SUPERVISED LEARNING

K-NN (Classification)

Para entrenarlo ->

```
from sklearn.neighbors import KNeighborsClassifier
X = ...
Y = ...
mod = KNeighborsClassifier(n_neighbors = N)
mod.fit()
```

-> Para medir ACCURACY ->

```
mod.score(X_test, Y_test)
```

Para hacer un split de los datos ->

```
from sklearn.model_selection import train_test_split
```

Linear Regression

```
sklearn.linear_model import LinearRegression
```

-> Para ver R^2 ->

```
mod.score(X_test, Y_test)
```

-> Para ver MSE ->

```
from sklearn.metrics import mean_squared_error
mean_s...(y_test, y_pred, squared=False)
```

-> Para RMSE

Cross_validation ->

```
from sklearn.model_selection import cross_val_score, KFold
kf = KFold(n_splits..., shuffle=True)
mod = Linear()
cv_result = cross_val_score(mod, X, y, cv=kf)
```

Con regularizaciones -> Ridge ->

```
from sklearn.linear_model import Ridge
mod = Ridge(alpha=alpha)
```

-> Lasso -> Igual que ridge peo Lasso.

Implementacion de métricas en mod de classificacion

Matriz de confusion ->

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

Para ver todas las métricas ->

```
from sklearn.metrics import classification_report
classification_report(y_test, y_pred
```

Logistic Regression (output probabilities)

Se importa de ->

```
from sklearn.linear_model import LogisticRegression
```

Para predecir probabilidades de cada clase ->

```
y_pred_probs = mod.predict_proba(X_test)[:,1]
```

Para ver la ROC curve ->

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)
```

-> Para el AUC ->

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, y_pred_probs)
```

-> A más cerca de 1 mejor

SUPERVISED LEARNING

Hyperparameter tuning

Grid search cross-val →
(Es muy costoso)

```
from sklearn.model_selection import GridSearchCV
 kf = KFold( ... )
param_grid = {'alpha' : np.arange ...          }
mod = Ridge()
mod_cv = GridSearchCV(mod, param_grid, cv=kf)
```

```
mod_cv.fit(X_train, y_train)
```

Para ver resultados → Mejores params → `mod_cv.best_params_`
↓
Mejor Resultado → `mod_cv.best_score_`

Random search cross-val →

```
from sklearn.model_selection import RandomizedSearchCV
    ⋮
mod_cv = Ra.... (mod, param_grid, cv=kf, n_iter=N)
```

Normalizando los datos

Escalar los datos →
(Standarizar)

```
from sklearn.preprocessing import StandardScaler
   ..
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Para hacer un pipeline →

```
from sklearn.pipeline import Pipeline

steps = [('scaler' = Stand..   ()),
         ('knn' : KNei..    (  )]

pipeline = Pipeline (steps)
modelo = pipeline.fit (X_train, y_train)
```

Evaluando multiples modelos

Cosas importantes → Tamaño del set de datos
→ Interpretabilidad
→ Flexibilidad

Pruebo varios modelos y hago crossval y despues un boxplot con los resultados