

# Tema 1: Introducción a ML con Python

# Profesor

## Eduardo Fernández Carrión

Lead Data Scientist (STRATIO BD).

PhD (UCM), Métodos Estadístico-Matemáticos para el Tratamiento Computacional de la Información.

Ciencias Matemáticas (UCM) e Ingeniería Informática (URJC).

Email: [eduardofernandezcarrion@gmail.com](mailto:eduardofernandezcarrion@gmail.com)



# Tipos de Problemas

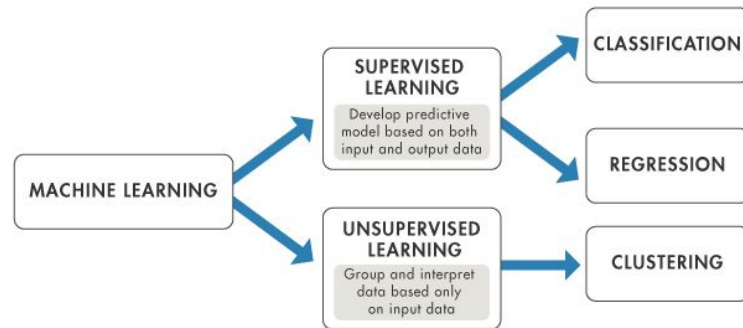
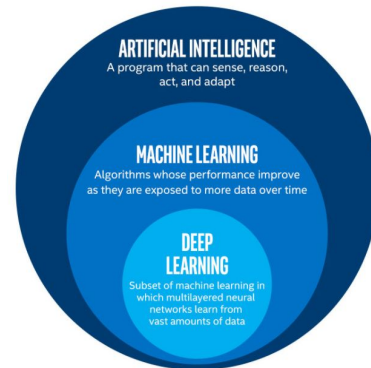
## Diferencias entre tipos de Problemas

Aprendizaje supervisado:

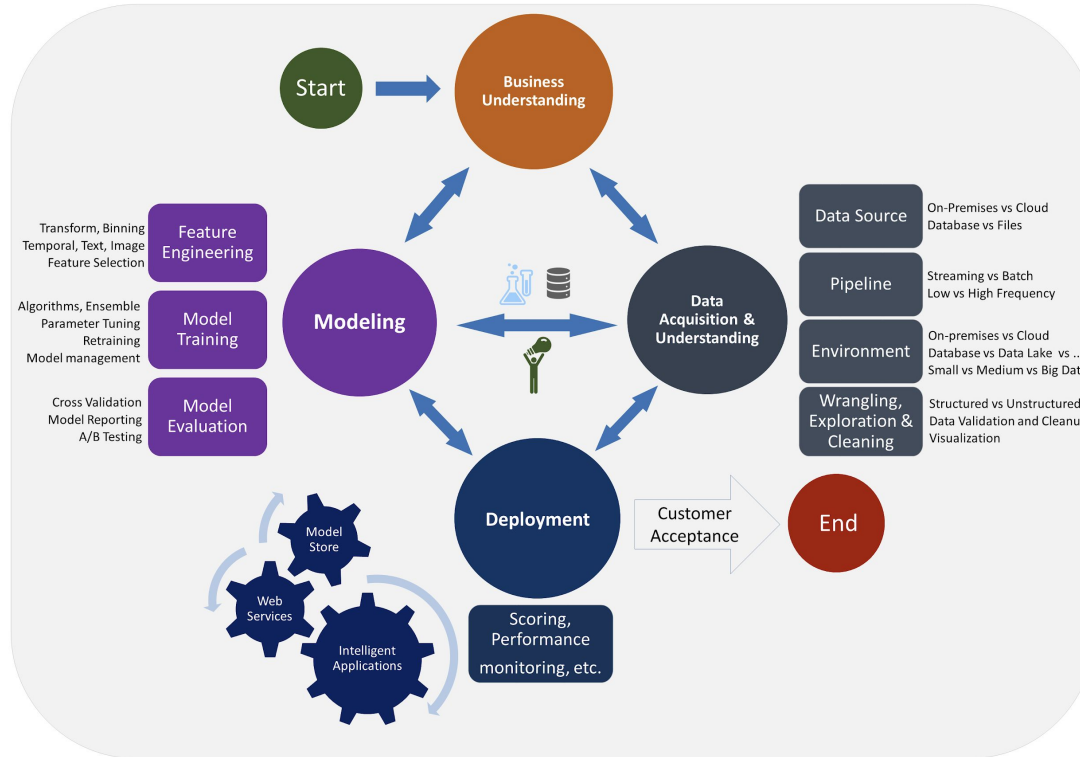
- Problema de **Clasificación**: Cuando la variable objetivo es categórica
- Problema de **Regresión**: Cuando la variable objetivo es numérica

Aprendizaje no supervisado:

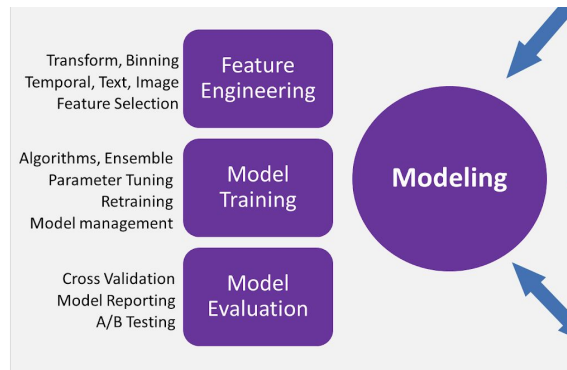
- **Clusterización**: Cuando no hay variable objetivo



# Data Science Lifecycle



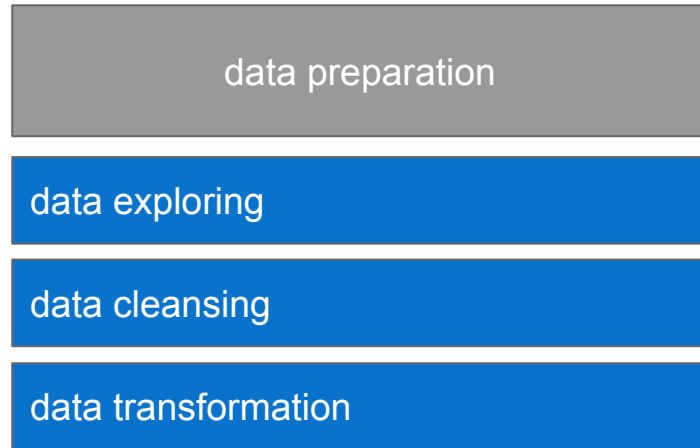
# Modeling Process



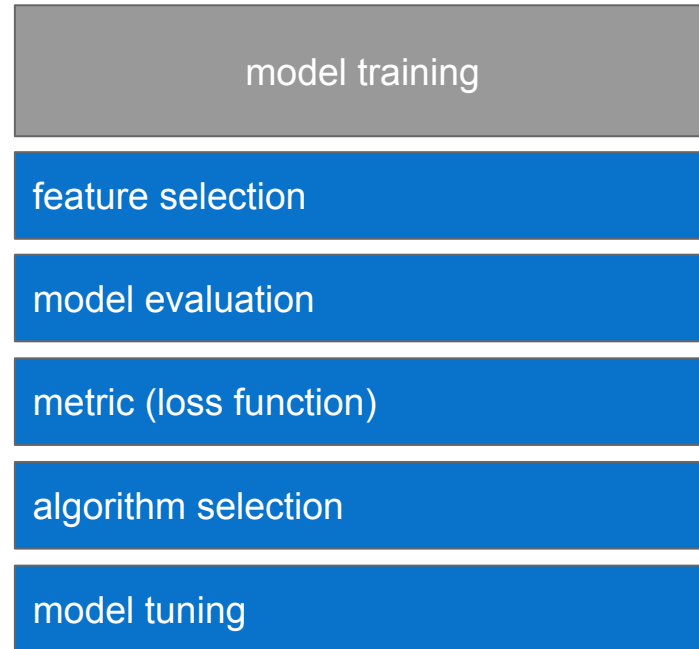
Para hacer un buen modelo de ML, una vez entendido el problema tenemos que:

1. Decidir cuáles serán las variables predictoras y la variable objetivo. Pero, por el camino, podría ser que
  - a. tuviéramos que limpiar datos, o que
  - b. tuviéramos que transformar algunas variables.
2. Probar diferentes algoritmos predictivos, teniendo en cuenta que existen,
  - a. diferentes parametrizaciones de cada modelo,
  - b. diferentes formas de entrenar un modelo
  - c. diferentes métricas de evaluación de un modelo

# Modeling Process



80% time



20% time



# SciPy

## Python Ecosystem for Data Science

pandas library:

[https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide)

numpy library:

<https://numpy.org/doc/stable/reference/index.html>

sklearn library:

<https://scikit-learn.org/stable/>

```
import pandas
import numpy
import sklearn
```

  
pandas

```
pandas.__version__
```

```
'0.25.3'
```

 NumPy

```
numpy.__version__
```

```
'1.18.1'
```

 scikit  
learn

```
sklearn.__version__
```

```
'0.24.0'
```

# Ejemplo End-to-End

## Clasificación

load data

fit model

save model

```
digits = datasets.load_digits()
features = digits.data
target = digits.target

model = LogisticRegression()
model.fit(features, target)
model.score(features, target)

dump(model, open('./models/2021-01-22-lr.pkl', 'wb'))
```



# Ejemplo End-to-End

## Clasificación

load data

split data

fit model

save model

```
digits = datasets.load_digits()
features = digits.data
target = digits.target

features_train, features_test, target_train, target_test = train_test_split(features, target)

model = LogisticRegression()
model.fit(features_train, target_train)
model.score(features_test, target_test)

dump(model, open('./models/2021-01-22-lr.pkl', 'wb'))
```

# Ejemplo End-to-End

## Clasificación

load data

split data

transform data

fit model

save model

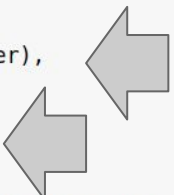
```
digits = datasets.load_digits()
features = digits.data
target = digits.target

features_train, features_test, target_train, target_test = train_test_split(features, target)

model = LogisticRegression()
standardizer = StandardScaler()
pipeline = Pipeline([('standardizer', standardizer),
                      ('lr', model)])

pipeline.fit(features_train, target_train)
pipeline.score(features_test, target_test)

dump(pipeline, open('./models/2021-01-22-lr-standscaler.pkl', 'wb'))
```



# Ejercicio

## Regresión

Cargar el dataset Boston:

<https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets>

Realizar dos modelos de regresión Lasso:

[https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear\\_model](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model)

- `model1`: Sin transformación de datos.
- `model2`: Con transformación `StandardScaler()`.



# Generación Números Aleatorios

## Seed

Muchos algoritmos necesitan, en algún momento, generar números aleatorios (basados o no en distribuciones conocidas).

El resultado de estos algoritmos, por tanto, puede ser distinto cada vez.

Para controlar que la generación de números aleatorios pueda replicarse, utilizamos semillas (seeds).

```
for i in range(5):  
    n = np.random.rand()  
    print(n)
```

```
0.5843822841126954  
0.9487017425250871  
0.31866667163743956  
0.10756070569052278  
0.5452390454746973
```

```
for i in range(5):  
    np.random.seed(10)  
    n = np.random.rand()  
    print(n)
```

```
0.771320643266746  
0.771320643266746  
0.771320643266746  
0.771320643266746  
0.771320643266746
```

# Generación Números Aleatorios

## Seed

Generalmente, en scikit-learn, se utiliza el parámetro *random\_state* para controlar las semillas.

```
digits = datasets.load_digits()
features = digits.data
target = digits.target

X_train, X_test, y_train, y_test = train_test_split(features, target, random_state=1)

model = LogisticRegression(random_state=1)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

0.9688888888888889



# Comparación de Algoritmos

En scikit-learn, hay una colección extensa de algoritmos para cada tipo de problema:

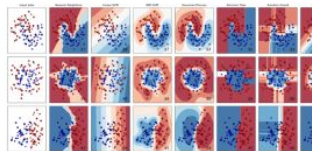
<https://scikit-learn.org/stable/index.html>

## Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, and more...



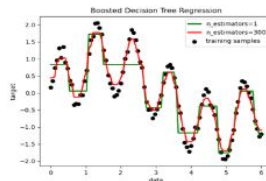
Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, nearest neighbors, random forest, and more...



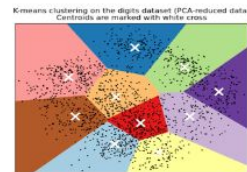
Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, and more...



Examples

# Comparación de Algoritmos

## Clasificación

Un primer trabajo del data scientists  
sería evaluar un conjunto de modelos.

```
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('DTC', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('RFC', RandomForestClassifier()))
models.append(('SVM', SVC()))

X_train, X_test, y_train, y_test = train_test_split(features, target)

for name, model in models:
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    print(name, ': ', str(score))

LR : 0.9688888888888889
LDA : 0.9666666666666667
KNN : 0.9933333333333333
DTC : 0.8644444444444445
NB : 0.8533333333333334
RFC : 0.98
SVM : 0.9888888888888889
```



# Ejercicio



## Regresión

- Cargar el dataset Boston.
- Train/Test split de los datos. Semilla 99.
- Evaluar, en este orden los siguientes algoritmos con semilla 99:
  - Linear Regression, Lasso, Random Forest, KNN, Decision Tree.



