TREE BASED MODELS

Decision-Tree (Classification)

No requiere estandarización →

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth = 2)
dt.fit(X_train, y_train
. .
```

Decision-Tree (Regression)

```
from sklearn.tree import DecisionTreeRegressor
dt = Dec... (max_depth = , min_samples_leaf = 0.1 . .)
```

Cada leaf debe tener al menos un 10% de los datos

Generalization Error

Cuanto generaliza el modelo en datos no vistos →

$bias^2 + varianza + irreducible\ error$ (irreducible error: ruido)

- Problemas de Varianza (overfitting) → CV error > training set error
  - Soluciones → decrease model complexity
  - gather more data
- Problemas de Bias (underfitting) → CV error ≈ training set error >> desired error
  - Soluciones → increase model complexity

Ensemble Learning

Usa varios modelos → agrega los resultados y los usa para hacer una prediccion

```
from sklearn.ensemble import VotingClassifier
mod1 . .
mod2 . .
mod3 . . .
classifiers = [('. . . ', mod1),
               ('. . . ', mod2),
               ('. . . ', dt)]
entrenamos los clasificadores
mod_ensamble = VotingClassifier(estimators = classifiers)
ve.fit(X_train, y_train)
```

Bagging (Bootstrap aggregation)

```
from sklearn.ensemble import BaggingClassifier
mod_base = . . .
bc = BaggingClassifier(base_estimator = mod_base , n_estimators = N , n_jobs = -1)
```

TREE BASED MODELS

Out of Bag Evaluation (OOB)

```
bc = BaggingClassifier(base_estimator..., n_estimator..., oob_score=True)
bc.oob_score_
```

Random Forests

Para clasificacion → `from sklearn.ensemble RandomForestClassifier`

" regresion → `from sklearn.ensemble import RandomForestRegressor`

```
rf = R... (n_estimators=...,
           min_sample_leaf=...,
           )
```

Para ver la importancia de los atributos en un modelo de arbol
↳ `rf.feature_importances_`

AdaBoost (Adaptative Boosting)

Para clasificacion → `from sklearn.ensemble import AdaBoostClassifier`
`mod = AdaB... (base_estimator=dt, n_estimators=N)`

→ Aprende de los errores del arbol anterior

" regresion → Igual pero AdaBoostRegressor

Gradient Boosting (Es lo mejor)

Para regression → `from sklearn.ensemble import GradientBoostingRegressor`
`mod = G... (n_estimators=N, max_depth=N)`

Para clasificacion → Igual pero GradientBoostingClassifier

Stochastic Gradient Boosting (más optimo) → Las samples son sin reemplazo

Por ello → Se usa `GradientBoostingRegressor(...., subsample=0.8, max_features=0.2)`

Son independientes

TREE BASED MODELS

Random Forest Hyperparameters (Complementare con il tuning)