Práctica 2 sobre Regresión

En esta práctica vamos a aprender a realizar regresión lineal y logística con Scikit-learn.

PARTE A – Regresión lineal

Instalación

(Si no la tenéis aún instalada de la práctica anterior). Para instalar la librería que se necesita, vamos a instalar la librería Scikit-learn:

https://scikit-learn.org/

Para instalar Scikit-learn se necesita los siguientes requisitos previos:

- Python (>= 2.6 or >= 3.3),
- NumPy (>= 1.6.1),
- SciPy (>= 0.9).

En la siguiente web, se explica cómo instalar esta librería para cada sistema operativo:

https://scikit-learn.org/0.16/install.html

Por ejemplo, en el caso de Windows, se debe ejecutar lo siguiente desde la consola de comandos:

$> pip install -U scikit-learn

Puede variar dependiendo como tengáis configurado vuestro entorno de Python en vuestro editor/herramienta correspondiente.

Aprendizaje

Tómate tu tiempo para explorar a fondo la web de Scikit-learn: https://scikit-learn.org/

Mira cómo está organizada la web y qué grandes apartados de aprendizaje automático soporta. Entre en cada uno de ellos viendo toda su estructura y entra a leer aquellos aspectos que te llame más la atención. En concreto, mira con más detalle el apartado de regresión y busca la parte de regresión lineal.

Después de esta revisión global, céntrate en cómo se puede aplicar regresión lineal por minimización de la suma de los cuadrados de los errores, siguiendo y haciendo los ejemplos de esta web:

https://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares

En concreto, usa la clase LinearRegression, y muestra un ejemplo de dos variables de entrada, con el siguiente código:

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression()
>>> reg.coef_
array([0.5, 0.5])
```

Recuerda que este código se ejecuta sobre línea de comandos de Python, mostrando también la salida, pero se debe adaptar para ejecutarlo desde un archivo Python. Por ejemplo, se necesitaría usar el comando “print” para mostrar los coeficientes (`print(“Coefficients: ”, reg.coef_)´). Ejecuta dicho ejemplo.

Entra en la documentación de clase LinearRegression, pinchando en enlace de la anterior web, y observa todos los atributos y operaciones que tiene dicha clase. En concreto aprende el significado del atributo intercept_

Práctica

En una secuencia de tiempo de 1 minuto, tenemos una pulsera inteligente que mide la frecuencia cardiaca (FC). Nos gustaría saber cuál es la regresión lineal de dichos pares tiempo y FC para saber en qué medida están incrementando los valores o disminuyendo.

El primer paso será generar los datos realistas a partir de un coeficiente y una variación aleatoria. Para ello, se debe generar un array “x” con los instantes de tiempos (i.e. de 1 a 60, representando los segundos de un minuto) y un array “y” con las frecuencias cardiacas para dichos tiempos. Por ejemplo, podrías usar la siguiente fórmula, pero también es bueno que pruebas con otras fórmulas:

Y = 0.7*x + 60 + <número aleatorio entre -5.9 y +5.9>

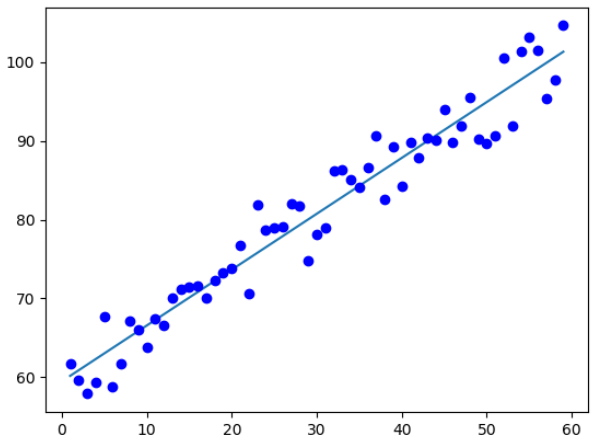
Para generar “x” puedes usar función “arange(inicio, fin)” de “numpy”. Para generar “y” puedes usar operaciones matemáticas sobre array y la operación “random.uniform(inicio, fin)” para generar los números aleatorios. Se puede implementar en un método llamado “generarDatosFC” que devuelva una secuencia “x, y”.

El segundo paso será implementar una función que realice toda esta parte de la práctica: que tome estos datos generados y que haga lo siguiente.

- Adaptar los datos de entrada X para la entrada LinearRegression. En la x, se necesita [[1], [2], …] en vez de [1, 2, …]. Se puede hacer con las “List Comprehensions”.
- Obtener el coeficiente de la función lineal aprendida y la parte independiente (i.e. intercept), y mostrarlos por pantalla.
- Obener los datos “y” (e.g. “yLinealResult”) de la función lineal aprendida.
- Usar Matplotlib para mostrar los puntos de mediciones de FC generados y la función lineal aprendida.

Un ejemplo de salida podría ser lo siguiente:

Coefficients: [0.70915241] Intercept: 59.47330187403128



Se observará en qué medida se puede estimar apropiadamente la regresión lineal, comparando los valores usados para la generación y los valores aprendidos en la regresión lineal.

Explora otras cualidades de LinearRegression y comenta los atributos que más te han llamado la atención.

PARTE B – Regresión logística

Aprendizaje

Mira con detalle el apartado de regresión y busca la parte de regresión logística, partiendo de la web de Scikit-learn: https://scikit-learn.org/

En concreto, céntrate en cómo se puede aplicar la siguiente regresión logística: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

Entra en la documentación de clase LogisticRegression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

Usa la clase LogisticRegression, y observa el siguiente ejemplo:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

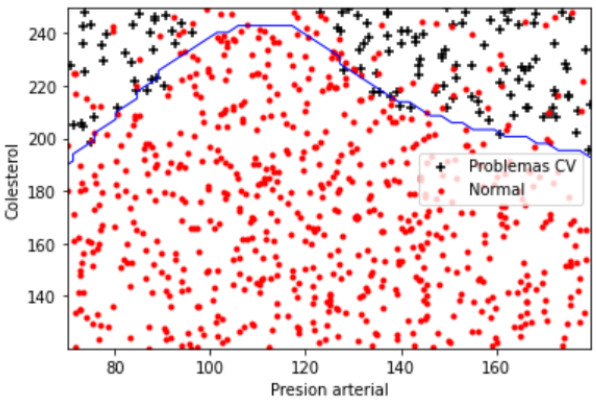
En general, observa todos los atributos y operaciones que tiene dicha clase. En concreto aprende el significado del atributo intercept_ y otros que te resulten interesantes.

Práctica

El hecho de si hay riesgo cardiovascular depende de diversos factores como por ejemplo la presión arterial y el colesterol.

1. Genera un dataset sintético que va a contener 800 instancias que representen personas. Cada persona va a tener los siguientes atributos:
   - Valores de presión arterial y colesterol. Ten en cuenta que el nivel normal de colesterol se considera por debajo de 180, aunque puede variar entre 120 y 250, y el nivel normal de presión arterial sistólica se encuentra entre 100 y 120, aunque puede variar entre 70 y 180.
   - Valores aleatorios de otro atributo que simule otros datos de la persona (pueden ser inventado).
2. Para cada instancia, calcula la probabilidad de tener problemas vasculares. Ten en cuenta las consideraciones sobre los valores de presión arterial y colesterol.
3. En base a un umbral, para cada instancia (teniendo en cuenta la probabilidad calculada en el punto 2), determina la clase de tener riesgo cardiovascular (1) o no (0).

4. Dibuja la gráfica de los datos con respecto a la presión arterial (eje X por ejemplo) y el colesterol (eje Y por ejemplo). Muestra las predicciones utilizando diferentes dibujos para cada punto. Al mostrar los datos generados, observa si guardan cierta coherencia, para que sea posible al regresor aprender. Si lo necesitas, cambia los datos generados.



5. Entrena un regresor logístico (e.g. con clase LogisticRegression) con esos datos.
6. Vuelve a dibujar la gráfica de los datos, dibujando también la curva de la regresión que separa las clases.
7. Genera regresión logística para otro conjunto de personas (500) instancias y dibuja la misma gráfica que en el punto 6. Compara el resultado de la gráfica obtenida con el nuevo conjunto de personas y con el conjunto de personas anterior.

Todo el código, decisiones tomadas, y análisis de resultados y gráficas debe ser comentado y justificado debidamente.

Normas de Entrega

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual. Se entregarán dos archivos jupyter notebook, uno por cada parte, incluyendo el código desarrollado y los comentarios y gráficas que se estimen más adecuados para explicar los resultados obtenidos.