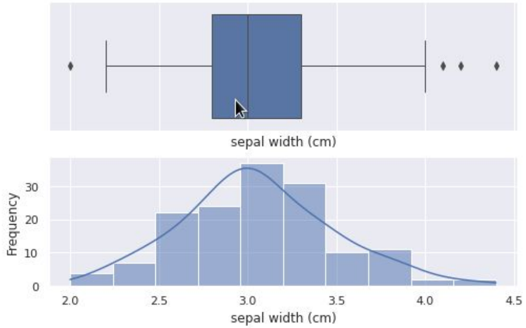


Classification

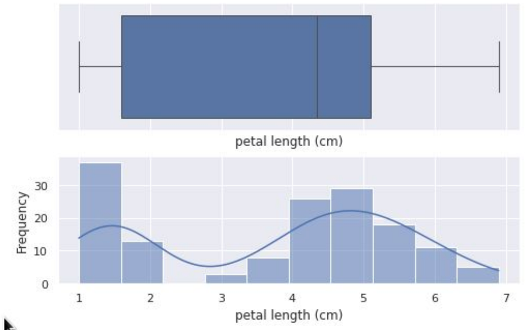
```python
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

for i in df.columns:
    plt.figure()
    plt.tight_layout()
    sns.set(rc={"figure.figsize":(8, 5)})
    f, (ax_box, ax_hist) = plt.subplots(2, sharex=True)
    plt.gca().set(xlabel= i,ylabel='Frequency')
    sns.boxplot(df[i],  ax=ax_box , linewidth= 1.0)
    sns.histplot(df[i], ax=ax_hist , bins = 10,kde=True)
```



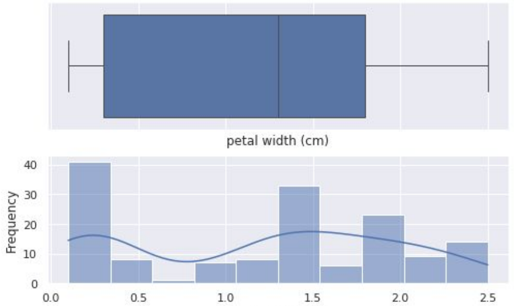
<Figure size 576x360 with 0 Axes>

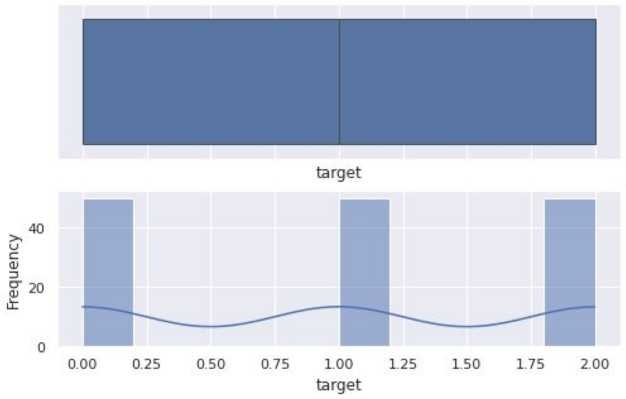


<Figure size 576x360 with 0 Axes>



<Figure size 576x360 with 0 Axes>





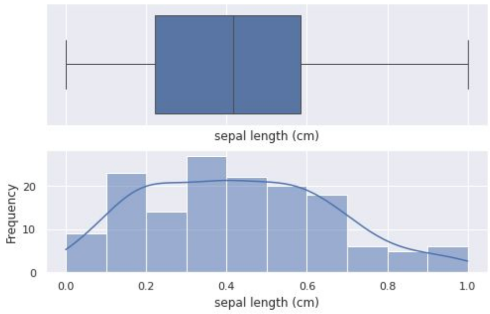
Classification

1.2 Preprocessat

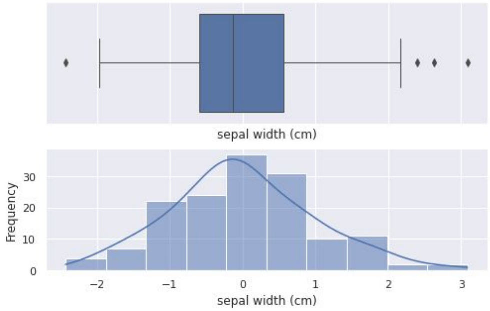
```
df.isnull().values.any()
```

```
False
```

> No tenim valors nuls i considerem que no existeixen dades anomeles




```
<Figure size 576x360 with 0 Axes>
```



> - La distribució de sepal width és normal, per tant aplicarem estandarització
> - Les altres variables no contenen outliers, per tant utilitzarem normalització

```python
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

standColumns = ['sepal width (cm)']
scalerStand = preprocessing.StandardScaler().fit(df[standColumns])
df[standColumns] = scalerStand.transform(df[standColumns])

normColumns = ['sepal length (cm)', 'petal length (cm)','petal width (cm)']
scalerNorm = preprocessing.MinMaxScaler().fit(df[normColumns])
df[normColumns] = scalerNorm.transform(df[normColumns])
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

for i in df.columns:
    plt.figure()
    plt.tight_layout()
    sns.set(rc={"figure.figsize":(8, 5)})
    f, (ax_box, ax_hist) = plt.subplots(2, sharex=True)
    plt.gca().set(xlabel= i,ylabel='Frequency')
    sns.boxplot(df[i],  ax=ax_box , linewidth= 1.0)
    sns.histplot(df[i], ax=ax_hist , bins = 10,kde=True)
```

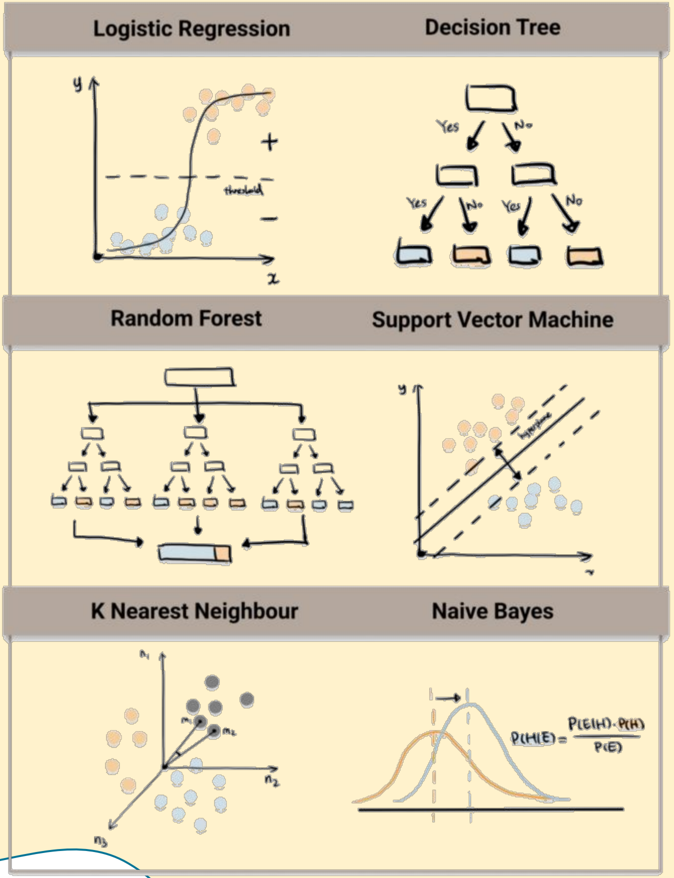
Classification

1.3 Test/train

```
from sklearn.model_selection import train_test_split

X = df.drop(['target'],axis=1)
y = df['target']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

Hem utilitzat un test del 20% perquè tenim poques mostres

Classification



1.4 Models

```python
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state=42)
```

```python
from sklearn.svm import SVC
svc = SVC(kernel='linear')
```

```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=2, random_state=42)
```

Training the models

```python
lr.fit(X_train, y_train)
svc.fit(X_train, y_train)
rf.fit(X_train, y_train)
```

...

```python
y_pred_lr = lr.predict(X_test)
y_pred_svc = svc.predict(X_test)
y_pred_rf = rf.predict(X_test)
```

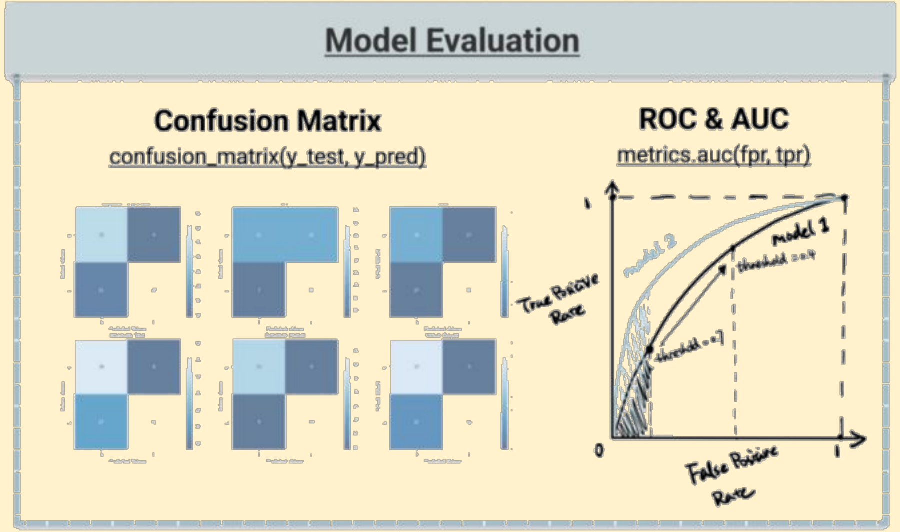
```python
y_pred_rf
```

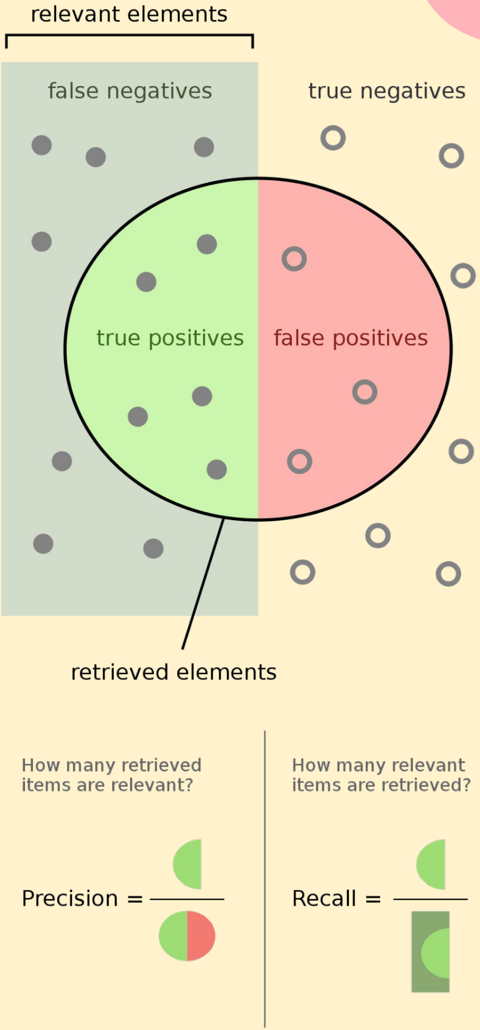
```
array([2, 1, 2, 0, 2, 1, 2, 0, 0, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 2, 2, 1,
       1, 2, 2, 2, 1, 2, 1, 0])
```

Com aplicar Machine Learning

1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. Procés enginyeria
4. Determinar train i test
5. Escollir i entrenar el model
6. **Mètriques i validació creuada**
7. Modificar paràmetres

Classification





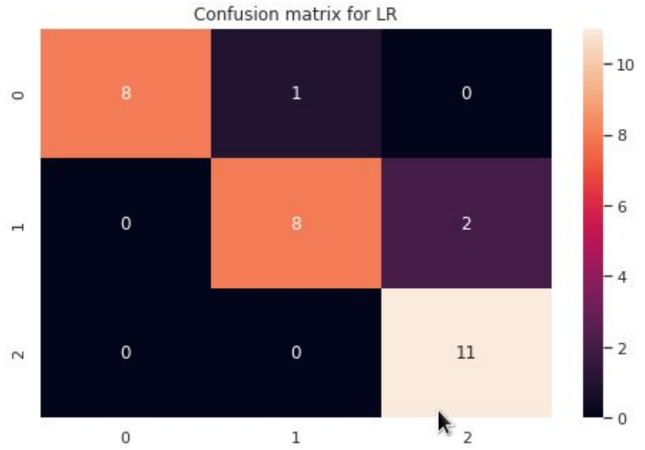
Classification

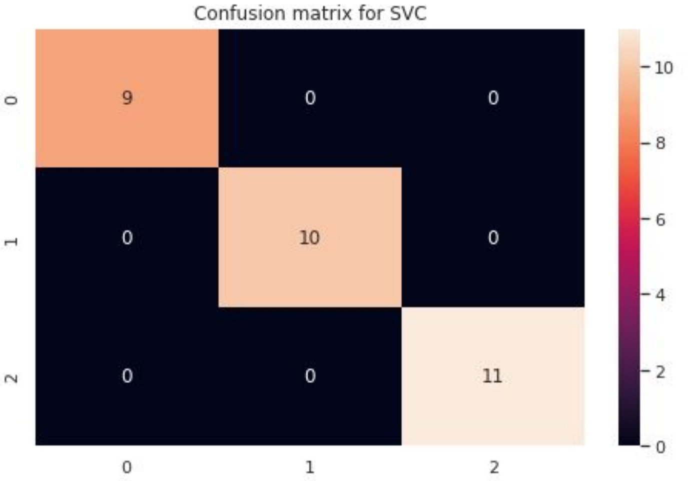
5 Evaluació

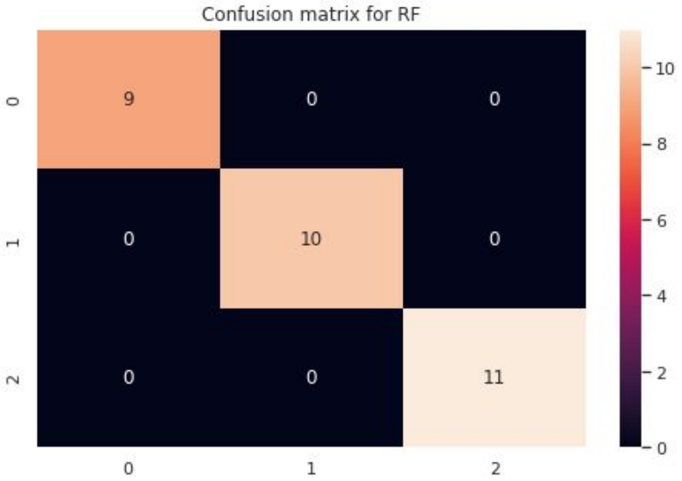
```python
from sklearn.metrics import confusion_matrix
cf_matrix_lr = confusion_matrix(y_test, y_pred_lr)
cf_matrix_svc = confusion_matrix(y_test, y_pred_svc)
cf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
```

```python
sns.heatmap(cf_matrix_lr, annot=True).set(title='Confusion matrix for LR')
```

```
[Text(0.5, 1.0, 'Confusion matrix for LR')]
```





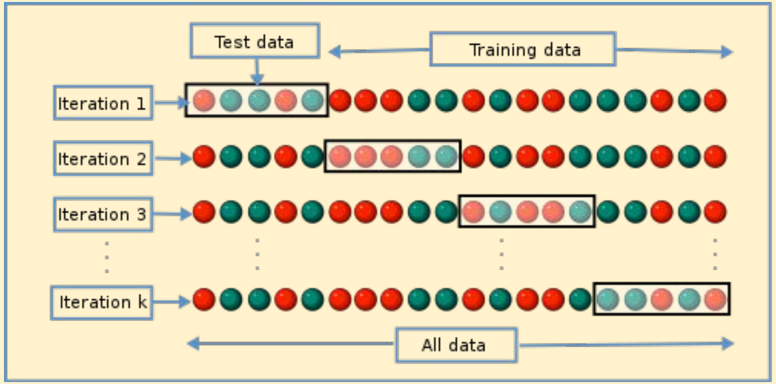



```python
from sklearn.metrics import f1_score
f1_lr = f1_score(y_test, y_pred_lr, average='macro')
f1_svc = f1_score(y_test, y_pred_svc, average='macro')
f1_rf = f1_score(y_test, y_pred_rf, average='macro')
print("F1 for LR: {:.2f}, SVC: {:.2f}, RF: {:.2f}".format(f1_lr, f1_svc, f1_rf) )
```

```
F1 for LR: 0.90, SVC: 1.00, RF: 1.00
```

Validació creuada

Quan creem el test/train pot donar la casualitat que el nostre set afavoreix el nostre model. Per tal de trobar el valor real dels nostres paràmetres, utilitzem CrossValidation.



Cross Validation

```
from sklearn.model_selection import cross_val_score

print(cross_val_score(lr, X, y, cv=5, scoring='f1_macro'))
```

Validació creuada

Classification

```
cv_lr = cross_val_score(lr, X, y, cv=5, scoring='f1_macro')
print("F1 for LR mean: {:.2f}, std: {:.2f}".format(cv_lr.mean(), cv_lr.std()) )

cv_svc = cross_val_score(svc, X, y, cv=5, scoring='f1_macro')
print("F1 for SVC mean: {:.2f}, std: {:.2f}".format(cv_svc.mean(), cv_svc.std()) )

cv_rf = cross_val_score(rf, X, y, cv=5, scoring='f1_macro')
print("F1 for RF mean: {:.2f}, std: {:.2f}".format(cv_rf.mean(), cv_rf.std()) )
```

```
F1 for LR mean: 0.92, std: 0.05
F1 for SVC mean: 0.96, std: 0.03
F1 for RF mean: 0.95, std: 0.03
```

Com aplicar Machine Learning

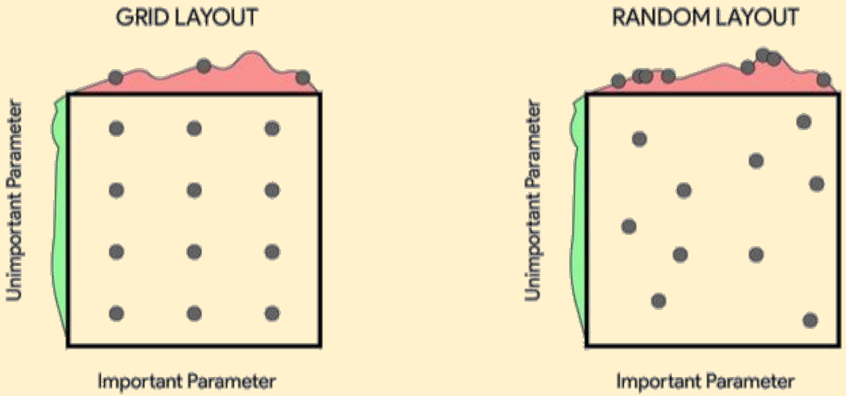
1. Determinar objectiu i estudi de les dades
2. Preprocessat
3. Procés enginyeria
4. Determinar train i test
5. Escollir i entrenar el model
6. Mètriques i validació creuada
7. **Modificar paràmetres**

7. Modificar paràmetres

El model conté diferents **paràmetres** que modifiquen el seu comportament.

És important intentar trobar aquells paràmetres que donen els millors resultats.

Podem intentar trobar-los directament o utilitzant el **GridSearch** o un **Randomized Parameter Optimization**, que ens provarà totes les combinacions possibles que li diguem i trobarà el millor resultat.



Classification

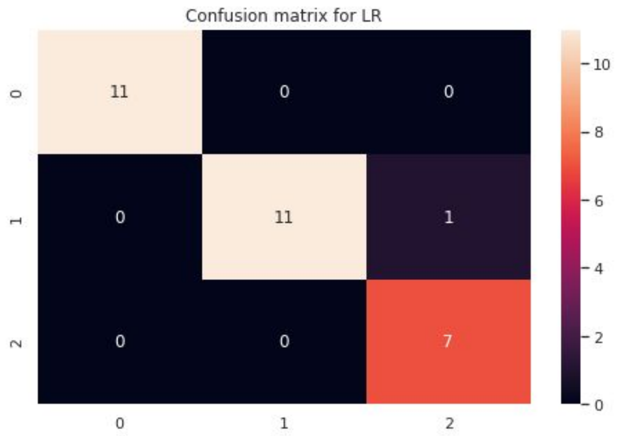
6. Millors paràmetres

```
from sklearn.model_selection import RandomizedSearchCV

params = {"solver": ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], "penalty": ["l1", "l2", "elasticnet"]}
rsearch = RandomizedSearchCV(estimator=lr,
                             param_distributions=params,cv=5,
                             n_iter=14,random_state=42,scoring='f1_macro')
rsearch.fit(X, y)
print(rsearch.best_score_)
print(rsearch.best_params_)
```

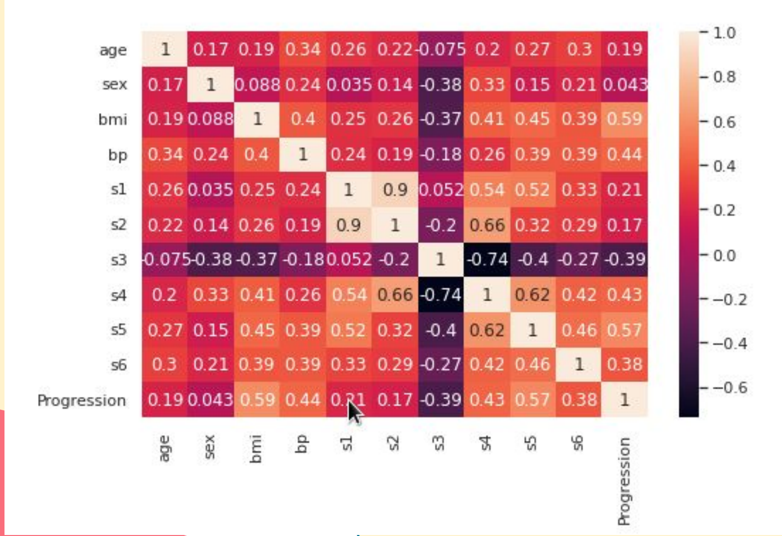
```
0.9530138477506899
{'solver': 'saga', 'penalty': 'l1'}
```

```
y_pred_RS_ls = LogisticRegression(random_state=42, solver = 'saga', penalty = 'l1')\
    .fit(X_train,y_train).predict(X_test)
cf_matrix_RS_lr = confusion_matrix(y_test, y_pred_RS_ls)
sns.heatmap(cf_matrix_RS_lr, annot=True).set(title='Confusion matrix for LR')
```

```
[Text(0.5, 1.0, 'Confusion matrix for LR')]
```



Regression



Machine Learning

Regression

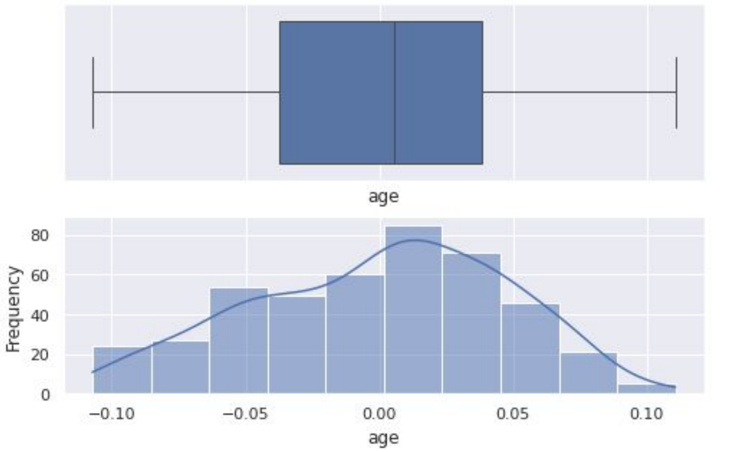
Utilitzarem la base de dades Iris, que ens proporciona un Data Set sobre tipus de plantes i els seus pètals

```
from sklearn import neighbors, datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

diabetes = datasets.load_diabetes()
```

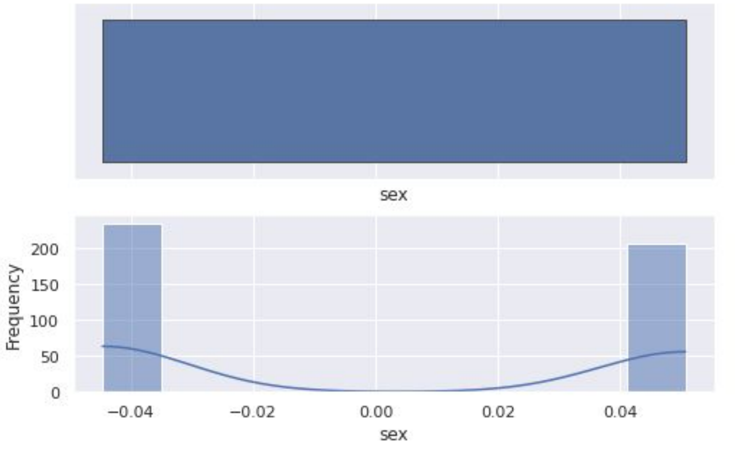
```
import pandas as panda
import numpy as np

df = panda.DataFrame(diabetes.data,columns=diabetes.feature_names)
df['Progression'] = diabetes.target

df.head()
```

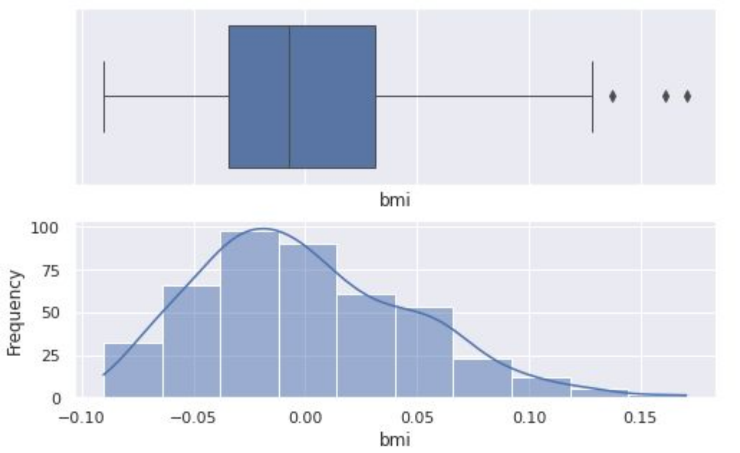
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	Progression
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0

Regression

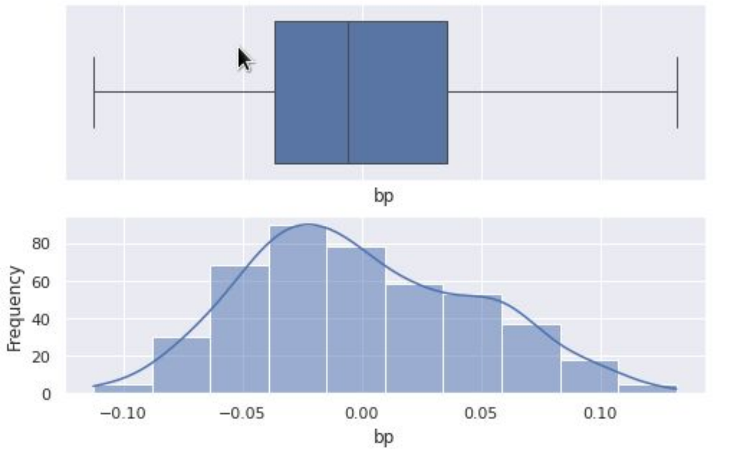


<Figure size 576x360 with 0 Axes>

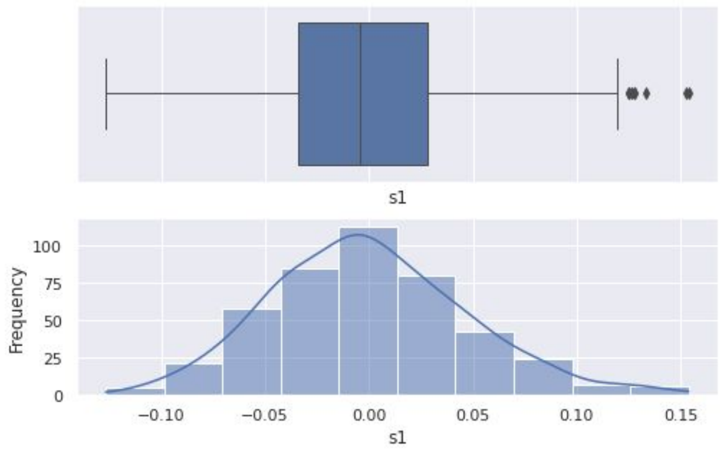




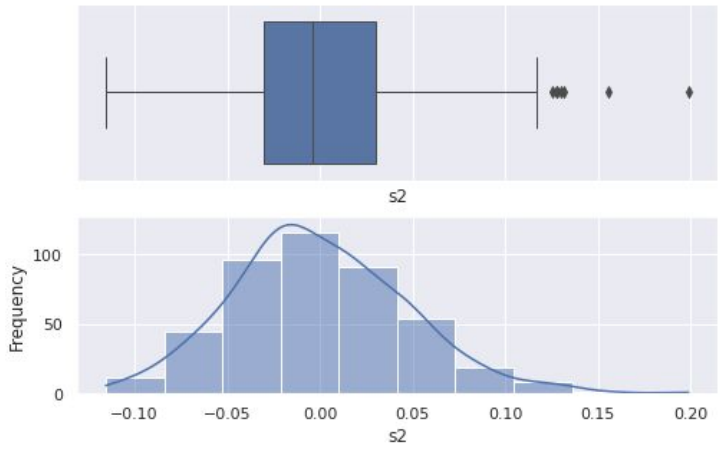
<Figure size 576x360 with 0 Axes>

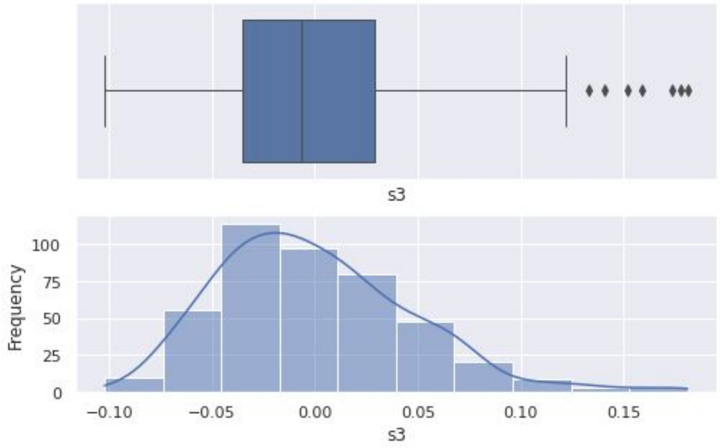


Regression

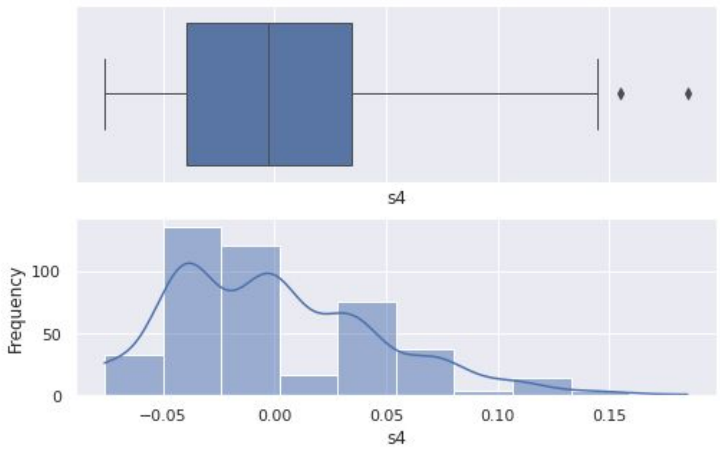


<Figure size 576x360 with 0 Axes>

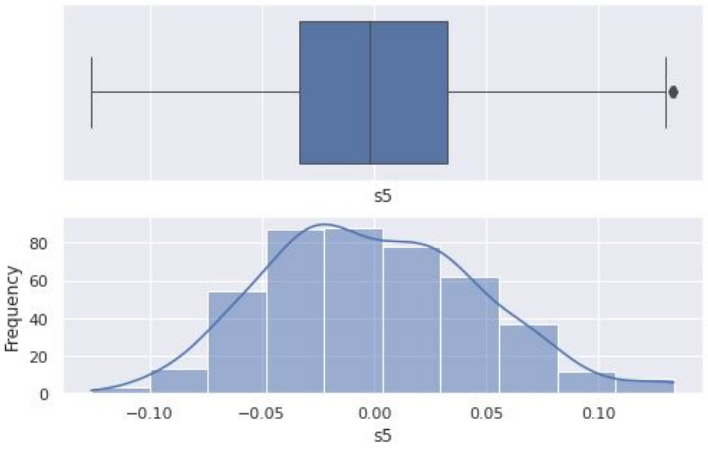




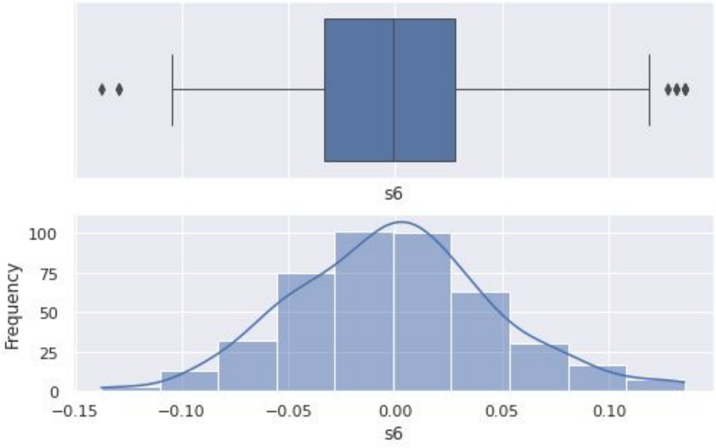
<Figure size 576x360 with 0 Axes>

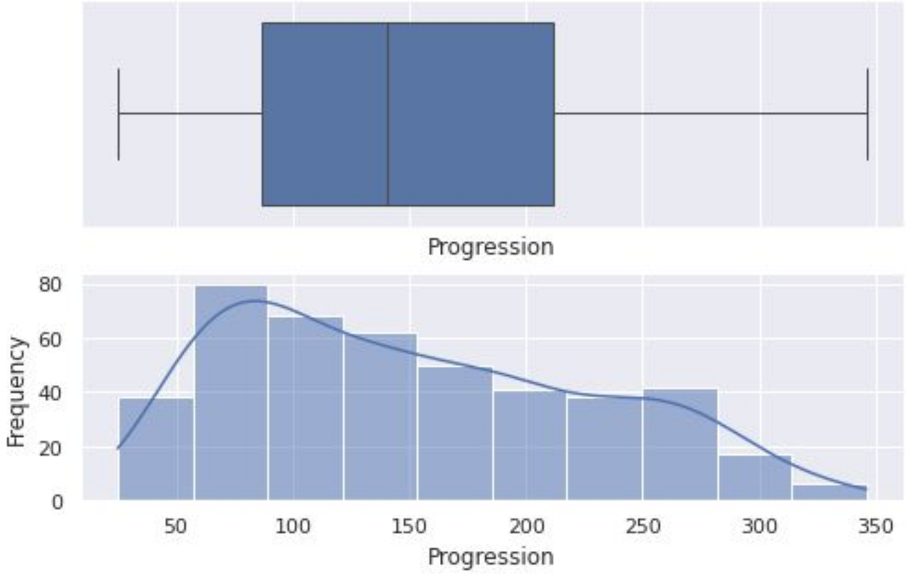


Regression



<Figure size 576x360 with 0 Axes>

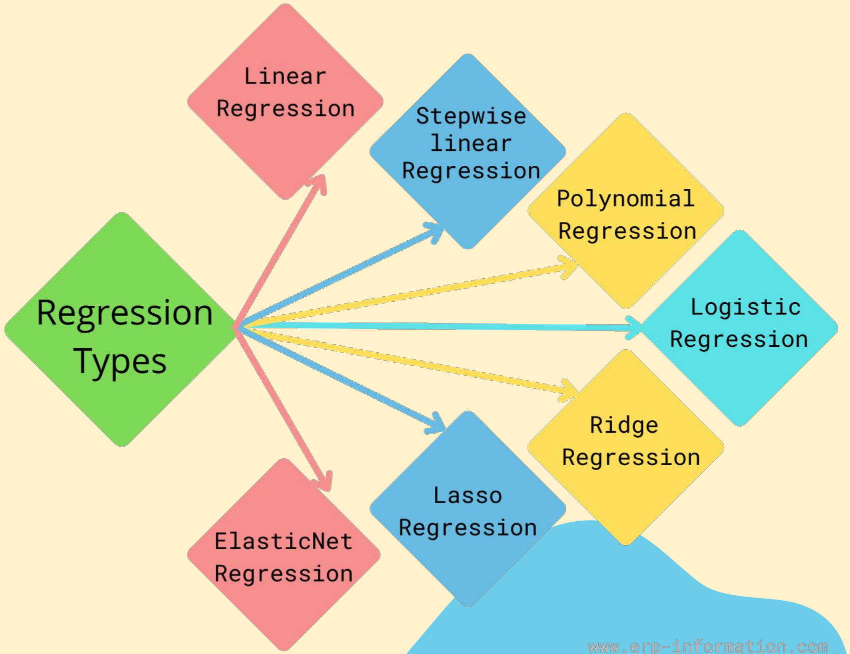




Regression

3 Test/train

```python
from sklearn.model_selection import train_test_split

X = df.drop(['Progression'],axis=1)
y = df['Progression']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```



4 Models

```python
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

```python
from sklearn.svm import SVR
svr = SVR()
```

Training the models

```python
lr.fit(X_train, y_train)
svr.fit(X_train, y_train)
```

...

```python
y_pred_lr = lr.predict(X_test)
y_pred_svr = svr.predict(X_test)
```

```python
y_pred_lr[1:5]
```

```
array([135.69448628, 196.53507727,  74.45375974,  87.28375355])
```

Regression

5 Evaluació

```
from sklearn.metrics import r2_score

r2_lr = r2_score(y_test, y_pred_lr)
r2_svr = r2_score(y_test, y_pred_svr)
```

```
from sklearn.metrics import mean_squared_error

mse_lr = mean_squared_error(y_test, y_pred_lr)
mse_svr = mean_squared_error(y_test, y_pred_svr)
```

```
metrics = panda.DataFrame( data = [['Linear Regression', r2_lr, mse_lr],
                                   ['Support Vector Machines', r2_svr, mse_svr]],
                                   columns = ['Model', 'R2', 'MSE'])
metrics.head()
```

	Model	R2	MSE
0	Linear Regression	0.502632	3289.974895
1	Support Vector Machines	0.157471	5573.138008

Cross Validation

```
from sklearn.model_selection import cross_val_score

lr = LinearRegression()
cv_lr = cross_val_score(lr, X, y, cv=5, scoring='r2')
cv_svr = cross_val_score(svr, X, y, cv=5, scoring='r2')
```

```
cv_lr_mean = [cv_lr.mean(), cv_svr.mean()]
metrics['R2 with CV'] = cv_lr_mean
metrics.head()
```

	Model	R2	MSE	R2 with CV
0	Linear Regression	0.255382	3761.752362	0.482316
1	Support Vector Machines	0.126417	4413.269770	0.161335

Regression

6. Millors paràmetres

```
from sklearn.model_selection import RandomizedSearchCV
param = {'kernel' : ('linear', 'poly', 'rbf', 'sigmoid'),'C' : [1,5,10],'degree' : [3,8],
         'coef0' : [0.01,10,0.5],'gamma' : ('auto','scale')}

lr = LinearRegression()
rsearch = RandomizedSearchCV(estimator=svr,
                             param_distributions=param,cv=5,
                             n_iter=10,random_state=42,scoring='r2')
rsearch.fit(X, y)
print(rsearch.best_score_)
print(rsearch.best_params_)
```

```
0.47480829897832244
{'kernel': 'linear', 'gamma': 'scale', 'degree': 3, 'coef0': 0.5, 'C': 10}
```

```
svr_bp = SVR(kernel='linear', gamma='scale',degree= 3,coef0= 0.5,C= 10).fit(X_train, y_train)
y_pred_svr_bp = svr_bp.predict(X_test)
```

```
metrics_bp = panda.DataFrame( data = [['SVR best params', r2_score(y_test, y_pred_svr_bp),
                          mean_squared_error(y_test, y_pred_svr_bp),
                          cross_val_score(svr_bp, X, y, cv=5, scoring='r2').mean()]],
                               columns = ['Model', 'R2', 'MSE', 'R2 with CV'])

metrics = metrics.append(metrics_bp, ignore_index=True)
```

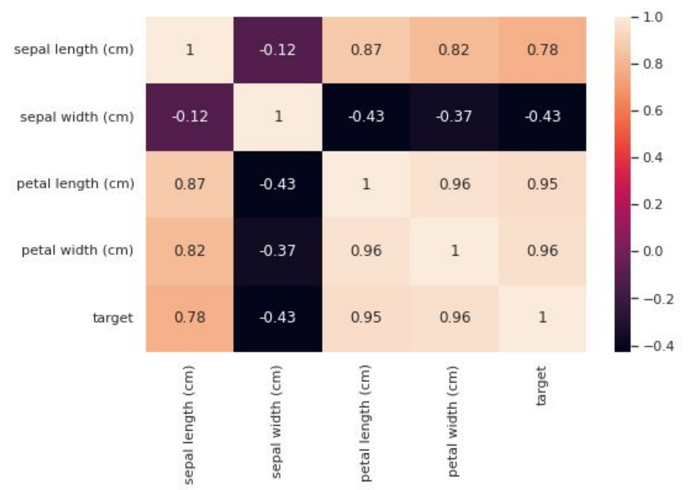
```
metrics.head()
```

	Model	R2	MSE	R2 with CV
0	Linear Regression	0.255382	3761.752362	0.482316
1	Support Vector Machines	0.126417	4413.269770	0.161335
2	SVR best params	0.273890	3668.252470	0.474808

Unsupervised

```python
corr = df.corr()
sns.heatmap(corr, annot=True)
```

<AxesSubplot:>




```python
from sklearn import neighbors, datasets, preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
```

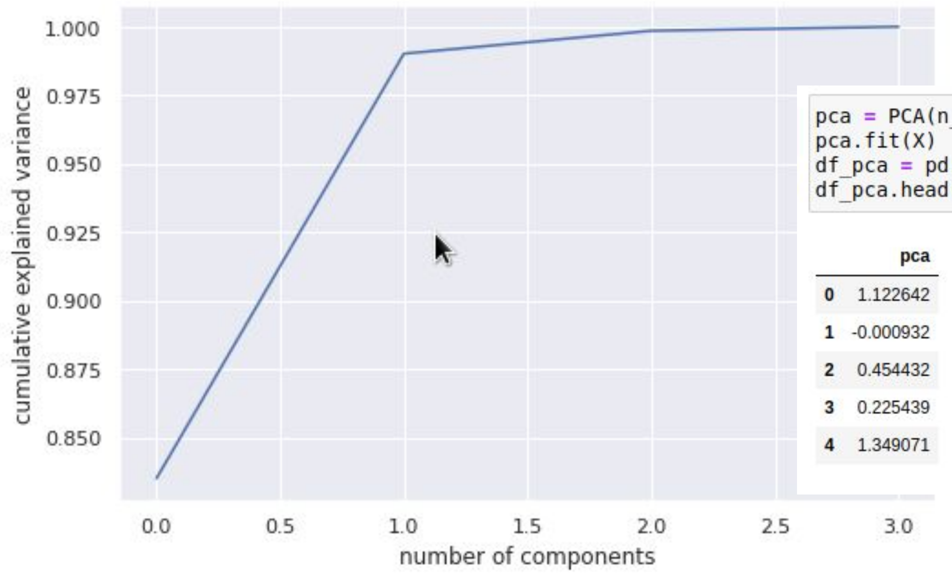
```python
import pandas as pd
import numpy as np

df= pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                 columns= iris['feature_names'] + ['target'])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

Unsupervised

3 PCA (Principal Component Analisis)

```
from sklearn.decomposition import PCA

X = df.drop(['target'],axis=1)
y = df['target']

pca = PCA().fit(X)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```




```
pca = PCA(n_components=1)
pca.fit(X)
df_pca = pd.DataFrame(pca.transform(X), columns=['pca'], index=df.index)
df_pca.head()
```

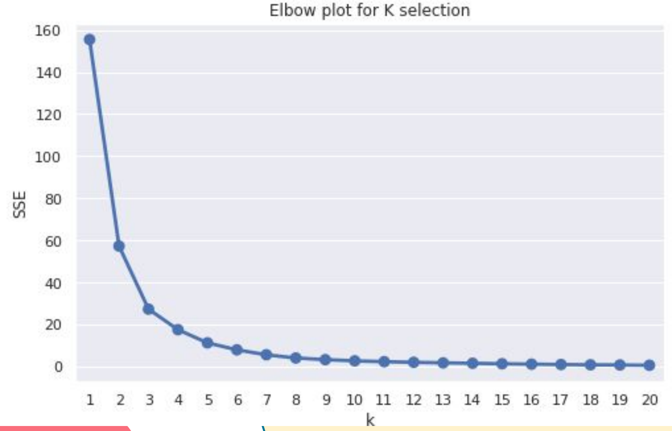
	pca
0	1.122642
1	-0.000932
2	0.454432
3	0.225439
4	1.349071

Unsupervised

4 Models

```python
from sklearn.cluster import KMeans
from kneed import KneeLocator

def elbow_plot(df):
    """Create elbow plot from normalized data"""
    sse = {}
    sse_r = []
    for k in range(1, 21):
        kmeans = KMeans(n_clusters=k, random_state=1)
        kmeans.fit(df)
        sse[k] = kmeans.inertia_
        sse_r.append(kmeans.inertia_)

    plt.title('Elbow plot for K selection')
    plt.xlabel('k')
    plt.ylabel('SSE')
    sns.pointplot(x=list(sse.keys()),
                  y=list(sse.values()))
    plt.show()
    return sse_r
```

```python
sse = elbow_plot(df_pca)
```




```python
kl = KneeLocator(range(1, 21), sse, curve="convex", direction="decreasing")
kl.elbow
```

```
4
```

```python
k_means = KMeans(n_clusters=4, random_state=42)
k_means.fit(X)
```

...

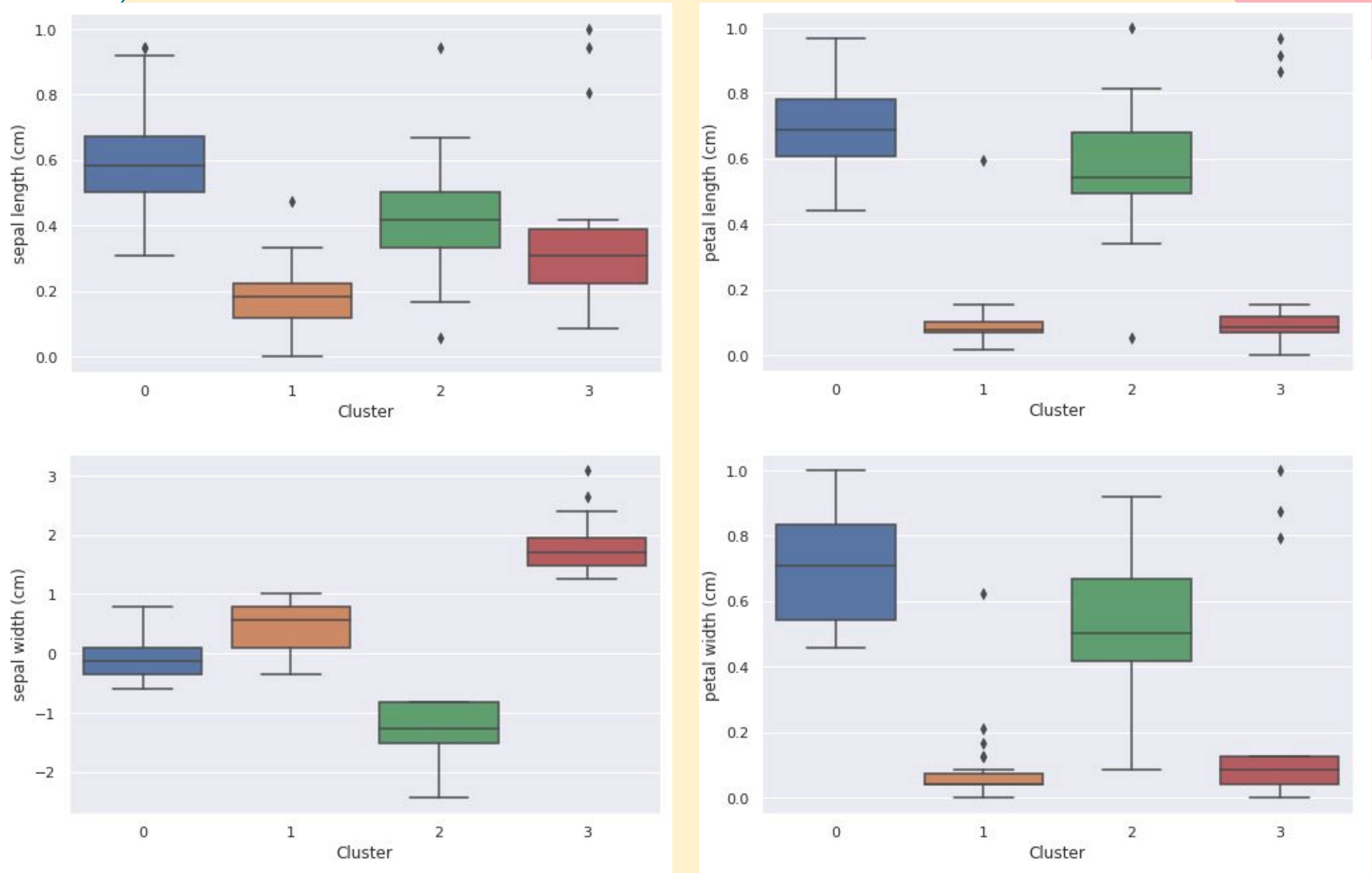
```python
y_pred = k_means.predict(X)
y_pred
```

```
array([1, 1, 1, 1, 3, 3, 1, 1, 1, 1, 3, 1, 1, 1, 3, 3, 3, 1, 3, 3, 1, 3,
       3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1,
       3, 1, 3, 1, 3, 1, 0, 0, 0, 2, 0, 0, 0, 2, 0, 2, 2, 0, 2, 0, 0, 0,
       0, 2, 2, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 0, 1, 0, 2,
       0, 2, 2, 0, 2, 2, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 2, 3,
       0, 2, 0, 2, 0, 0, 0, 3, 2, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 3,
       0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0], dtype=int32)
```

```python
df['Cluster'] = y_pred
df.head()
```

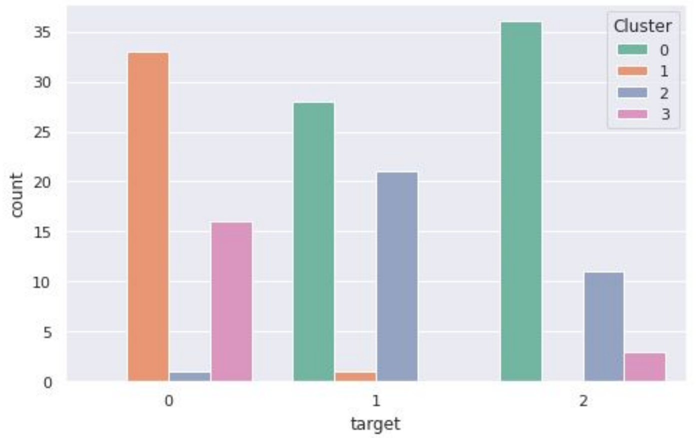
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	Cluster
0	0.222222	1.019004	0.067797	0.041667	0	1
1	0.166667	-0.131979	0.067797	0.041667	0	1
2	0.111111	0.328414	0.050847	0.041667	0	1
3	0.083333	0.098217	0.084746	0.041667	0	1
4	0.194444	1.249201	0.067797	0.041667	0	3

Unsupervised

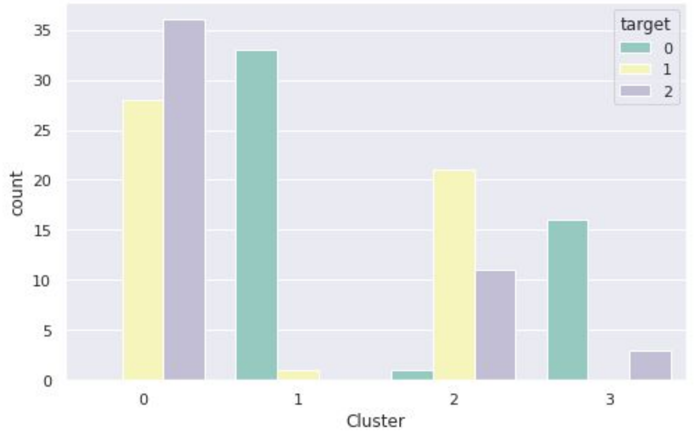


Unsupervised

```python
ax = sns.countplot(x="target", hue="Cluster", data=df, palette="Set2")
```




```python
ax = sns.countplot(x="Cluster", hue="target", data=df, palette="Set3")
```



Ja hem acabat la part 2!

Gràcies a tots!