PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

AYUDANTÍA 3

Modelos Predictivos

Ayudante: Felipe Fuentes Porras

UTILIZAREMOS

DATAFRAME Y SERIES

DATAFRAME:

	Comuna	Manzana	Predial	Línea de construcción	Material estructural	Calidad construcción	Año construcción
0	9201	1	1	1	E	4	1940
1	9201	1	1	2	E	4	1960
2	9201	1	2	1	E	4	1930
3	9201	1	3	1	E	4	1960

Series: Columnas

ANALIZAR LOS DATOS

Tips:

- Analizar las distribuciones de los datos
- Graficar los datos y sus distribuciones

```
df.describe()
```

Da métricas sobre los datos generales de la tabla como mean, media, cuartiles, desviación, entre otros.

ALGUNOS FUNCIONES UTILES

Aggregation	Description
`count()`	Total number of items
`first()`, `last()`	First and last item
`mean()`, `median()`	Mean and median
`min()`, `max()`	Minimum and maximum
`std()`, `var()`	Standard deviation and variance
`mad()`	Mean absolute deviation
`prod()`	Product of all items
`sum()`	Sum of all items

LIMPIEZA Y DEPURACIÓN DE DATOS

- Revisar valores nulos

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

- Se pueden eliminar las filas o columnas con valores nulos. Se debe decidir en cada caso que resulta ser correcto.
- Se pueden completar los datos perdidos utilizando:
  - Promedios
  - Probabilidad
  - Reconociendo clusters
  - Reconocer patrones

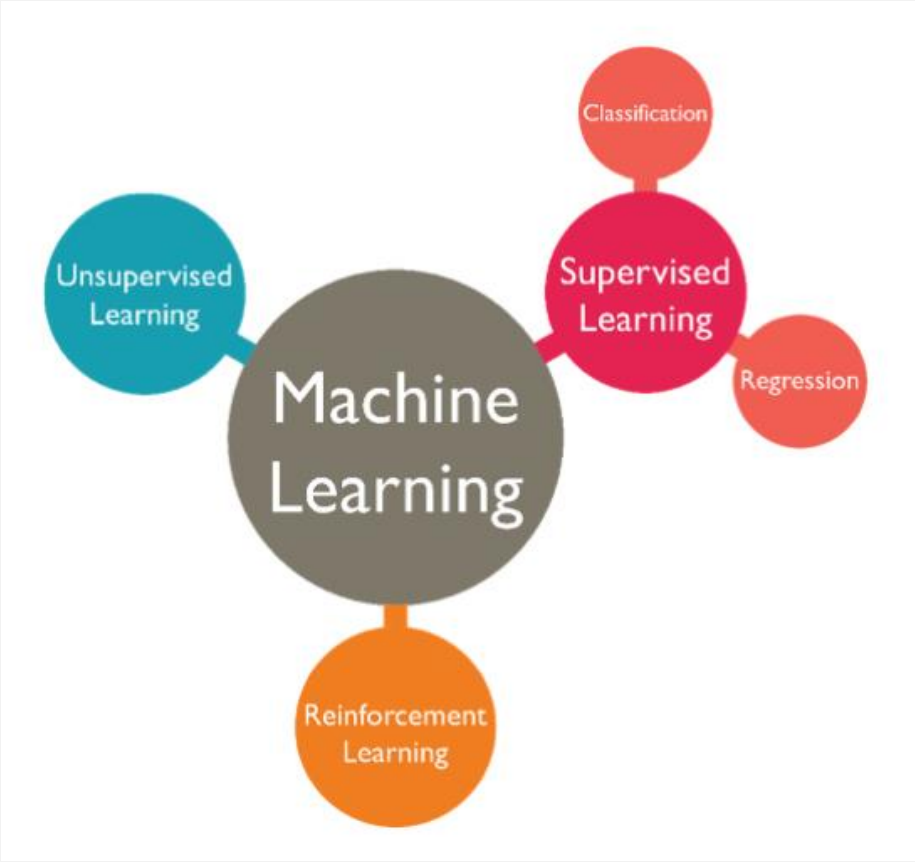
Lo importante es entender bien los datos y justificar porque ese eligió reemplazarlo.

DATOS EXTREMOS

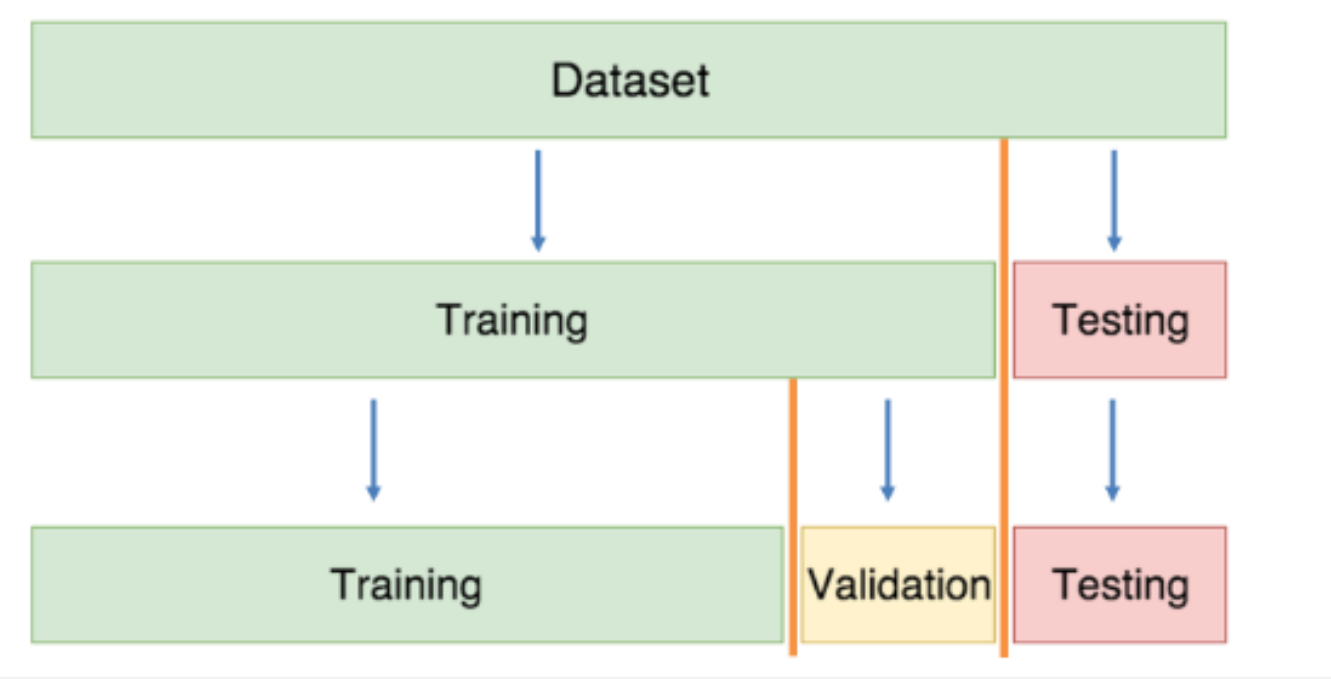
- No siempre es correcto eliminarlos.
- Aplicar funciones para que los casos extremos no afecten la modelación: función logarítmica

```
df['LoanAmount_log'] = np.log(df['LoanAmount'])
df['LoanAmount_log'].hist(bins=20)
plt.show()
```

MODELOS PREDICTIVOS: SKITLEARN

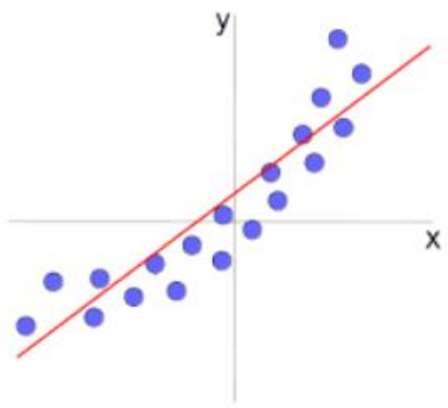


ENTRENAR MODELO



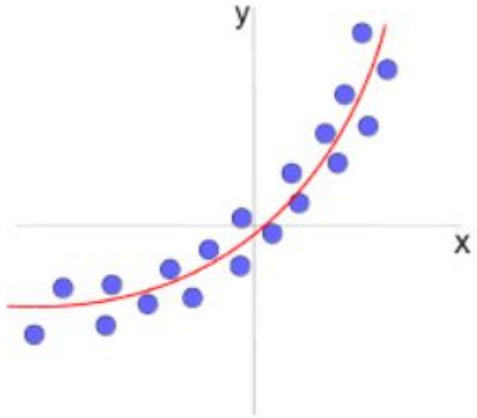
CONSECUENCIAS DEL ENTRENAMIENTO

Underfitting



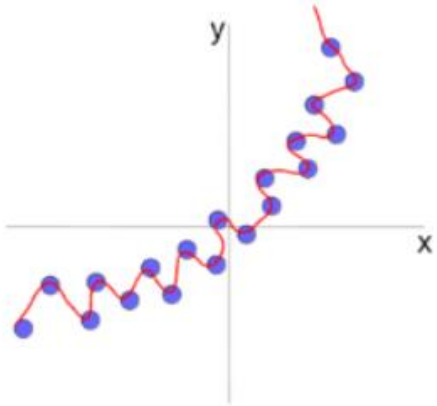
Modelo es demasiado simple para capturar el comportamiento de los datos (*underfitting*).

Good fit



Modelo tiene la complejidad necesaria para capturar los patrones relevantes.

Overfitting



Modelo es muy complejo, y captura hasta el ruido presente en los ejemplos (*overfitting*).

TIPOS DE MODELOS DE APRENDIZAJE

- Existen varios, se debe probar cual es el que se ajusta mejor a tu set de datos, el que obtiene los mejores resultados según los parámetros que defines.
- Ejemplos:
  - Regresión lineal
  - Árboles de decisión y regresión (acepta valores no numéricos)
  - KNN

PASAR VALORES CUALITATIVOS A CUANTITATIVOS

```
from sklearn.preprocessing import LabelEncoder

cat_vars = ['species']
label_encoder = LabelEncoder()
for i in cat_vars:
    iris_dataset[i] = label_encoder.fit_transform(iris_dataset[i])
iris_dataset.dtypes
```

SEPARAR LOS DATOS DE ENTRENAMIENTO Y NORMALIZACIÓN DE ESTOS

- Separar los datos en entrenamiento y testing:

```
training_set, test_set = train_test_split(iris_dataset.copy(), test_size = 0.3)
training_set, validation_set = train_test_split(training_set.copy(), test_size = 0.1)
```

- Normalizar los datos: (excluyendo la columna/s target)

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
features = ['sepal_length','sepal_width', 'petal_length', 'petal_width']

training_set[features] = scaler.fit_transform(training_set[features])
validation_set[features] = scaler.transform(validation_set[features])
test_set[features] = scaler.transform(test_set[features])

training_set.head()
```

ENTRENAR EL MODELO

```
def training_and_eval(model, training, test, training_target, test_target):
    model.fit(training, training_target)
    predictions = model.predict(test)
    accuracy = metrics.accuracy_score(predictions, test_target)
    print(f"Accuracy: {accuracy: .2}")
    test_mse = metrics.mean_squared_error(predictions, test_target)
    print(f"MSE: {test_mse}")
```