Proyecto de Data Science

DBlandIT

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Datos & Datasets

1. **MovieLens**
   http://grouplens.org/datasets/movielens/
2. **AWS Public Datasets**
   http://aws.amazon.com/fr/datasets/
3. **Kaggle**
   https://www.kaggle.com/datasets
4. **Quora.com**
   https://www.quora.com/Where-can-I-find-large-datasets-open-to-the-public

DBlandIT

Datos & Datasets

Algunos dataset locales...

1. **Buenos Aires Data**
   http://data.buenosaires.gob.ar/
2. **La Nacion Data**
   http://data.lanacion.com.ar/
3. **Dat.ar**
   http://datar.noip.me/

DBlandIT

Proyecto de DS

Metodología y procedimientos

DBlandIT

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Pasos de un proyecto de DS

1. Ver el panorama general del problema.
2. Obtener datos.
3. Explorar los datos (técnicas visualización).
4. Preparación & limpieza de datos (Feature Engineering)
5. Seleccionar un modelo/algoritmo.
6. Entrenar y evaluar el modelo/algoritmo.
7. Optimizar el modelo/algoritmo (y evaluar).
8. Despliegue!
9. Monitoreo.
10. Mantenimiento.

DBlandIT

Pregunta: Definir el *qué*?

1. Definir **pregunta qué se quiere responder** con el nuevo modelo.

DBlandIT

Objetivo: El *por qué*?

2. Entender el **objetivo** de proyecto y **negocio**.
   - Por qué lo están haciendo?
   - Cómo y para qué lo va a usar el cliente?
   - Qué beneficio esperan obtener a partir de los resultados?
   - Es un proyecto de *único uso* vs. *integrar a su ciclo productivo*?

DBlandIT

Contexto

3. Entender el contexto de proyecto.
   - Como es la solución actual, si la hay?
   - Han intentado atacar la problemática previamente? Cuál fue el resultado?
   - Qué datos se utilizarán/están disponibles? De donde provienen? Quienes son los *Owners*?
   - Cómo será la interacción con otros sistemas? Cómo se integrará a la arquitectura actual?

DBlandIT

Nos ayudará a..

Comprender y acotar:

1. Cómo encarar el problema.
2. Tipo de problemática
3. Velocidad y disponibilidad de los resultados.
4. Por ende, algoritmos utilizar.
5. Calidad de datos, formato y volumen de los datos que vamos a recibir.
6. Medidas de performance para evaluar el modelo.
7. Confianza en los resultados y precisión del modelo.
8. Riesgos o potenciales problemas.

DBlandIT

DBlandIT

Quite possibly the most important part in the machine learning process is under standing the data you are working with and how it relates to the task you want to solve. It will not be effective to randomly choose an algorithm and throw your data at it. It is necessary to understand what is going on in your dataset before you begin building a model. Each algorithm is different in terms of what kind of data and what problem setting it works best for. While you are building a machine learning solution, you should answer, or at least keep in mind, the following questions:

What question(s) am I trying to answer? Do I think the data collected can answer that question?

· What is the best way to phrase my question(s) as a machine learning problem?

· Have I collected enough data to represent the problem I want to solve?

· What features of the data did I extract, and will these enable the right predictions?

· How will I measure success in my application?

· How will the machine learning solution interact with other parts of my research or business product?

In a larger context, the algorithms and methods in machine learning are only one part of a greater process to solve a particular problem, and it is good to keep the big picture in mind at all times. Many people spend a lot of time building complex machine learning solutions, only to find out they don't solve the right problem.

When going deep into the technical aspects of machine learning (as we will in this book), it is easy to lose sight of the ultimate goals. While we will not discuss the questions listed here in detail, we still encourage you to keep in mind all the assumptions that you might be making, explicitly or implicitly, when you start building machine learning models.

Medida de performance

Investigamos sobre T, veamos cómo medir...

Training set
(Experiencia)

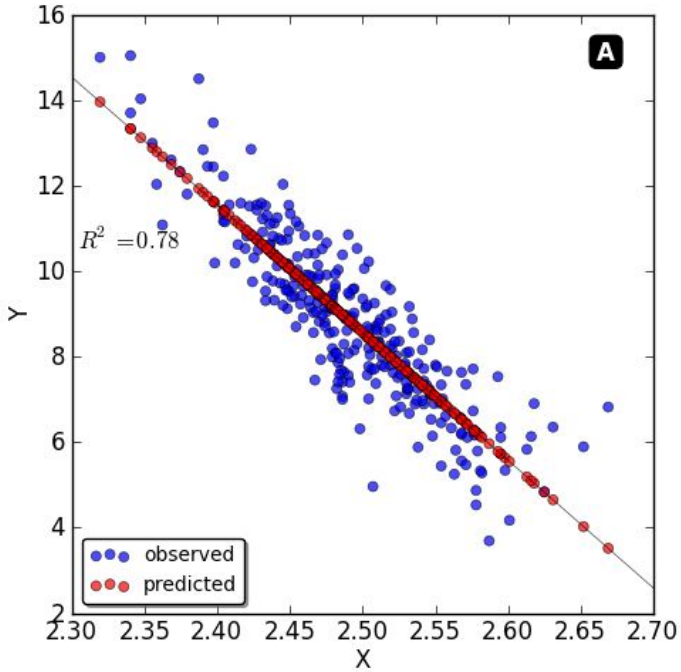
- Instance / sample 1
- Instance / sample 2
...
- Instance / sample *n*

Modelo
(Programa)

Measure
(Desempeño)

DBlandIT

Medida de performance



Determinar cómo vamos a **medir qué tan buenos son los resultados** de nuestro modelo.

DBlandIT

DBlandIT

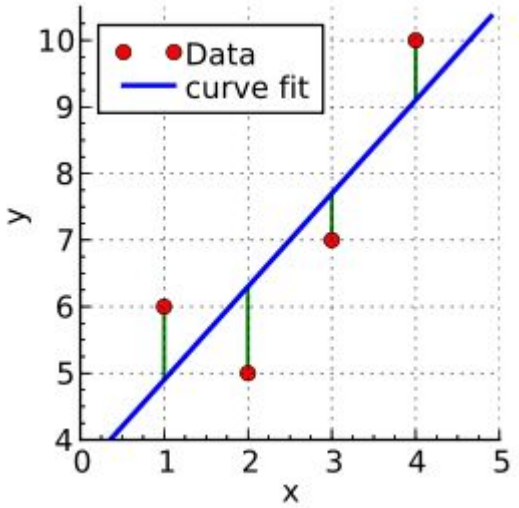
Medida de performance - RMSE

Root Mean Square Error (RMSE)

$$\sqrt{\sum \frac{(y_{pred} - y_{ref})^2}{N}}$$

Mide la desviación estándar del error de las predicciones que hace el sistema.

Muy utilizado en problemas de regresión.



DBlandIT

DBlandIT

Medida de performance - RMSE

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

X(i): Vector con los valores de la medición *i* para cada una de las *features*.

y(i): *Label*, valor real (y predicción deseada) para la medición *i*.

h: La función de predicción (nuestro modelo), tal qué:

$\hat{y}(i) = h(x(i)) \cong y(i)$

m: Cantidad de mediciones.

DBlandIT

Medida de performance - RMSE

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

X: Matrix con todos los valores de las *features* para todas las mediciones

Filas -> medición

Columnas -> *feature*

RMSE(X,h): Función de costo, usando nuestro modelo (función de predicción *h*) y dataset X.

DBlandIT

Medida de performance - MAE

Existen varias medidas de performance. Dependiendo las situaciones algunas son mejores que otras.

Mean Absolute Error (MAE)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

Ambas, RMSE y MAE miden la distancia entre dos vectores, el de las mediciones y el de las predicciones.

Menos susceptible a outliers (valores extremos).

DBlandIT

Medida de performance - MAE

Dataset con errores:

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

Dataset con outliers:

ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

DBlandIT

Obtener los datos

Suelen haber distintos fuentes de datos:

1. Base de datos NOSQL/RDBMS.
2. Archivos: Excels, CSV, txt...
3. Sistemas externos.

Realizar la **importación mediante procesos automáticos** (ejemplo scripts de python).

- Importaciones más rápidas, menos propensas a errores humanos.
- Obtener nuevos datos.
- Datos cambiantes y cron de importación.
- Facilita deploy y/o distribuir el dataset en múltiples servidores/ambientes.

No pasar por alto: obtener credenciales, **documentación**, DER, “maestros implícitos”, codificaciones estándar y **referentes de consulta** para poder entender cómo interpretar dichos valores.

DBlandIT

Obtener los datos

Identifying the zip code from handwritten digits on an envelope:

Here the input is a scan of the handwriting, and the desired output is the actual digits in the zip code. To create a dataset for building a machine learning model, you need to collect many envelopes. Then you can read the zip codes yourself and store the digits as your desired outcomes.

Determining whether a tumor is benign based on a medical image;

Here the input is the image, and the output is whether the tumor is benign. To create a dataset for building a model, you need a database of medical images. You also need an expert opinion, so a doctor needs to look at all of the images and decide which tumors are benign and which are not. It might even be necessary to do additional diagnosis beyond the content of the image to determine whether the tumor in the image is cancerous or not.

Detecting fraudulent activity in credit card transactions:

Here the input is a record of the credit card transaction, and the output is whether it is likely to be fraudulent or not. Assuming that you are the entity distributing the credit cards, collecting a dataset means storing all transactions and recording if a user reports any transaction as fraudulent.

An interesting thing to note about these examples is that although the inputs and outputs look fairly straightforward, the data collection process for these three tasks is vastly different. While reading envelopes is laborious, it is easy and cheap. Obtaining medical imaging and diagnoses, on the other hand, requires not only expensive machinery but also rare and expensive expert knowledge, not to mention the ethical concerns and privacy issues. In the example of detecting credit card fraud, data collection is much simpler. Your customers will provide you with the desired output, as they will report fraud. All you have to do to obtain the input/output pairs of fraudulent and nonfraudulent activity is wait.

DBlandIT

Obtener los datos

.For both supervised and unsupervised learning tasks, it is important to have a representation of your input data that a computer can understand. Often it is helpful to think of your data as a table. Each data point that you want to reason about (each email, each customer, each transaction) is a row, and each property that describes that data point (say, the age of a customer or the amount or location of a transaction) is a column. You might describe users by their age, their gender, when they created an account, and how often they have bought from your online shop. You might describe the image of a tumor by the grayscale values of each pixel, or maybe by using the size, shape, and color of the tumor.

Each entity or row here is known as a sample (or data point) in machine learning, while the columns—the properties that describe these entities—are called features.

Later in this book we will go into more detail on the topic of building a good representation of your data, which is called feature extraction or feature engineering. You should keep in mind, however, that no machine learning algorithm will be able to

DBlandIT

SciKit API

Dataset

Datasets:

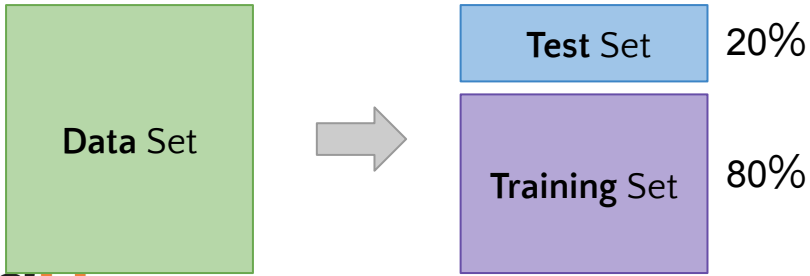
No implementarlo con estructuras o clase propias!

Implementados como **DataFrames de Pandas**.

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0

Test Set

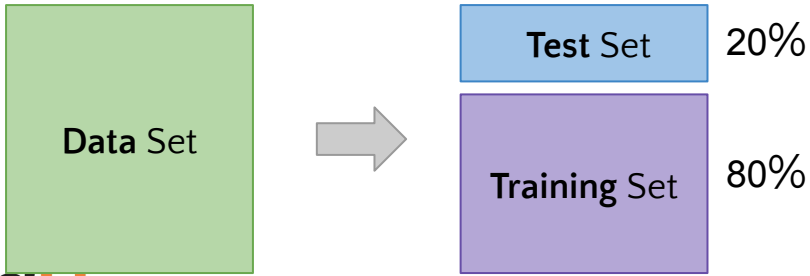
- Separar parte de los datos en el **Test Set**.
  Lo utilizaremos para **evaluar qué tan bien performa el modelo** antes de su puesta en producción.
- Existen **diferente estrategias**:
  - **Random**: Tomar 20% de las muestras/observaciones al azar.



Test Set

- Separar parte de los datos en el **Test Set**.
  Lo utilizaremos para **evaluar qué tan bien performa el modelo** antes de su puesta en producción.
- Existen **diferente estrategias**:
  - **Random**: Tomar 20% de las muestras/observaciones al azar.

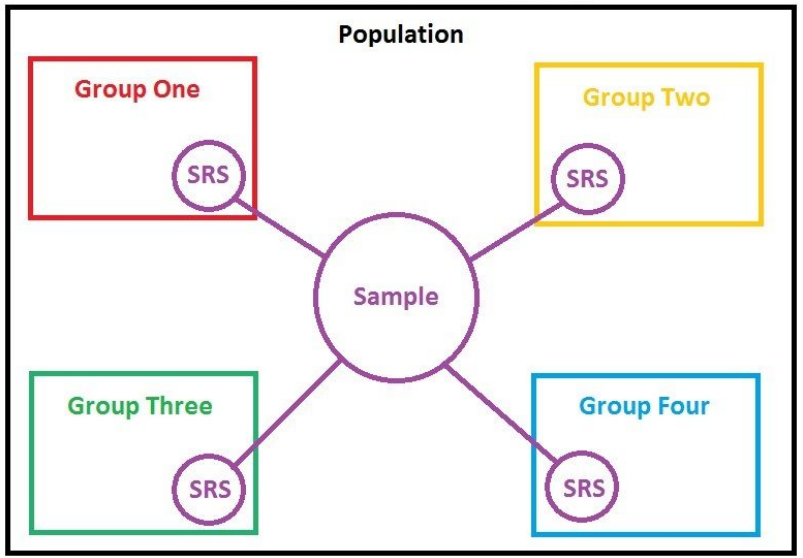
Posibles problemas?



DBlandIT

Test Set

- **Stratified sampling**:
  - Dataset se divide en grupos homogéneos (stratas).
  - Mutuamente excluyentes.
  - De cada Strata, se seleccionan cantidad representativa de observaciones.
    Aplicar Random sampling.



DBlandIT

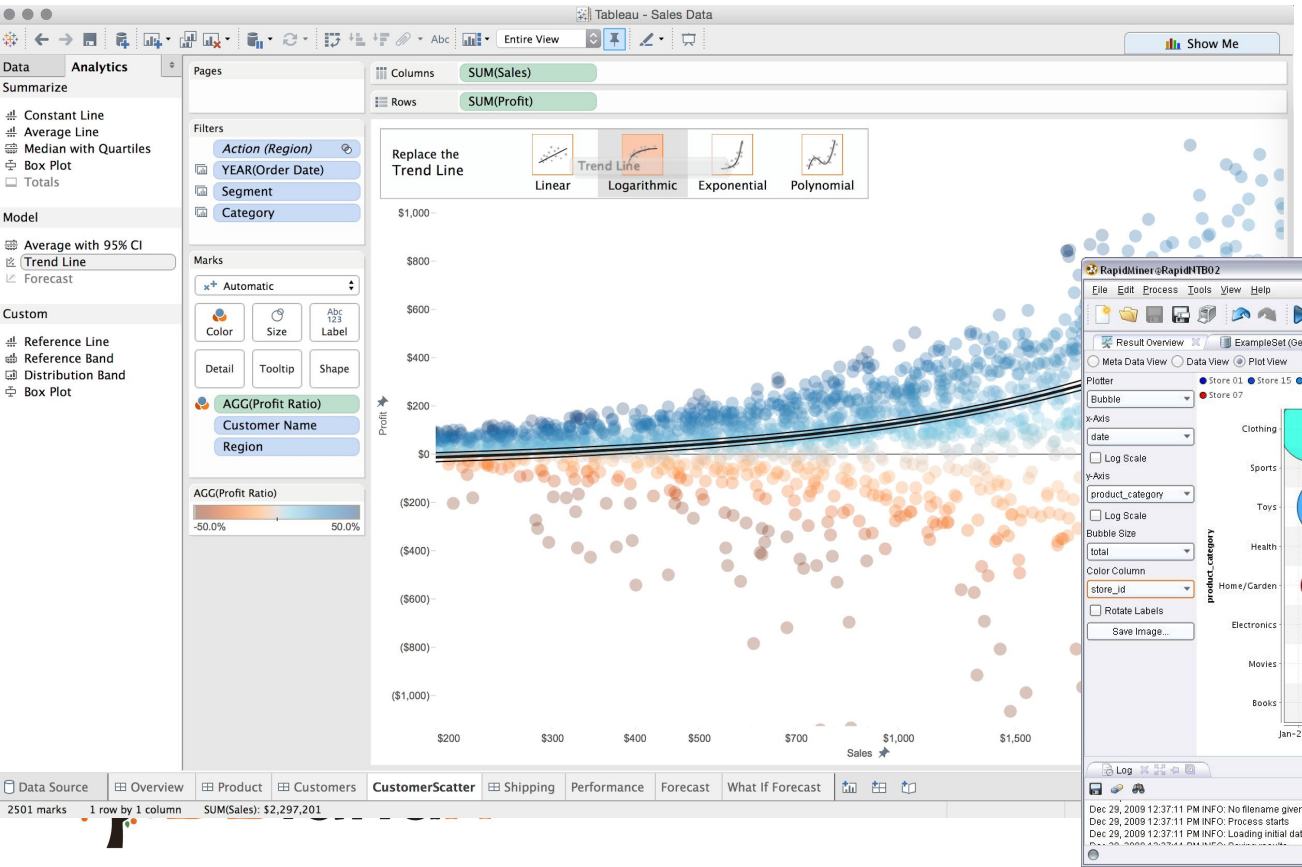
DBlandIT

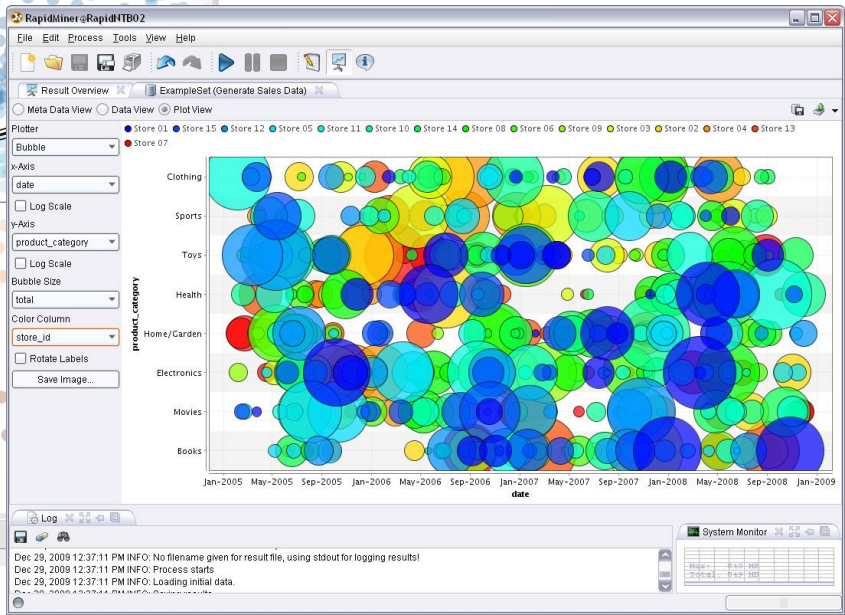
Investigar los datos

- Entender los datos:
  - Qué atributos tienen?
  - Análisis por variables:
    - Categóricos vs. continuos?
    - Media/Max/Min/Moda? Distribución & Frecuencia?
    - Calidad de datos: Completitud & estandarización.
  - Buscar correlaciones
- Sobre el Training Set.
- Herramientas de visualización:

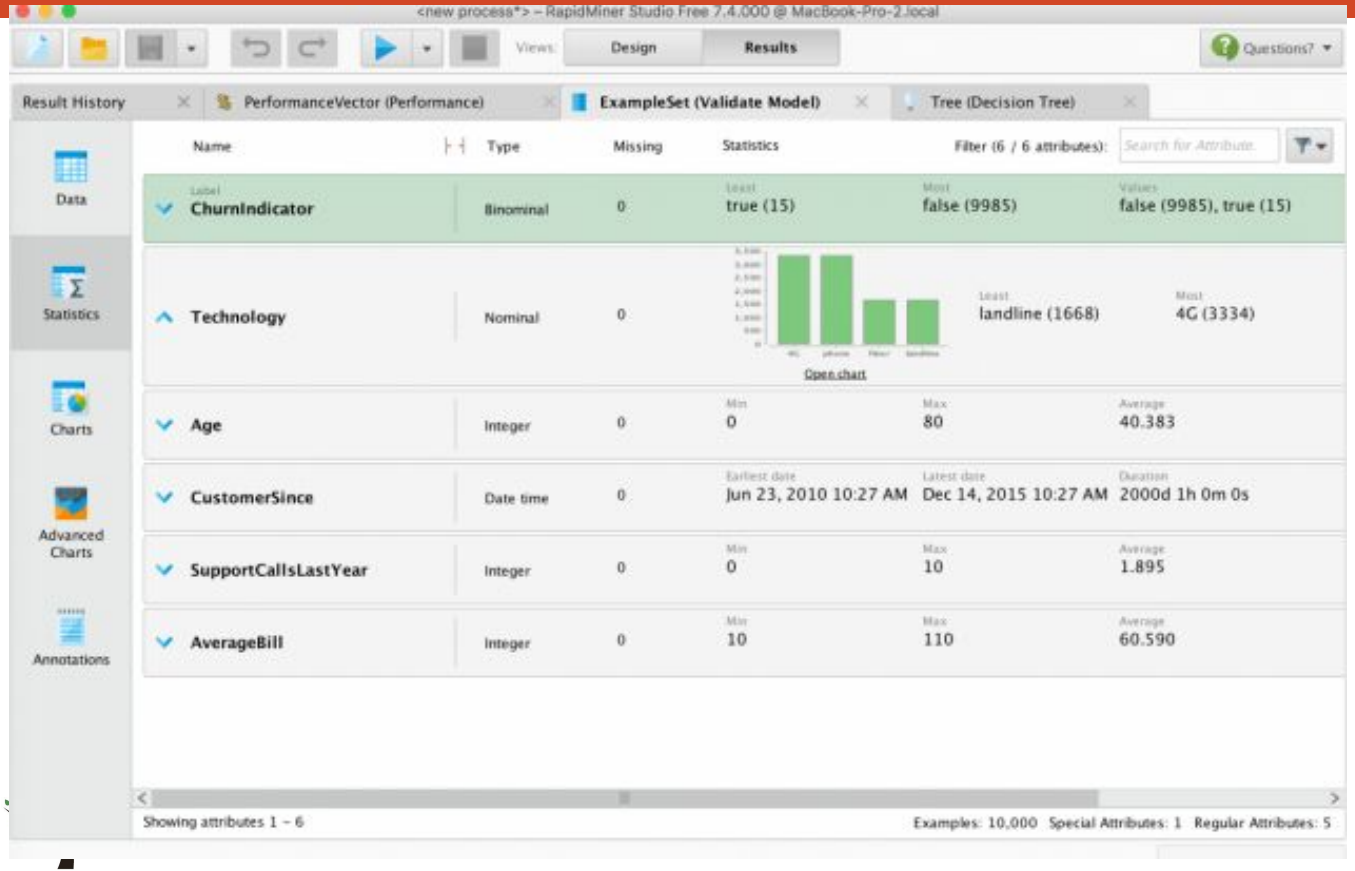
DBlandIT

Herramientas de visualización





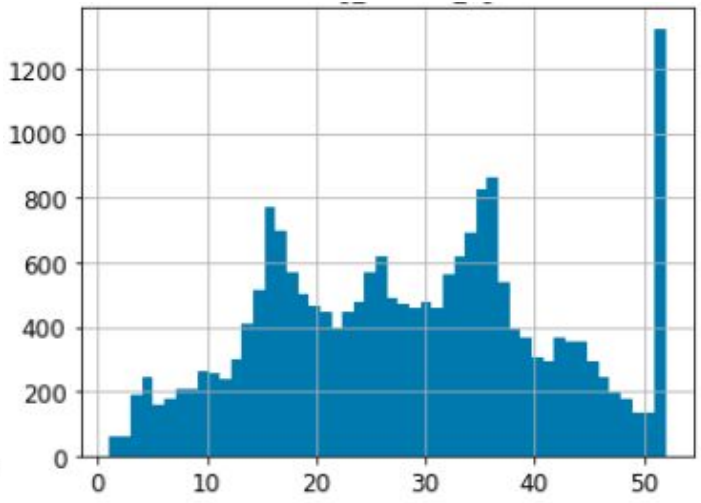
Herramientas de visualización



DBlandIT

Matplotlib

```
%matplotlib inline
# only in a Jupyter notebook
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```



Plot() API

```
DataFrame.plot(x, y, kind…)
```

x : label or position, default None
y : label or position, default None
kind : str
- ‘line’ : line plot (default)
- ‘bar’ : vertical bar plot
- ‘barh’ : horizontal bar plot
- ‘hist’ : histogram
- ‘scatter’ : scatter plot

figsize : a tuple (width, height) in inches

colormap : str or matplotlib colormap object, default None

legend : False/True/’reverse’

Place legend on axis subplots

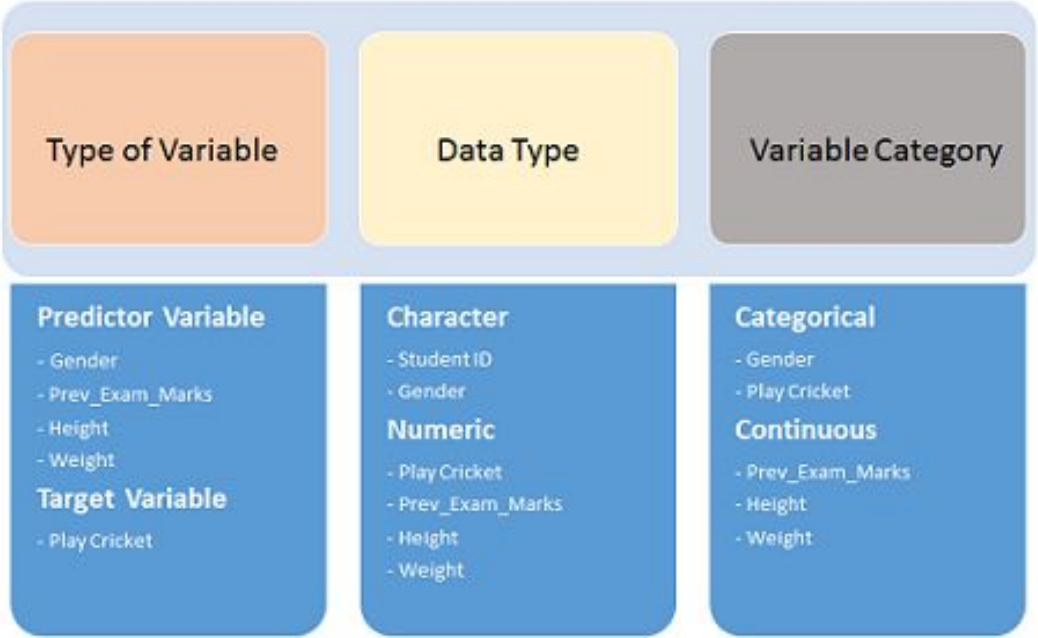
loglog : boolean, default False

Use log scaling on both x and y axes

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html

DBlandIT

Investigar los datos



DBlandIT

Tipos de atributos/Features

- **Features Categóricas/Discretas:**
  Cualitativos, Discretos, Operaciones limitadas.
  - **Nominales y Binarios:** Sin orden. Igualdad y desigualdad.
  - **Ordinales:** Orden definido. Igualdad, desigualdad, mayor y menor.
- **Features Numéricas:**
  Cuantitativos, Continuos o Discretos, Números y tratados como tales.
  - **Intervalo:** No existe cero verdadero. Operaciones anteriores, sumas, restas.
  - **Radio:** Intervalo + Cero verdadero, qué representa ausencia. Operaciones anteriores, multiplicación, división.

Existen atributos que pueden parecer numéricos pero son categóricos.

DBlandIT

Ejemplos de tipos de atributos

- Cantidad de televisores en la casa: Numérico (radio), discreto.
- Tamaño de un combo (Pequeñas, Medianas, Grandes): Categórico (ordinal), discreto.
- Número de llamadas realizadas durante un mes: Numérico (radio), discreto.
- Duración de la llamada más larga: Numérico (radio), continuo.
- Precio de un libro: Numérico (radio), discreto (trabajemos con dos decimales).
- Código postal: Categórico (nominal), discreto.
- Temperatura en grados Centígrados: Numérico (intervalo), continuo.
- Temperatura en grados Farenheit: Numérico (intervalo), continuo.
- Temperatura en grados Kelvin: Numérico (radio), continuo.

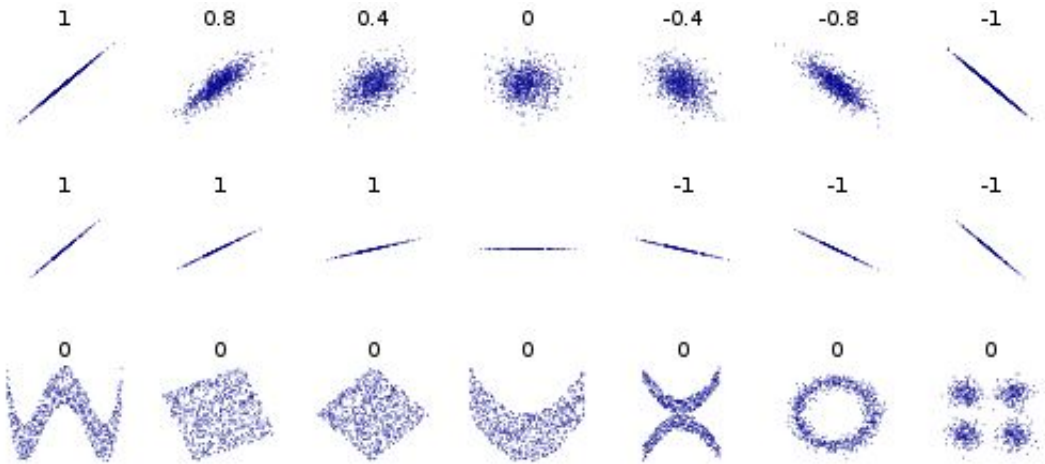
DBlandIT

Operaciones por tipos de atributos

Operación	Nominal	Ordinal	Interval	Ratio
Ejemplos	*Sexo ('F' o 'M')*	*Calificación (E, MB, B, R)*	*Temp. en Celsius (23 °C…)*	*Sueldo ($ 0 … $ 10000 ... $ 50000)*
Distribution & Frecuencia	Si	Si	Si	Si
Mediana, percentiles	No	Si	Si	Si
Suma, resta	No	No	Si	Si
Media/promedio, Desviación estándar	No	No	Si	Si
Ratios, Porcentajes de variación	No	No	Si	Si

Correlaciones

- Dependencia entre variables.
- *Pearson's correlation coefficient:*
  - Rango: [–1,1]
  - No informa sobre correlaciones no lineales.
- No correlación lineal no quiere decir que las variables sean independientes.



DBlandIT

DBlandIT

Regresión Lineal

```
from sklearn.linear_model import
LinearRegression

model = LinearRegression([n_jobs=<int>])

model.fit(X_traning_data, y_target)

model.predict(X_samples)
```


DBlandIT

Proyecto de DS:

Un Ejemplo...

DBlandIT

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

El proyecto...

Supongamos...

DBlandIT

Objetivo: El *por qué*?

1. Definir pregunta qué se quiere responder con el nuevo modelo.
2. Entender el **objetivo** de proyecto y **negocio**.
   - Por qué lo están haciendo?
   - Cómo y para qué lo va a usar el cliente?
   - Qué beneficio esperan obtener a partir de los resultados?
   - Es un proyecto de *único uso* vs. *integrar a su ciclo productivo*?

DBlandIT

Objetivo: El *por qué*?

3. Entender el contexto de proyecto.
   - Como es la solución actual, si la hay?
   - Han intentado atacar la problemática previamente? Cuál fue el resultado?
   - Qué datos se utilizarán/están disponibles? De donde provienen? Quienes son los *Owners*?
   - Cómo será la interacción con otros sistemas? Cómo se integrará a la arquitectura actual?

DBlandIT


DBlandIT


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Limpieza & Preparación de Datos

DBlandIT

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

SciKit API

Dataset

Datasets:

No implementarlo con estructuras o clase propias!

Implementados como **matrices Pandas**.

Hyperparameters

Hyperparameters

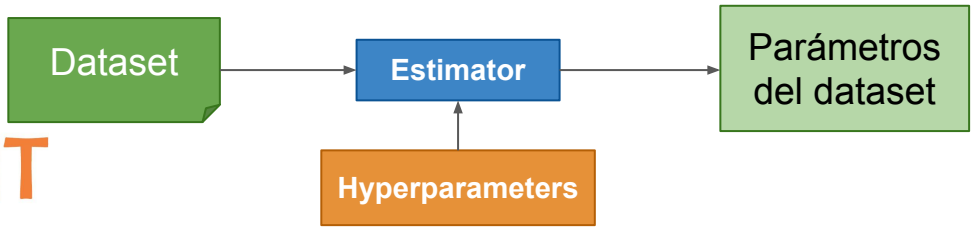
Implementados como **strings** o **números**.

Valores

Estimators

Estimator

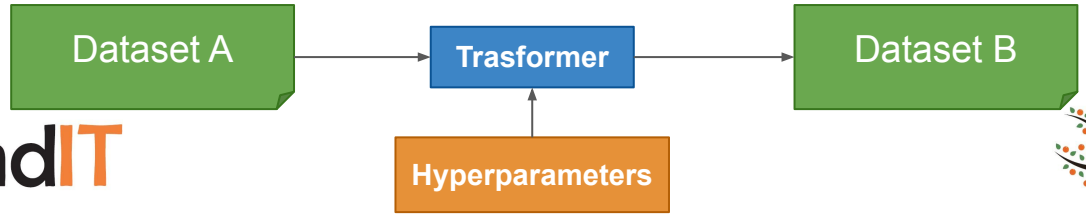
- Objetos que pueden **estimar parámetros a partir de un dataset**.
- Proceso de estimación se implementa mediante el método **.fit**(*dataset*, *[dataSetWithLabels]*)
- El proceso de estimación puede **configurarse mediante Hyperparameters**. Variable de instancia definida en el constructor del estimador.



Valores

Transformers

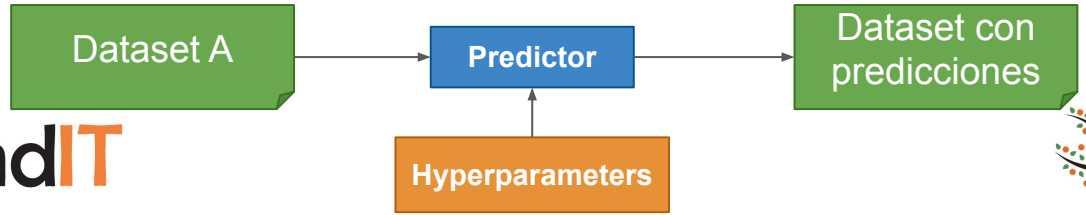
- *Estimators* que pueden **transformar un dataset**.
- Proceso de transformación se implementa mediante el método **.transform**(*dataset*)
- Devuelve un nuevo dataset transformado.
- Conveniencia .fit_transform() = .fit() + .transform()
- *Ejemplo: Imputer*



Valores

Predictors

- *Estimators* que pueden realizar **Predicciones**.
- Proceso de predicción se realiza mediante el método **.predict**(*mediciones*)
- Devuelve un dataset con las predicciones correspondientes
- Método **.score()** mide la calidad de la predicción para un dataset de prueba.
- *Ejemplo: LinearRegresion*



Valores Faltantes

- Interfiere en los comportamientos y relaciones a aprender por el algoritmo.
- Muchos algoritmos no pueden trabajar con features (valores) faltantes.

Name	Weight	Gender	Play Cricket/ Not
Mr. Amit	58	M	Y
Mr. Anil	61	M	Y
Miss Swati	58	F	N
Miss Richa	55		Y
Mr. Steve	55	M	N
Miss Reena	64	F	Y
Miss Rashmi	57		Y
Mr. Kunal	57	M	N

Gender	#Students	#Play Cricket	%Play Cricket
F	2	1	50%
M	4	2	50%
Missing	2	2	100%

Name	Weight	Gender	Play Cricket/ Not
Mr. Amit	58	M	Y
Mr. Anil	61	M	Y
Miss Swati	58	F	N
Miss Richa	55	F	Y
Mr. Steve	55	M	N
Miss Reena	64	F	Y
Miss Rashmi	57	F	Y
Mr. Kunal	57	M	N

Gender	#Students	#Play Cricket	%Play Cricket
F	4	3	75%
M	4	2	50%

Valores Faltantes

Estrategias:

Name	Weight	Gender	Play Cricket/ Not
Mr. Amit	58	M	Y
Mr. Anil	61	M	Y
Miss Swati	58	F	N
Miss Richa	55		Y
Mr. Steve	55	M	N
Miss Reena	64	F	Y
Miss Rashmi	57		Y
Mr. Kunal	57	M	N

- **Descartar mediciones**
  Remover del dataset mediciones completas a las que le falte una feature.
- **Descartar atributo**
  Remover el atributo completo de todas la mediciones, removiendolo del análisis.
- **Definir un valor**
  Completar los feature faltantes con un valor por default: cero, promedio, mediana, moda, etc.
- **Predecir un valor**
  Completar mediante el uso de un modelo entrenado con el subconjunto de datos completos.

Valores Faltantes en scikit

1. Remover mediciones:
   `unDataFrame.dropna(subset=["atributo"])`

2. Remover un attributo:
   `unDataFrame.drop(columns=["atributo"])`

3. Definir un valor, por ejemplo la media:
   ```
   mediaDelAtr = unDataFrame["atributo"].median()
   unDataFrame["atributo"].fillna(mediaDelAtr)
   ```

Si se usa la alternativa 3...

- Entrenar sobre training set (y guardar).
- Reemplazar valores faltantes tanto en el:
  - Dataset de entrenamiento.
  - Dataset el de prueba, al momento de evaluar algoritmo.
  - Mediciones con valores faltantes en el sistema productivo.

DBlandIT

Valores Faltantes en scikit

Imputer

- **Estimador** que ayuda a **lidiar con valores faltantes** en el data set.
- Sus **estadísticas generadas sobre el dataset** de almacenan en la **variable *statistics_*** del mismo.
- Uso:

1. Crear instancia, especificando **estrategia con la que se completara** el valor.

```
imputer = Imputer(strategy="median")
```

2. Crear copia del dataset, removiendo atributos categóricos (no numéricos).

```
dataset_temp = dataset.drop(columns=["categorical_att"])
```


DBlandIT

Valores Faltantes en scikit

3. Alimentar la instancia con el dataset mediante **fit()**.

```
imputer.fit(dataset_temp)
```

Se aplica a todo el dataset, es buena práctica ya que no se sabe qué variables podrían venir vacías en el futuro.

4. Usar **transform()** para completar valores faltantes:

```
X = imputer.transform(dataset_temp)
```

Resultado: Un array de numpy con las features completas.

5. Convertir a DataFrame:

```
dataset_comp = pd.DataFrame(X, columns=dataset_temp.columns)
```


DBlandIT

Atributos Categóricos

- Algoritmos suelen **preferir atributos numéricos**.

Estrategias:

ID	Color
1	Azul
2	Negro
3	Azul
4	Blanco
5	Blanco

Label Encoding:

ID	Color
1	0
2	1
3	0
4	2
5	2

One-hot-encoding:

ID	Azul	Negro	Blanco
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	0	1

Atributos Categóricos en scikit

- LabelEncoder: Encoding **texto a números**.
- OneHotEncoder:
  - Convierte **categoría numéricas a one-hot vectors**.
  - Matrices sparse para el one hot.
- LabelBinarizer = **LabelEncoder + OneHotEncoder**
  - Por default devuelve un array NumPy.
  - Se puede obtener una matriz sparce indicando sparse_output=True en su constructor.

DBlandIT

Feature Scaling

- Atributos en **diferentes escalas afectan negativamente** la performance.

Nombre Atributo	Altura (m.)	Peso (kg.)	Sueldo (ARS)
Min.	1,47	45	8.300,0
Max.	1,72	72	105.000,0

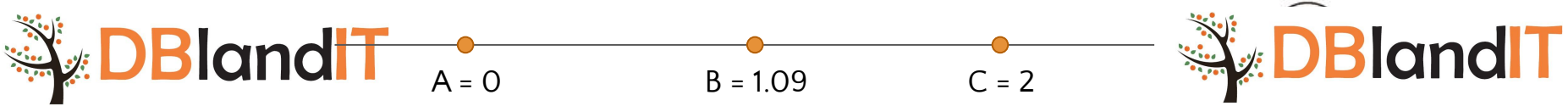
A	B	C
Altura: **1,47**	Altura: **1,65**	Altura: **1,72**
Peso: **45**	Peso: **55**	Peso: **72**
Talle: **small**	Talle: **?**	Talle: **large**
46,47	56,65	73,72

Feature Scaling

- Atributos en **diferentes escalas afectan negativamente** la performance.

Nombre Atributo	Altura (m.)	Peso (kg.)	Sueldo (ARS)
Min.	0	0	0
Max.	1	1	1

A	B	C
Altura: **0**	Altura: **0.72**	Altura: **1**
Peso: **0**	Peso: **0.37**	Peso: **1**
Talle: **small**	Talle: **?**	Talle: **large**
0	1.09	2



Feature Scaling

Estrategias:

- **Normalización/Min-Max:**
  [min, max] → [0, 1]
  Rango Fijo.
  Más susceptible a outliers/valores extremos.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

donde:
x: valor a normalizar
x_{new}: valor normalizado
x_{min}: mínimo del att.
x_{max}: máximo del att.

- **Estandarización:**
  Rango variable, problemas en algunos algoritmos.
  Menos susceptible a outliers/valores extremos.

$$x_{new} = \frac{x - \mu}{\sigma}$$

donde:
x: valor a normalizar
x_{new}: valor normalizado
μ: media del att.
σ: varianza del att.

DBlandIT

Preparación de Datos

Preparación manual vs funciones.

- Aplicar las transformaciones reiteradas veces.
- Reutilizar funciones en otros dataSet y/o proyectos.
- Migrar y utilizarlas en Producción.
- Probar varias transformaciones de forma simple.

DBlandIT

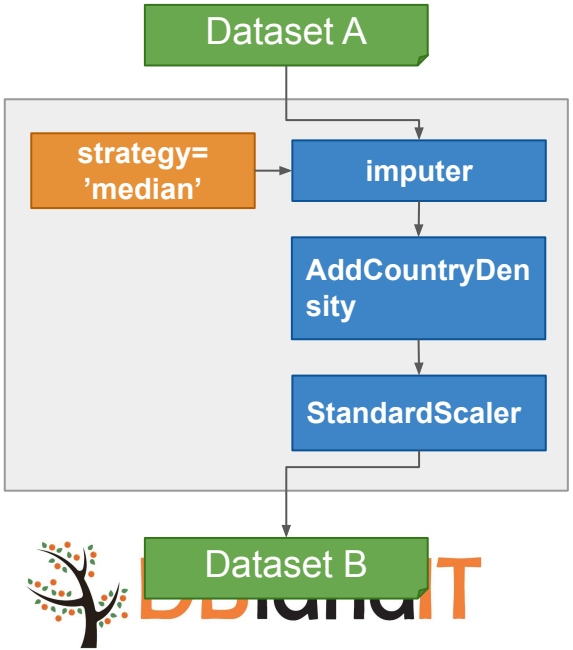
Custom Transformers

- Permiten escribir tareas propias, Ej:
  - Limpieza de datos según lógica de negocio.
  - Combinar atributos específicos.
- Al respetar la API de scikit, se integra bien a otras funcionalidades como pipelines.
- Scikit utiliza duck-typing. Métodos a implementar:
  - `fit()`
  - `transform()`
  - `fit_transform()` ≈ `fit()` + `transform()` [+ optimizaciones]

DBlandIT

Pipelines

Armar secuencia de transformaciones.




```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
        ('imputer', Imputer(strategy="median")),
        ('attribs_adder', AddCountryDensity()),
        ('std_scaler', StandardScaler()),
    ])

datasetB = num_pipeline.fit_transform(datasetA)
```

Pipelines

- Lista de pares: *Nombre – Estimator*
- `Pipeline.fit()` -> `Transformer.fit_transform()` secuencialmente hasta último donde `Estimator.fit()`
- Pipeline expone métodos del último *Estimator* de la secuencia.

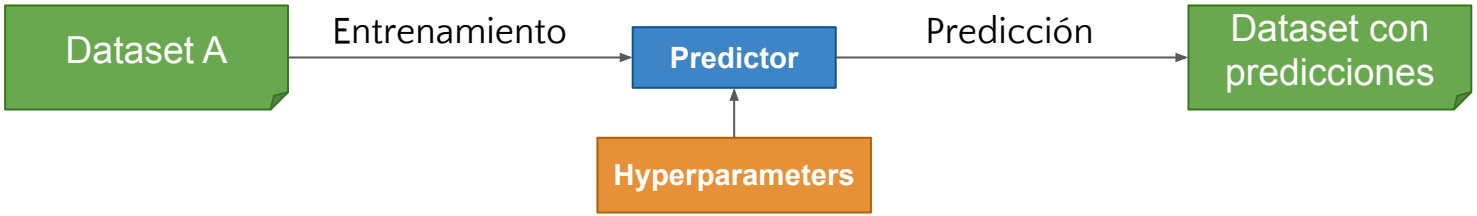
```
num_pipeline = Pipeline([
        ('nameStep1', Transformer),
        ('nameStep2', Transformer),
        ('nameStep3', Estimator),
    ])
```


DBlandIT

Entrenamiento del Modelo

Modelo

Modelo



Hyperparameter

- **Parámetro del algoritmo**, no del modelo. No es afectado por el algoritmo y entrenamiento.
- **Define comportamiento del algoritmo**. Configurado antes del y fijo durante el entrenamiento.
- Su selección da lugar a optimizaciones.

Regresión Lineal

```
from sklearn.linear_model import
LinearRegression

model = LinearRegression([n_jobs=<int>])

model.fit(X_traning_data, y_target)

model.predict(X_samples)
```

`.fit(X_train, y_target)`

X_train: Array numpy con shape:
(`n_samples`, `n_features`)

y_target: Array numpy con shape:
(`n_samples`, `n_targets`)

DBlandIT

Regresión Lineal

```
from sklearn.linear_model import
LinearRegression

model = LinearRegression([n_jobs=<int>])

model.fit(X_traning_data, y_target)

model.predict(X_samples)
```

`.predict(X_samples)`

X_samples: Array numpy con shape:
(n_samples, n_features)

DBlandIT

Mean Square Error (MSE)

```
from sklearn.metrics import
mean_squared_error

y_real = [3, -0.5, 2, 7]
y_pred = [2.5, 0.0, 2, 8]

mse = mean_squared_error(y_real, y_pred)

rmse = np.sqrt(mse)
```

`.mean_squared_error(y_real, y_pred)`

y_real: Array con shape: (`n_samples`)

y_pred: Array con shape: (`n_samples`)

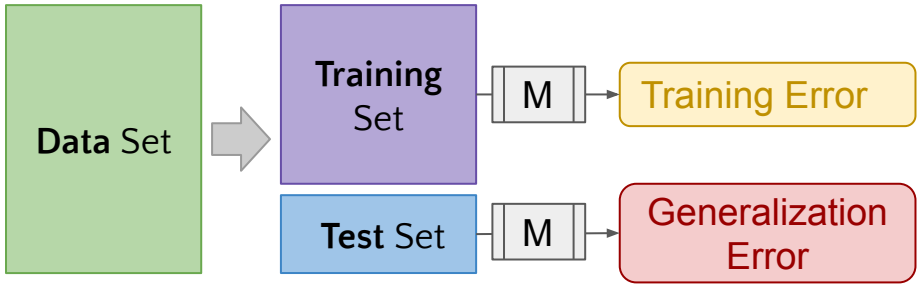
DBlandIT

Testing and Validation

Comprobar predicciones del modelo:

Opción A: Puesta de Producción.

Opción B: Utilizando **Test Set**.



Training Error Bajo, Generalization Error Alto -> Overfitting

¿Como corregir y ajustar el modelo?
¿Cómo comparar performance entre distintos modelos?

DBlandIT

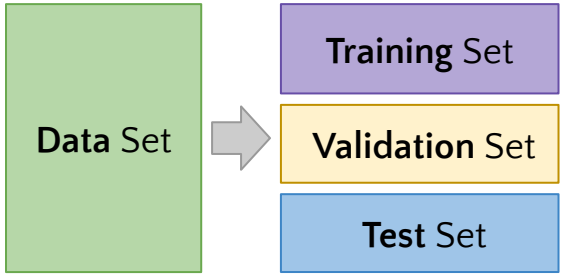
DBlandIT

Testing and Validation

Opción C: Utilizando **Validation Set**.

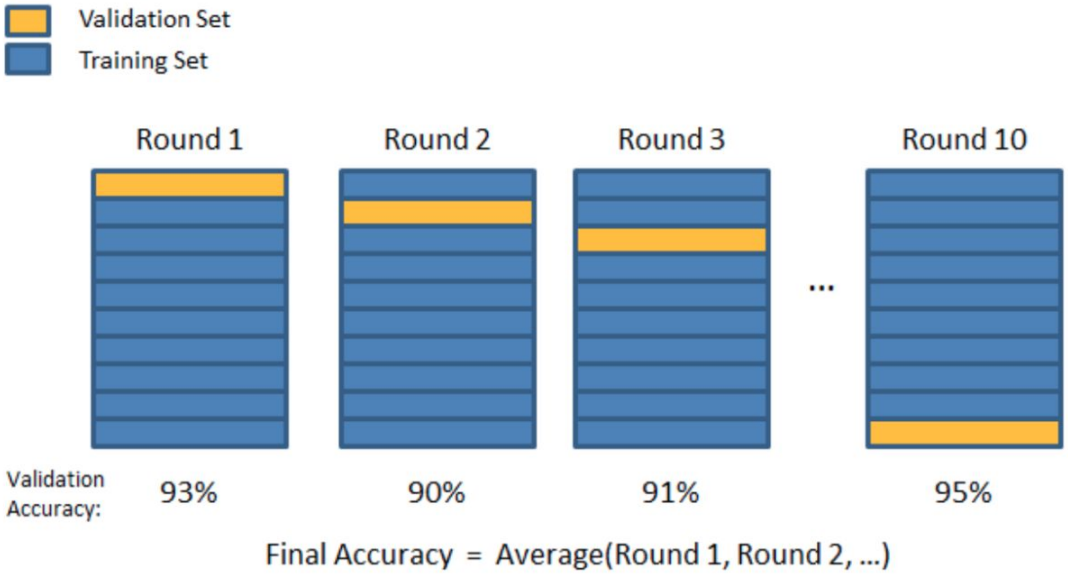
Entrenar modelo mediante el **training set**.
Optimizar y seleccionar modelo con mejor performance para el **validation set**.

Test set 20%, Validation set 20%, Training set 60%.



DBlandIT

Cross Validation



1. División del training set en **k-folds.**
2. Cada modelo:
   - Entrenado con test folds.
   - Validado con fold restante.
   - K iteraciones.
3. Modelo final y optimización, entrenado con training set completo.
4. Error de generalización medido mediante el test set.

DBlandIT

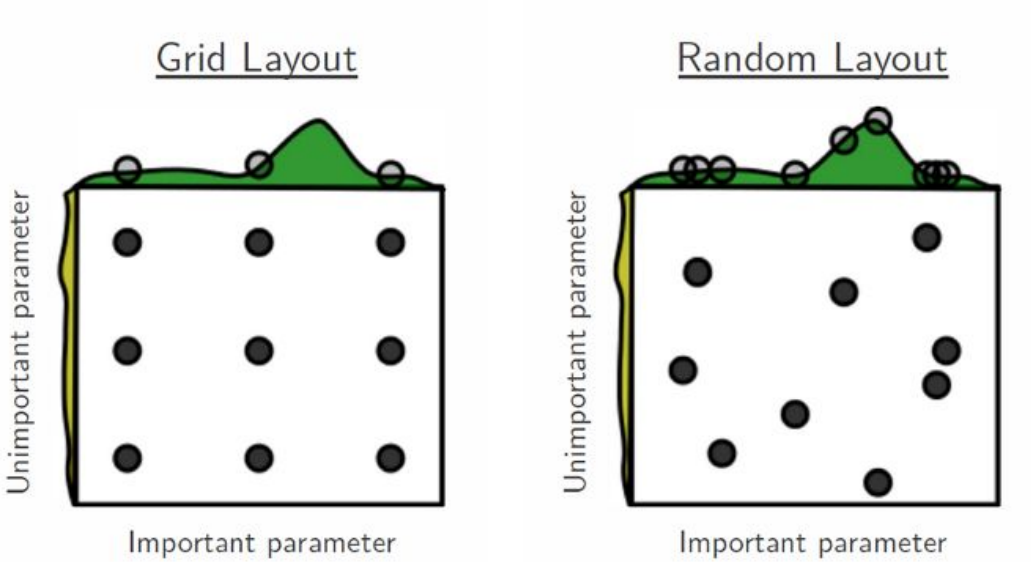
DBlandIT

Optimización de hyperparámetros

Selección de hyperpárametro:

1. **Grid Search**
   Prueba combinaciones a partir de lista de valores posibles.

2. **Randomized Search**
   Prueba n combinaciones de valores al azar.



DBlandIT

DBlandIT


DBlandIT


UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES