**Sistemas de Gestión de Bases de Datos**
$1^{er.}$ cuatrimestre 2019

Guía Práctica 5
**Aprendizaje Automático. Método: Regresión Lineal**

*Ingeniería Informática*
*Universidad Nacional de Avellaneda*

# 1. Introducción

Los objetivos de esta práctica son:

- Adaptar y filtrar el conjunto de datos
- Comprender el modelo de *regresión multilineal*
- Implementar el modelo con la librería *sci-kit learn* de python
- Comprender la técnica de *bootstrap*
- Comprender la técnica de *cross-validation*

La bibliografía que emplearemos es la siguiente:

- Data Science from Scratch, Joel Grus, O'Reilly (capítulos 14 y 15)

# 2. Ejercicios

El objetivo de esta práctica es implementar un tasador de propiedades de la CABA basado en el conjunto de datos utilizado en la práctica 2. Los ejercicios van siguiendo un esquema incremental en el sentido de que cada uno depende de los anteriores.

La idea será considerar un conjunto de características o *features* que nos permitan estimar el valor de las propiedades aproximando las observaciones del conjunto de datos a través de un *hiperplano* (un "hiperplano" es la generalización de una recta o un plano a muchas dimensiones). Concretamente para cada propiedad se considerarán las siguientes *features*:

- Nivel Socioeconómico (NS) del barrio
- Metros Cuadrados (MC)
- Cantidad de Ambientes (CA)

De modo tal que el hiperplano será de la forma:

$$\hat{VP} = \beta_0 + \beta_1 NS + \beta_2 MC + \beta_3 CA$$

donde $\hat{VP}$ será el precio estimado de la propiedad y los $\beta_i$ son los coeficientes del hiperplano que mejor ajusta las observaciones del conjunto de entrenamiento.

El modelo de regresión lineal se basa en una idea muy simple: minimizar el error de las aproximaciones. O sea, si nuestras observaciones del conjunto de entrenamiento son de la forma: $(NS_i, MC_i, CA_i, VP_i)$ entonces el error entre la predicción del modelo y la realidad inducen un error que podemos cuantificar del siguiente modo:

$$E = \sum_i (VP_i - \hat{VP_i})^2 = \sum_i (VP_i - (\beta_0 + \beta_1 NS + \beta_2 MC + \beta_3 CA))^2$$

Los coeficientes $\beta_i$ se calculan de modo de minimizar dicho error $E$. Por lo tanto, el hiperplano asociado es el que mejor "ajusta" a las observaciones del conjunto entrenamiento.

En esta práctica acotaremos el conjunto de datos del siguientes modo:

- Se considerarán las propiedades de la CABA
- Se considerarán las propiedades que tienen entre 1 y 8 ambientes

Además se deberán descartar aquellas propiedades cuyo valor se encuentre a mayor distancia de 3 desvíos estándar de la media. Consideraremos *outliers* a dichas observaciones.

## 2.1. Ejercicio 1: Sacar outliers

Escribir la función "sacarOutliers" en python que descarte aquellas propiedades que se encuentren a una distancia mayor de 3 desvíos estándar respecto a la media.

Más precisamente la función debe recibir un DataFrame que sea el subconjunto asociado a fijar un determinado barrio y una determinada cantidad de ambientes, por ejemplo: "Almagro", 3 ambientes. Y luego calcule la media ($\mu$) y el desvío estándar ($\sigma$) del valor de la propiedad de dicho conjunto. Y finalmente descarte aquellas propiedades cuyo valor sea superior a 3 desvíos estándar o inferior a tres desvíos estándar, es decir que se deben descartar aquellas propiedades que tengan un valor mayor que $\mu + 3\sigma$ o menor que $\mu - 3\sigma$.

## 2.2. Ejercicio 2: Estimar el nivel socioeconómico

Es importante notar que nuestro conjunto de datos no tiene explícitamente el nivel socioeconómico de los barrios. Por lo tanto debemos estimarlo.

Escribir la función "estimarNivelSocioeconomico" en python. La idea será estimarlo mediante el valor medio de las propiedades de 3 ambientes. La estrategia será la siguiente:

1. Filtrar del conjunto las propiedades de 3 ambientes de CABA
2. Para cada barrio del conjunto anterior:
   *a*) Filtrar los outliers (o sea: ejecutar la función del ejercicio anterior)
   *b*) Calcular la media del valor de las propiedades
3. Ordenar los barrios de acuerdo a las medias calculadas (de forma ascendente)
4. Devolver un diccionario con los siguientes pares (clave,valor): (Barrio, Nro. de orden de la media). Por ejemplo si la lista del punto anterior es: (Constitución, Almagro, Belgrano), el diccionario deberá contener los pares: (Constitución, 1), (Almagro, 2), (Belgrano, 3)

## 2.3. Ejercicio 3: Generar el conjunto de datos

Escribir el programa "generar_dataset.py" en python que genere el dataset que se utilizará para entrenar el modelo.

La secuencia de pasos debería ser la siguiente:

1. Considerar el conjunto de propiedades de la CABA que tengan entre 1 y 8 ambientes
2. Conservar las propiedades que tengan todas las columnas definidas: barrio, cantidad de ambientes (CA), metros cuadrados (MC) y el valor de la propiedad (VP)
3. Para cada par (barrio, cantidad de ambientes) del conjunto anterior:
   *a*) Filtrar los outliers (o sea: ejecutar la función del ejercicio anterior)
   *b*) Para cada propiedad imprimir por pantalla la siguiente información: NS,MC,CA,VP (notar que para determinar NS se requiere el diccionario del punto anterior)

Finalmente hay que generar el archivo "dataset.csv" redireccionando el output a un archivo, por ejemplo:

```
generar_dataset.py > dataset.csv
```

## 2.4. Ejercicio 4: Generar el modelo de regresión multilineal

En base al modelo de *regresión multilineal* descripto en el capítulo 15 del libro escribir el programa "regresion_lineal.py" que construya un modelo. El programa debe imprimir por pantalla los coeficientes y el *r-squared* del modelo. El *r-squared* mide la capacidad del modelo de explicar la variablidad de los datos.

Referencias:

- Para mayor información sobre *r-squared*, ver pág. 246 del libro

## 2.5. Ejercicio 5: Verificar los coeficientes con *sci-kit learn*

La librería de python *sci-kit learn* implementa diversos modelos de aprendizaje automático.

Escribir un programa que genere un modelo de regresión lineal sobre el mismo conjunto de datos. El objetivo es comparar los coeficientes con los del programa anterior.

Referencias:

- `https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`

## 2.6. Ejercicio 6: Estimar el desvío de los coeficientes con *bootstrap*

La técnica *bootstrap* permite cuantificar el desvío estándar de los coeficientes calculados. Si los coeficientes tienen poco desvío entonces tendremos garantía de que sus respectivos valores son "correctos".

La técnica esencialmente consiste en muestrear (con reposición) varias veces los datos. Para cada muestra hay que calcular los coeficientes. Y finalmente hay que calcular el desvío estándar asociado a cada coeficiente. Ver la sección "Standard Errors of Regression Coefficients" (página 263 del libro).

Escribir un programa (basado en el ej. 4) que implemente bootstrap para calcular el desvío estándar de los coeficientes. Considerar 100 repeticiones.

Referencia:

- `https://en.wikipedia.org/wiki/Bootstrapping_(statistics)` (ver ejemplo en la sección "Approach")

## 2.7. Ejercicio 7: Estimar el error del modelo con *k-fold cross-validation*

El método *k-fold cross-validation* permite cuantificar el poder predictivo de un método de aprendizaje. El método sigue esta secuencia:

1. Particionar al azar el conjunto de datos en $k$ subconjuntos
2. Para cada partición $S$
   *a*) Entrenar el modelo con todas las otras particiones excepto $S$
   *b*) Utilizar la partición $S$ para validar el modelo calculado. Validar el modelo es equivalente a calcular el error cuadrático medio de las predicciones. O sea:

$$MSE_i = \frac{1}{|S|} \sum (VP - \hat{VP})$$

Donde $|S| = \frac{n}{k}$ es el tamaño de las particiones, $VP$ es el valor real de la propiedad y $\hat{VP}$ es la predicción hecha por el modelo. Más concretamente esto equivale a calcular el promedio de los errores. Finalmente hay que promediar todos los MSE calculados para cada partición:

$$CV = \frac{1}{k} \sum_i MSE_i$$

Escribir un programa (basado en el ej. 4) que calcule $CV$ con *k-fold cross-validation* para $k = 10$.

Referencia:

- `https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation`