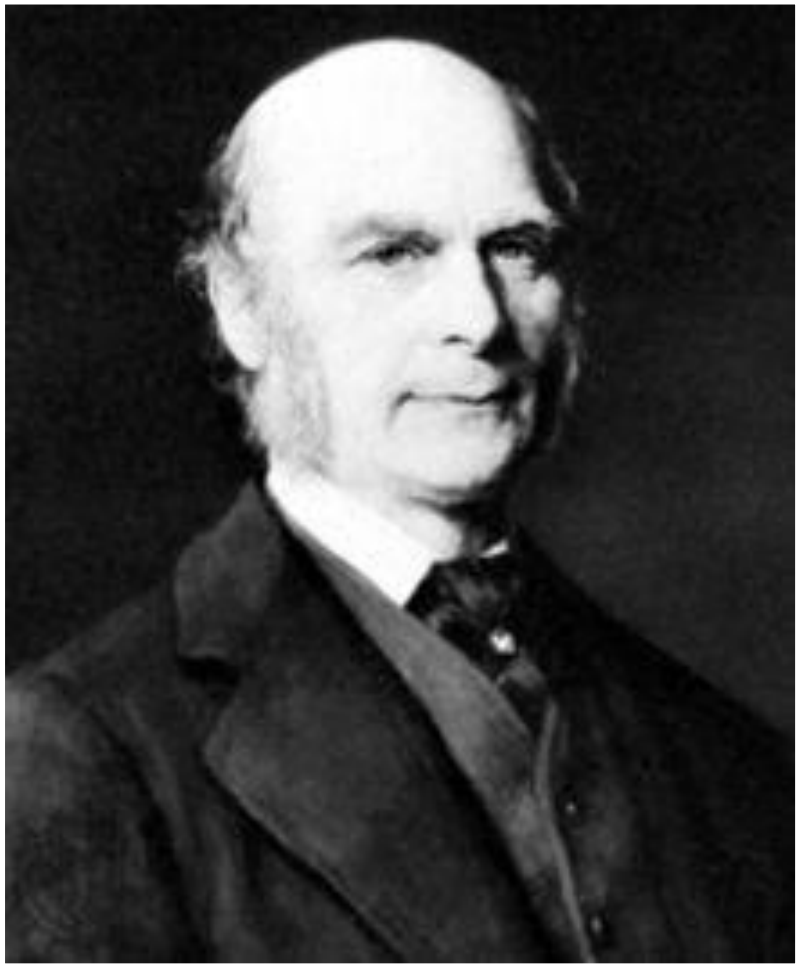
KIBERNUM
IT Academy

Módulo 5 – Aprendizaje de Máquina Supervisado

Regresiones Lineales

Especialización en Ciencia de Datos

Francis Galton (Inglaterra, 1822 – 1911)



Creó el concepto estadístico de correlación y regresión hacia la media, altamente promovido.

Él, fue el primero en aplicar métodos estadísticos para el estudio de las diferencias humanas y la herencia de la inteligencia, introdujo el uso de cuestionarios y encuestas para recoger datos sobre las comunidades humanas, que necesitaba para trabajos genealógicos y biográficos y para sus estudios antropométricos.

Dentro de otras cosas, estudió la relación que existe entre la altura de padres y sus hijos, llegando a la conclusión que los hijos son levemente más altos que sus padres. No obstante, descubrió que la altura de los hijos tendía a estar más cerca a la altura media de todas las personas. A esto le llamó “Regresión a la Media”.

Obtención de Datos

- Nótese que, en este caso, el archivo viene separado por tabulaciones. En este caso, se debe indicar el parámetro **sep** al momento de realizar la lectura.

```
import pandas as pd
import seaborn as sns

df = pd.read_csv('Galton.txt', sep='\t')
```

Galton.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

Family	Father	Mother	Gender	Height	Kids
1	78.5	67	M	73.2	4
1	78.5	67	F	69.2	4
1	78.5	67	F	69	4
1	78.5	67	F	69	4
2	75.5	66.5	M	73.5	4
2	75.5	66.5	M	72.5	4
2	75.5	66.5	F	65.5	4
2	75.5	66.5	F	65.5	4
3	75	64	M	71	2
3	75	64	F	68	2
4	75	64	M	70.5	5
4	75	64	M	68.5	5
4	75	64	F	67	5
4	75	64	F	64.5	5
4	75	64	F	63	5
5	75	58.5	M	72	6
5	75	58.5	M	69	6
5	75	58.5	M	68	6
5	75	58.5	F	66.5	6
5	75	58.5	F	62.5	6
5	75	58.5	F	62.5	6
6	74	68	F	69.5	1
7	74	68	M	76.5	6
7	74	68	M	74	6
7	74	68	M	73	6
7	74	68	M	73	6
7	74	68	F	70.5	6
7	74	68	F	64	6

Exploración

```
df.head()
```

	Family	Father	Mother	Gender	Height	Kids
0	1	78.5	67.0	M	73.2	4
1	1	78.5	67.0	F	69.2	4
2	1	78.5	67.0	F	69.0	4
3	1	78.5	67.0	F	69.0	4
4	2	75.5	66.5	M	73.5	4

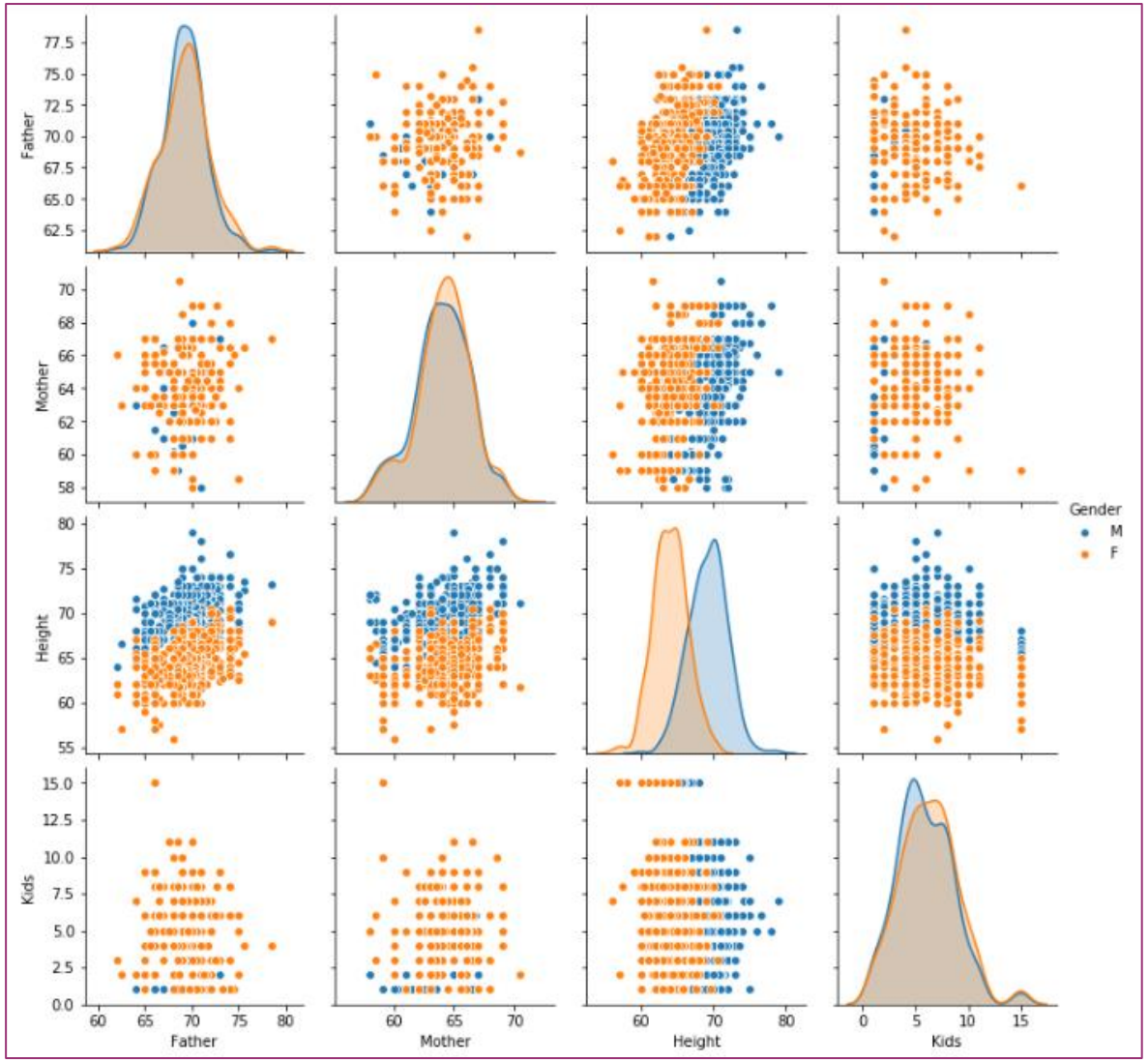
```
df.shape
```

```
(898, 6)
```

```
df.groupby('Gender').count()
```

	Family	Father	Mother	Height	Kids
Gender					
F	433	433	433	433	433
M	465	465	465	465	465

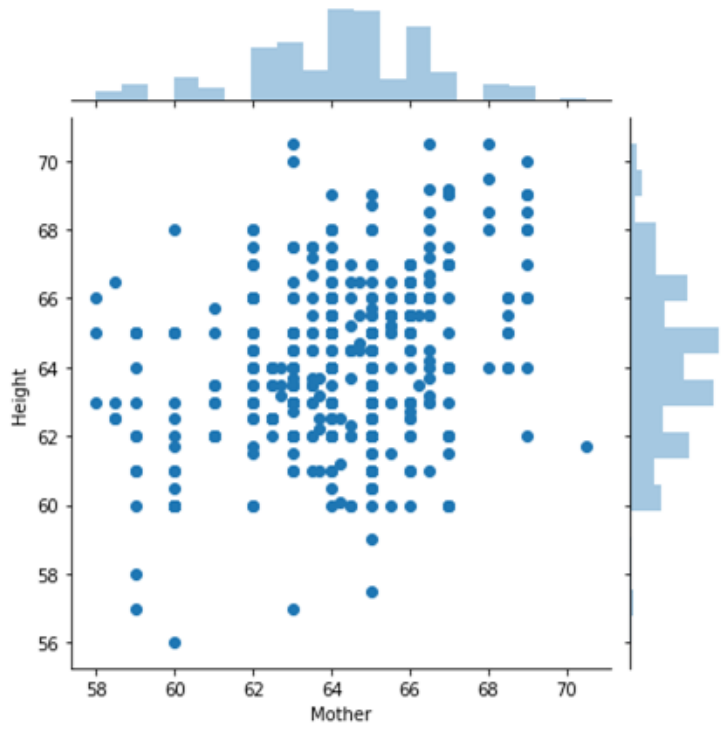
```
sns.pairplot(data=df, hue='Gender')
```



Hipótesis de Trabajo

```
df_f = df[ df['Gender']=='F' ]
```

```
sns.jointplot(data=df_f, x='Mother', y='Height')
```



- En este modelo, vamos a establecer la hipótesis que ***existe una relación lineal entre la estatura de la hija a partir de la estatura de la madre***.
- Por lo tanto, se requiere generar un nuevo dataframe con valores filtrados.
- Nótese que en este caso estamos formulando un modelo regresivo simple, es decir, utilizaremos un solo predictor (feature). Posteriormente, analizaremos si es necesario agregar la estatura del padre al modelo.

Formulación del Modelo

In [8]:

```
1 X = df_f[['Mother']]
2 y = df_f['Height']
```

In [9]:

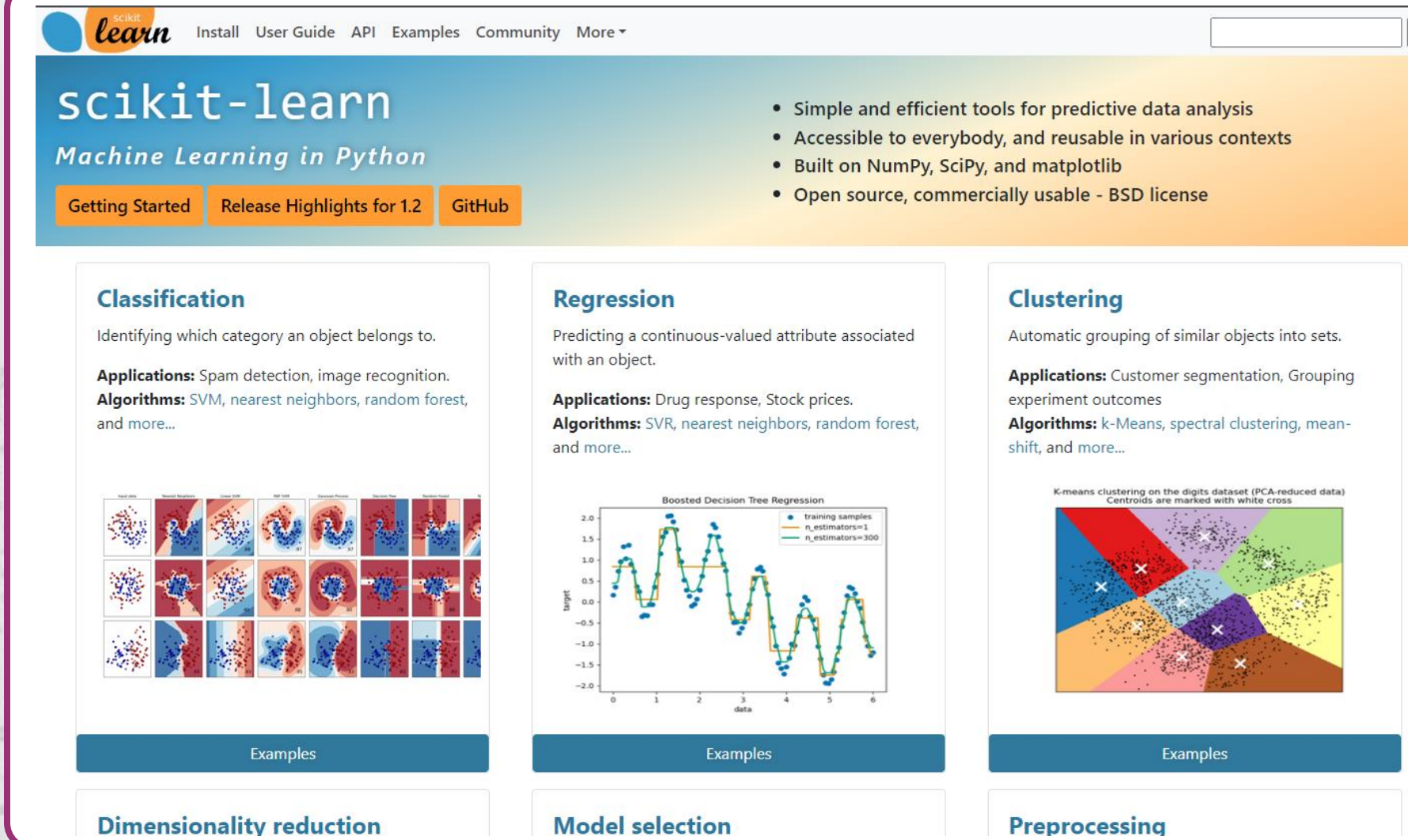
```
1 X.head(2)
```

Out[9]:

	Mother
1	67.0
2	67.0

Definiremos nuestra matriz de features (X) y nuestro vector de resultado (y). Nótese que, en este caso, la matriz X tiene una sola columna.

Librería Scikit-Learn



https://scikit-learn.org/

KIBERNUM

Librería Scikit-Learn

- Scikit-learn es una de las bibliotecas de aprendizaje automático más populares en Python y ofrece una amplia gama de herramientas para el análisis de datos y la construcción de modelos de aprendizaje automático. Algunas de las características principales de Scikit-learn son:

- **Fácil de usar**: Scikit-learn es fácil de usar y proporciona una interfaz consistente, ocultando las complejidades de implementar un algoritmo desde cero.

- **Amplia variedad de algoritmos de aprendizaje automático**: Scikit-learn ofrece una amplia variedad de algoritmos de aprendizaje automático para la clasificación, regresión, clustering, reducción de dimensionalidad, selección de modelos y otros problemas.

Librería Scikit-Learn

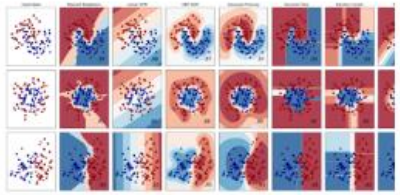
- **Integración con otras bibliotecas**: Scikit-learn se integra bien con otras bibliotecas de análisis de datos en Python, como NumPy, Pandas y Matplotlib.
- **Buena documentación**: Scikit-learn tiene una documentación completa y bien organizada, que incluye ejemplos de código y tutoriales.
- **Escalable**: Scikit-learn es escalable y puede manejar grandes conjuntos de datos.
- **Licencia de código abierto**: Scikit-learn es una biblioteca de código abierto y se puede utilizar de forma gratuita.

Librería Scikit-Learn

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

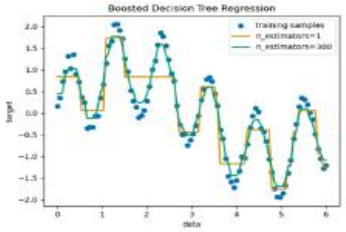


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

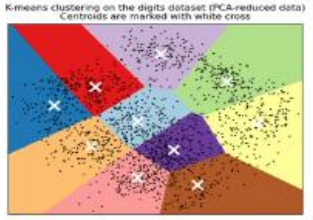


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

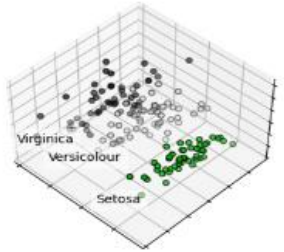


Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...

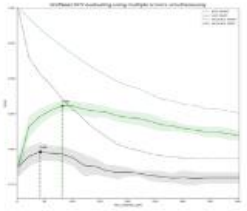


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

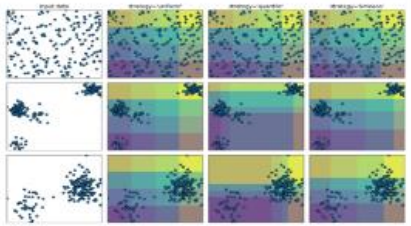


Examples

Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Examples

KIBERNUM

Validación Cruzada

Dividiremos el set de datos, en set de entrenamiento y en set de test, en una proporción de 80% para entrenamiento y 20% para test. Para esto, utilizaremos la librería scikit-learn.

```
[ ]
1 from sklearn.model_selection import train_test_split
```

```
[ ]
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ]
1 print(X_train.shape)
2 print(X_test.shape)
```

```
(346, 1)
(87, 1)
```

Entrenamiento del modelo

A continuación, realizaremos el entrenamiento del modelo lineal importando la clase LinearRegression desde scikit-learn.

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
lm.fit(X_train,y_train)
```

```
LinearRegression()
```

Nótese que el modelo entrenado quedó almacenado en la variable **lm**

Nótese que el entrenamiento lo realizamos sobre el training set

```
# coeficientes ajustados
lm.coef_
```

```
array([0.32561679])
```

Este es el valor del coeficiente que determinó el algoritmo

```
# intercepto
lm.intercept_
```

```
43.17026674816181
```

Este es el valor del intercepto que determinó el algoritmo

Realizando algunas Predicciones

Con nuestro modelo lineal ya podemos realizar algunas predicciones, ya sea sobre un valor, varios valores, o todo el set de test.

```
# predicciones sobre un valor
p1 = [[62]]
lm.predict(p1)
```

```
array([63.35850788])
```

```
# predicciones sobre varios valores
p2 = [[50],
      [55],
      [60],
      [65],
      [70]]
lm.predict(p2)
```

```
array([59.45110637, 61.07919033, 62.70727429, 64.33535826, 65.96344222])
```

```
# predicciones sobre el set de test
y_pred = lm.predict(X_test)
```

```
y_pred[:5]
```

```
array([64.05480614, 64.70791076, 63.72825384, 64.38135845, 64.70791076])
```

```
y_test[:5]
```

```
881    60.0
137    66.5
357    64.0
54     64.5
765    65.5
Name: Height, dtype: float64
```

Comprobando la Hipótesis de Galton

Nótese que, si calculamos la media de las alturas de hijas de la población, nos arroja un valor de 64.11 pulgadas. Esto comprueba lo que Galton le llamó “regresión hacia la media”, puesto que, si una madre tiene una altura baja, digamos 50 pulgadas, la altura de su hija tenderá hacia la media, es decir, tendrá una altura mayor. Pero si la altura de la madre el de 70 pulgadas, la altura de la hija será mejor, tendiendo hacia la media de la población.

```
# el promedio de altura de las hijas mujeres
df_f['Height'].mean()
```

```
64.11016166281759
```

```
# predicciones sobre varios valores
p2 = [[50],
      [55],
      [60],
      [65],
      [70]]
lm.predict(p2)
```

```
array([59.45110637, 61.07919033, 62.70727429, 64.33535826, 65.96344222])
```

KIBERNUM

Métricas de Evaluación de Regresiones

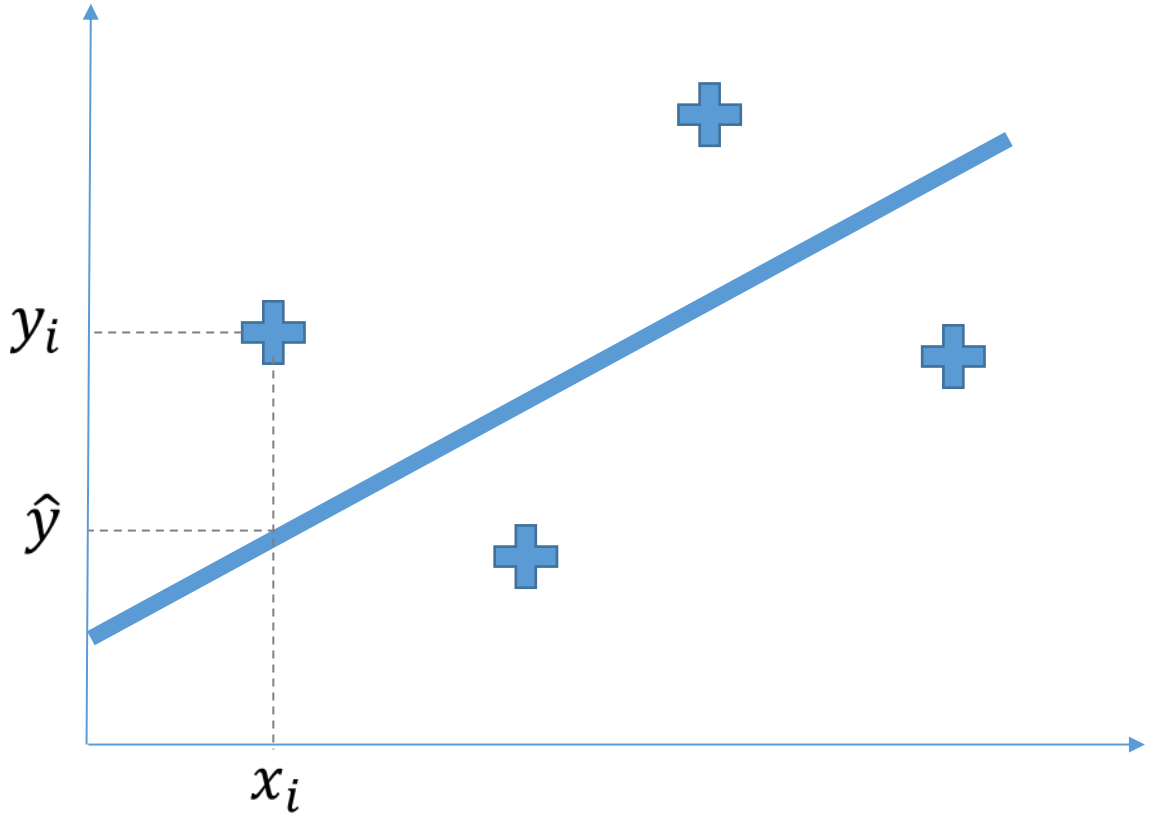
En una regresión lineal, las métricas de evaluación están enfocadas a cuantificar el error del modelo respecto a los valores reales en el set de datos donde realizamos la validación. Existen varias métricas de error, pero las más utilizadas son las siguientes:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

(MAE) Mean Absolute Error

- Es la métrica más sencilla de todas para medir el error de un modelo regresivo, corresponde a la suma de las diferencias absolutas entre el valor predicho y el valor real dividido por la cantidad de mediciones.

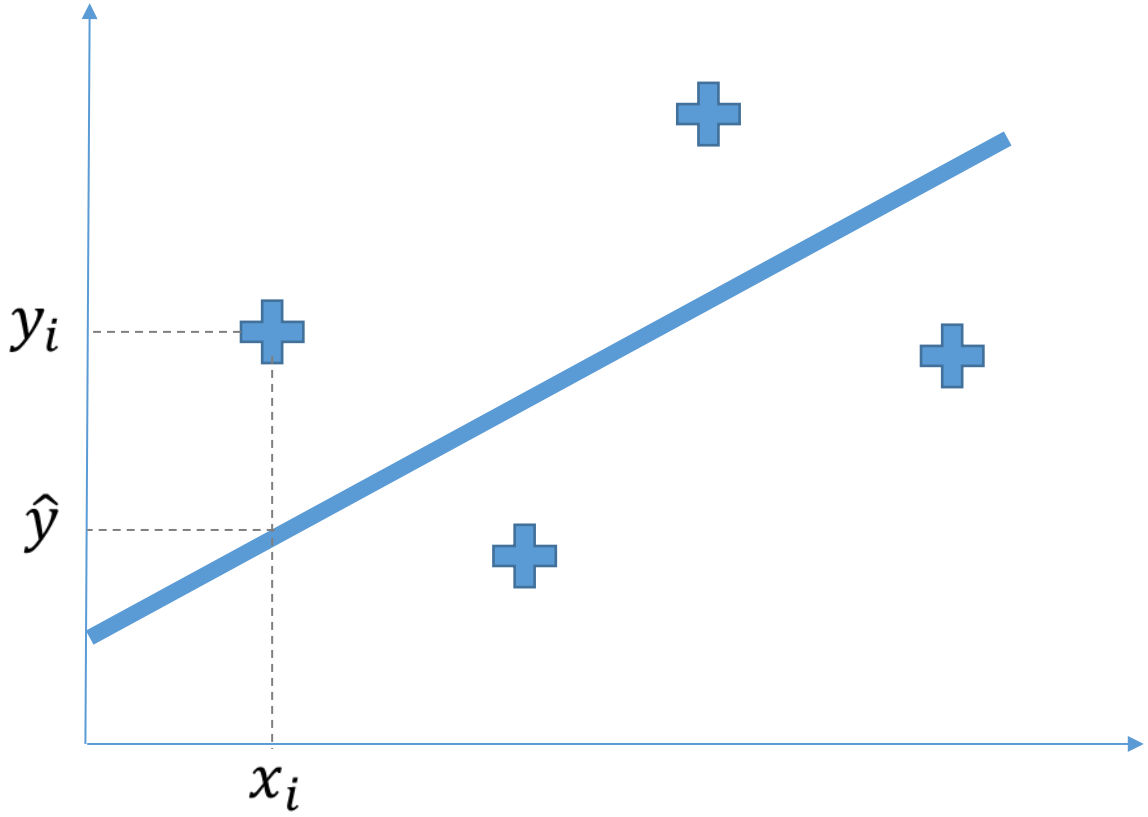
$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$



(MSE) Mean Squared Error

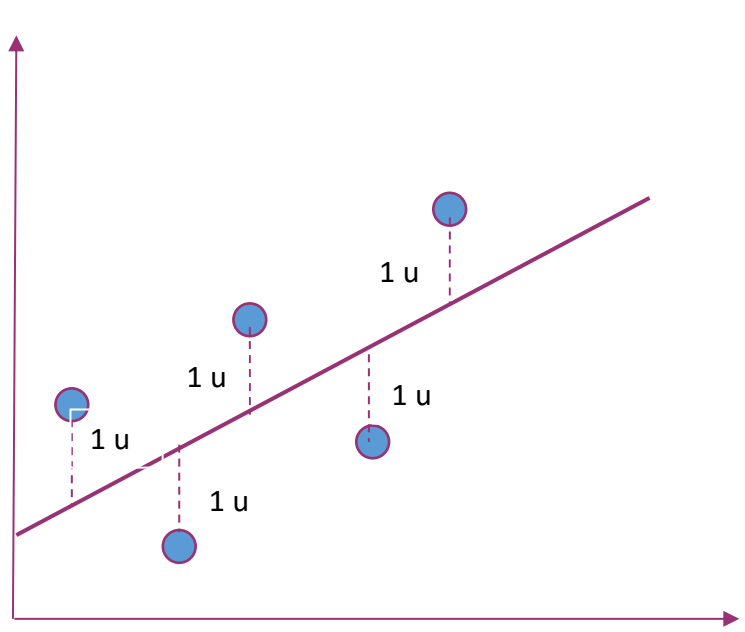
Es similar a la métrica anterior, también sirve para medir el error de un modelo regresivo, sin embargo, corresponde a la suma de las diferencias cuadradas entre el valor predicho y el valor real dividido por la cantidad de mediciones.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

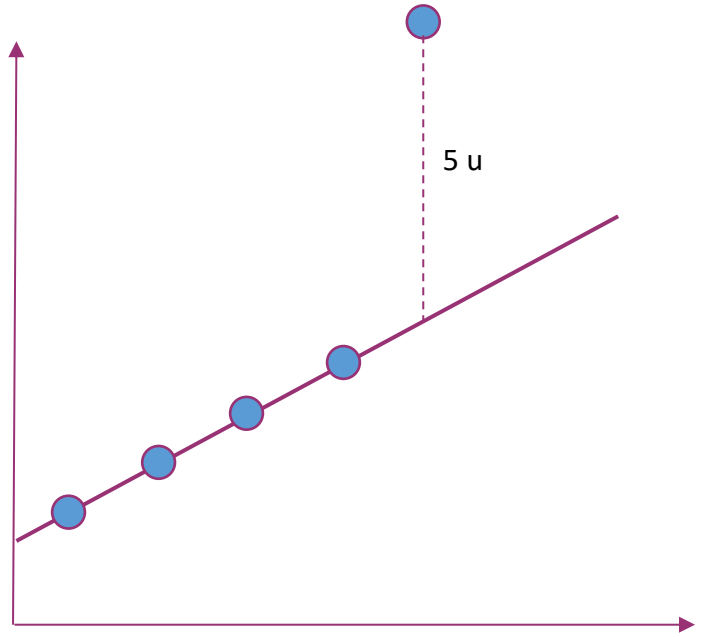


MAE v/s MSE

Para ilustrar la diferencia, veamos los siguientes ejemplos.



MAE = _____ MSE= _____



MAE = _____ MSE= _____

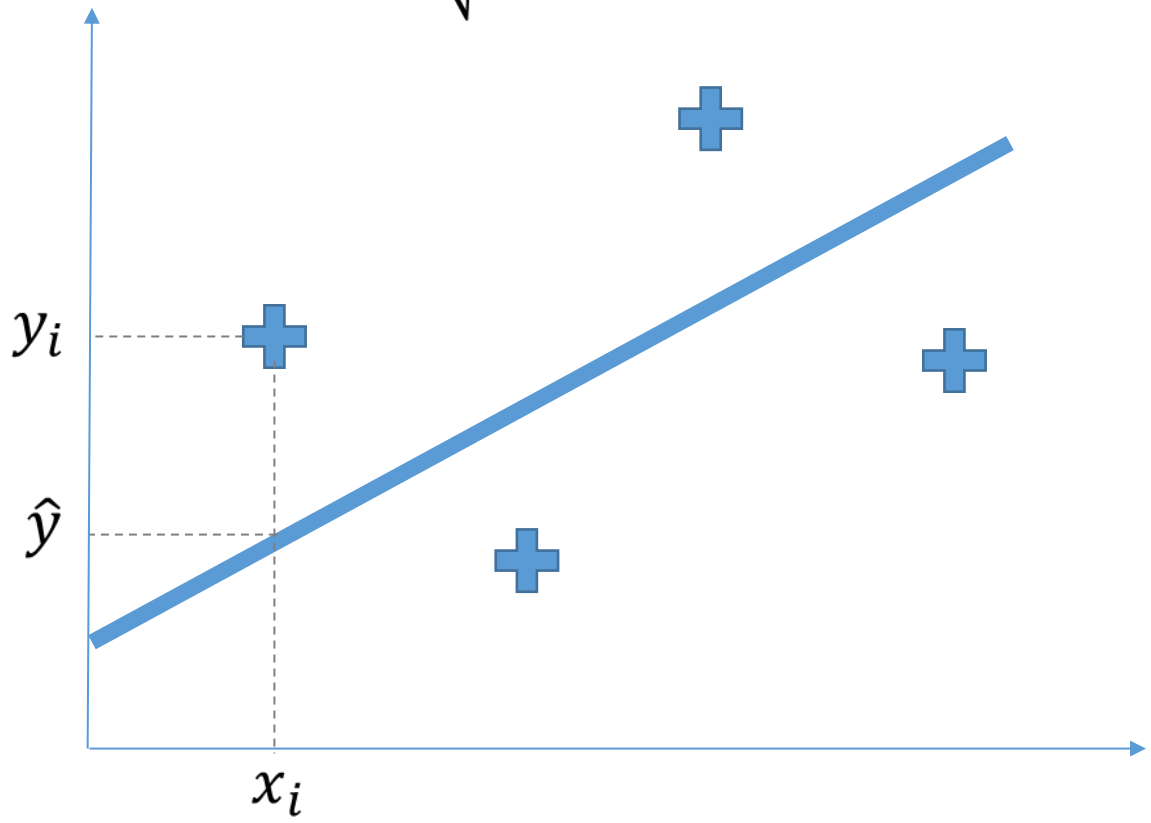
MAE v/s MSE

- La métrica MSE castiga más a los modelos que tienen puntos con alto error, sin embargo, esta métrica está en unidades al cuadrado.

Root Mean Squared Error (RMSE)

- Dado que el MSE tiene una unidad de medida elevada al cuadrado, lo cual hace que la métrica a veces no se comprenda, simplemente extraemos la raíz cuadrada para obtener el RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$



Métricas de Evaluación de Regresiones

Para utilizar las funciones de métricas de evaluación de la regresión, debemos importar la librería **metrics**.

```
[ ]: 1 from sklearn import metrics
```

Calculando las métricas de error

```
[ ]: 1 # Mean Absolute Error
     2 metrics.mean_absolute_error(y_test,y_pred)
```

```
[ ]: 1.962837799904773
```

```
[ ]: 1 # Mean Squared Error
     2 metrics.mean_squared_error(y_test,y_pred)
```

```
[ ]: 6.423322381500185
```

```
[ ]: 1 # Root Medium Squared Error
     2 metrics.mean_squared_error(y_test,y_pred)**.5
```

```
[ ]: 2.53442742675741
```

KIBERNUM

Coeficiente de Determinación (R-cuadrado)

Mide cuán bien ajusta nuestro modelo a los datos, es decir, mide la proporción de varianza de la variable objetivo que el modelo es capaz de explicar. Este coeficiente oscila entre 0 y 1, correspondiendo a 1 el ajuste perfecto del modelo y a 0 cuando la variable de salida no reacciona a nuestros features.

En los modelos de aprendizaje de máquina regresivos, el **score** corresponde justamente al coeficiente de determinación.

```
1  # R-Squared
2  lm.score(X_test,y_test)
```

```
0.06336700925967509
```

En este ejemplo, el modelo sólo es capaz de explicar el 6,3% de la varianza de la variable de salida.

Resultado Hipótesis de Trabajo N°1

El modelo planteado fue el siguiente:

Alt Hija = 43.17 + 0.32 * Alt Madre

Y la evaluación del modelo obtuvo un score de 6% y un RMSE de 2.53 pulgadas. Intentaremos hacerlo mejor en un nuevo modelo.

Hipótesis de Trabajo N°2

Dado que la hipótesis de trabajo inicial solamente logró un score de 6%, es decir, el modelo logra explicar sólo el 6% de la variabilidad de la estatura de la hija, vamos a establecer una nueva hipótesis de trabajo:

En este modelo, vamos a establecer la hipótesis que existe una relación lineal entre la estatura de la hija a partir de la estatura de la madre y del padre.

Es decir, armaremos un modelo con dos features.

$$\text{Alt Hija} = b_0 + b_1 * \text{Alt Madre} + b_2 * \text{Alt Padre}$$

Formulación del modelo

Repetiremos los pasos anteriores y formularemos un nuevo modelo.

```
x = df_f[['Father','Mother']]
y = df_f['Height']
```

```
x.head()
```

	Father	Mother
1	78.5	67.0
2	78.5	67.0
3	78.5	67.0
6	75.5	66.5
7	75.5	66.5

```
y[:5]
```

```
1    69.2
2    69.0
3    69.0
6    65.5
7    65.5
Name: Height, dtype: float64
```

Validación Cruzada

Al igual que anteriormente.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X_train.shape
```

```
(346, 2)
```

```
X_test.shape
```

```
(87, 2)
```

Entrenamiento

Al igual que en el modelo anterior.

```
lm = LinearRegression()
lm.fit(X_train,y_train)
```

```
# coeficientes ajustados del modelo
lm.coef_
```

```
array([0.38719195, 0.30575471])
```

```
# intercepto
lm.intercept_
```

```
17.62385573813038
```

```
# hacemos predicciones con el nuevo modelo
y_pred = lm.predict(X_test)
```

```
metrics.mean_absolute_error(y_test,y_pred)
```

```
1.7194203272019482
```

```
metrics.mean_squared_error(y_test,y_pred)
```

```
4.693701870870291
```

```
metrics.mean_squared_error(y_test,y_pred)**.5
```

```
2.1664952967570206
```

```
# R-Squared
lm.score(X_test,y_test)
```

```
0.31557599636935296
```

Evaluación del Modelo N°2

Obtenemos las métricas de error y score R2.

En este modelo, con dos features, se logra explicar el 31% de la varianza de "y" (alt hija), con un rmse de 2.1. Resultados mejores que el caso anterior.

Dudas y consultas

Fin Presentación

KIBERNUM