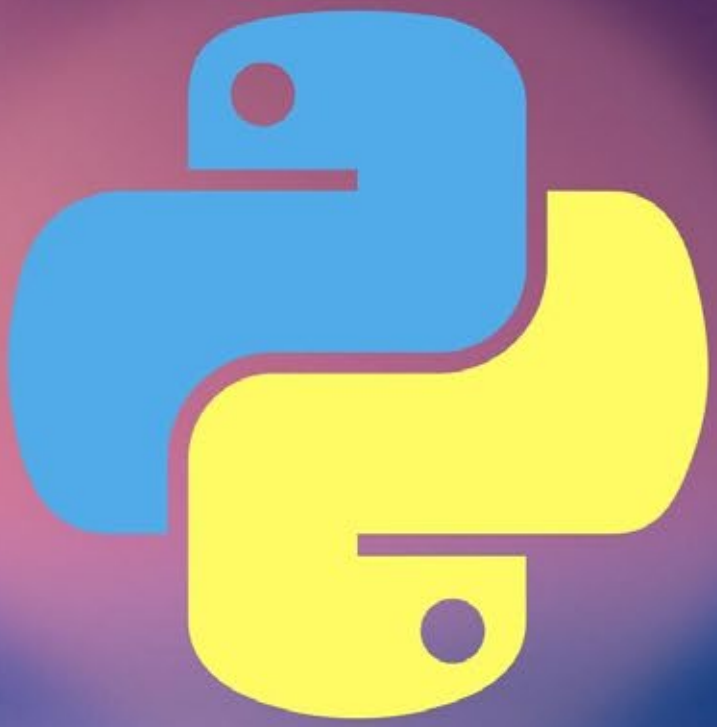


Ejemplos de aprendizaje supervisado

Metodologías de visualización de datos

Servicio Nacional de Aprendizaje - SENA

Regresión Lineal

Según los análisis estadísticos de este modelo, el desvío que se obtiene es pequeño, por lo tanto, el modelo es confiable para su interpretación. A continuación se presentan capturas de este proceso.

Figura 1 Paso 1 - Importación de librerías

```
# importando pandas, numpy y matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# importando los datasets de sklearn
from sklearn import datasets

boston = datasets.load_boston()
boston_df = pd.DataFrame(boston.data, columns=boston.feature_names)
boston_df['TARGET'] = boston.target
boston_df.head() # estructura de nuestro dataset.
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGET
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

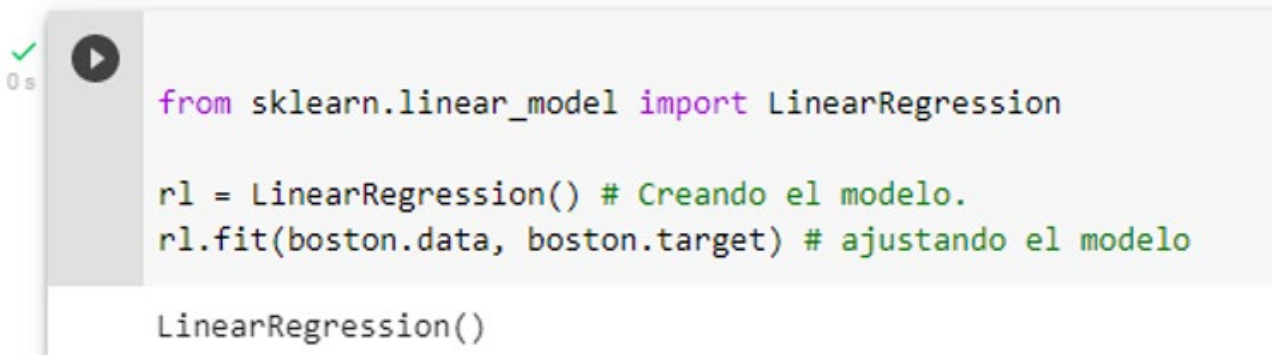


Figura 3 Paso 3 - Listado de coeficientes.

```
# Lista de coeficientes B para cada X
list(zip(boston.feature_names, rl.coef_))
```

```
[('CRIM', -0.10801135783679545),
 ('ZN', 0.04642045836688176),
 ('INDUS', 0.02055862636707862),
 ('CHAS', 2.6867338193448966),
 ('NOX', -17.766611228300167),
 ('RM', 3.809865206809212),
 ('AGE', 0.0006922246403425021),
 ('DIS', -1.475566845600255),
 ('RAD', 0.30604947898517226),
 ('TAX', -0.01233459391657437),
 ('PTRATIO', -0.9527472317072923),
 ('B', 0.009311683273793711),
 ('LSTAT', -0.5247583778554923)]
```

Figura 4 Paso 4 - Ejecución de predicciones.

```
# haciendo las predicciones
predicciones = rl.predict(boston.data)
predicciones_df = pd.DataFrame(predicciones, columns=['Pred'])
predicciones_df.head() # predicciones de las primeras 5 lineas
```

	Pred
0	30.003843
1	25.025562
2	30.567597
3	28.607036
4	27.943524

Figura 5 Paso 5 - Cálculo del desvío

```
# Calculando el desvio
np.mean(boston.target - predicciones)
```

```
2.5276223801742696e-16
```

Regresión logística

Entre los modelos lineales, se encuentran los de regresiones logísticas, que también son utilizados para clasificaciones; en este caso, se ajusta el modelo lineal pensando en que cierta clase o categoría suceda. La función que utiliza este modelo es la siguiente:

$$f(x) = \frac{1}{1 + e^{-1}}$$

Realizando un pequeño ejemplo en Python, tenemos el siguiente código:

Figura 6 Código inicial - Código inicial

```
# Creando un dataset de ejemplo
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4)
```

En la anterior imagen, se observa la creación de un *dataset* para el desarrollo de este sencillo ejemplo:

Figura 7 - Creación de un *dataset*

```
[1] # Creando un dataset de ejemplo
    from sklearn.datasets import make_classification
    X, y = make_classification(n_samples=1000, n_features=4)
```

```
# Importando el modelo
from sklearn.linear_model import LogisticRegression

rlog = LogisticRegression() # Creando el modelo

# Dividiendo el dataset en entrenamiento y evaluacion
X_entrenamiento = X[:-200]
X_evaluacion = X[-200:]
y_entrenamiento = y[:-200]
y_evaluacion = y[-200:]

rlog.fit(X_entrenamiento, y_entrenamiento) #ajustando el modelo

# Realizando las predicciones
y_predic_entrenamiento = rlog.predict(X_entrenamiento)
y_predic_evaluacion = rlog.predict(X_evaluacion)
```

Figura 8 Verificación del modelo

```
# Verificando la exactitud del modelo
entrenamiento = (y_predic_entrenamiento == y_entrenamiento).sum().astype(float) / y_entrenamiento.shape[0]
print("sobre datos de entrenamiento: {0:.2f}".format(entrenamiento))
evaluacion = (y_predic_evaluacion == y_evaluacion).sum().astype(float) / y_evaluacion.shape[0]
print("sobre datos de evaluación: {0:.2f}".format(evaluacion))
```

```
sobre datos de entrenamiento: 0.96
sobre datos de evaluación: 0.95
```

En este ejemplo, se observa la precisión que tiene el modelo clasificando las categorías del dataset que se declaró.

Árboles de decisión

Los árboles de decisión tienen cierta similitud a los diagramas de predicción basados en reglas, ya que en ambos casos su objetivo consiste en la secuencia de pasos lógicos que conllevan la resolución de un problema.

Los árboles de decisión están compuestos por nodos terminales y ramas que salen de los nodos interiores, estos últimos nodos contienen en su interior un atributo y cada rama representa un valor de ese atributo; cada rama hacia abajo conecta con un nodo, creando una segmentación de los datos. Ejemplarizamos lo anterior con este ejemplo en Python.

Figura 9 Árbol de decisión

```
# Importando el arbol de decisión
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

ad = DecisionTreeClassifier(criterion='entropy', max_depth=5) # Creando el modelo
ad.fit(X, y) # Ajustando el modelo

#generando archivo para graficar el arbol
with open("mi_arbol.dot", 'w') as archivo_dot:
    tree.export_graphviz(ad, out_file = archivo_dot)
```

```
# utilizando el lenguaje dot para graficar el arbol.
!dot -Tjpeg mi_arbol.dot -o arbol_decision.jpeg
```

Luego de utilizar el lenguaje dot para convertir la imagen en formato *.jpg, ya se puede visualizar su imagen.

Figura 10 Verificando la precisión

```
# verificando la precisión
print("precisión del modelo: {0: .2f}".format((y == ad.predict(X)).mean()))
```

```
precisión del modelo:  0.93
```

En este ejemplo, el árbol tiene una precisión del 89 %, sin embargo no sobra decir que los árboles de decisión tienden a sobreentrenar los datos.

Random Forest

La idea principal de este algoritmo es utilizar muchos árboles de decisión en lugar de uno.

"¿Por qué utilizar un solo árbol de decisión si podemos utilizar todo un bosque?" es la frase que rodea el entorno de este modelo, el enfoque que dan es en la construcción de muchos árboles de decisión no muy profundos y luego se toma la clase de cada árbol elegido.

Este procedimiento es muy potente en machine learning. Si contemplamos la idea de que un solo clasificador entrenado obtenga un resultado de acierto del 60 %, se puede entrenar un número considerable de clasificadores que sean acertados, para después aprovechar la sabiduría de todos los clasificadores juntos. Es la teoría que se predica y la más utilizada.

En Python, se haría de la siguiente manera:

Figura 11 Ejemplo de *radom forest*

```
# Importando el random forest
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier() # Creando el modelo
rf.fit(X, y) # Ajustando el modelo

# verificando la precisión
print("precisión del modelo: {0: .2f}".format((y == rf.predict(X)).mean()))
```

```
precisión del modelo:  1.00
```

SVM (Máquinas de vectores de soporte)

La idea dentro de este algoritmo es encontrar un plano que separe grupos dentro de los datos, de manera correcta y entendible. Cuando se habla de separación, se habla de que la elección del plano maximiza el margen entre los puntos más cercanos del plano; a estos puntos se les llama vectores de soporte; en Python, se vería de la siguiente manera:

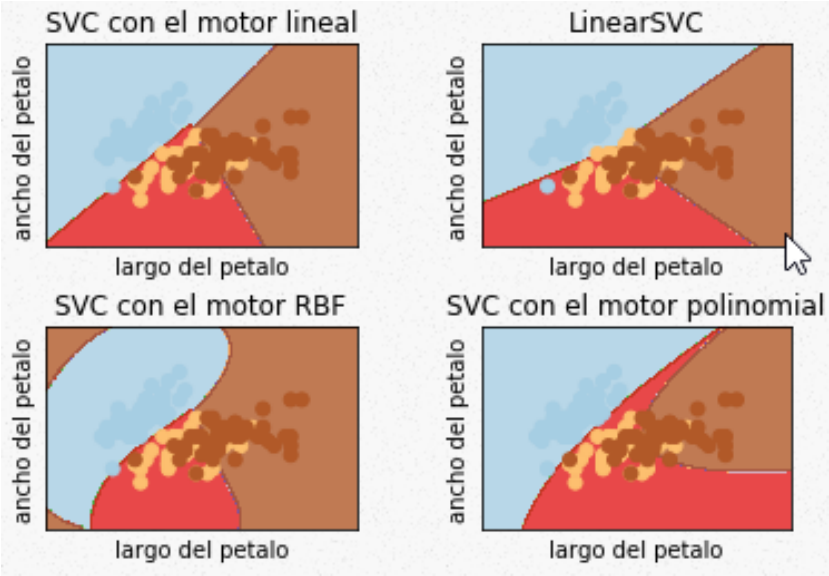
Figura 12 Ejemplo SVM

```
# importanto SVM
from sklearn import svm

# importando el dataset iris
iris = datasets.load_iris()
X = iris.data[:, :2]  # solo tomamos las primeras 2 características
y = iris.target

h = .02  # tamaño de la malla del grafico

# Creando el SVM con sus diferentes métodos
C = 1.0  # parametro de regulacion SVM
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
lin_svc = svm.LinearSVC(C=C).fit(X, y)

# crear el area para graficar
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

# titulos de los graficos
titles = ['SVC con el motor lineal',
          'LinearSVC',
          'SVC con el motor RBF',
          'SVC con el motor polinomial']
```

Figura 13 Código para graficación

```
for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
    # Realizando el gráfico, se le asigna un color a cada punto
    plt.subplot(2, 2, i + 1)
    plt.subplots_adjust(wspace=0.4, hspace=0.4)

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Paired, alpha=0.8)

    # Graficando tambien los puntos de datos
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
    plt.xlabel('largo del petalo')
    plt.ylabel('ancho del petalo')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title(titles[i])

plt.show()
```

Figura 14 Ejemplo gráfico de un SVC



Estos son algunos ejemplos de los algoritmos de regresión más utilizados en la ciencia de datos.