Tecnológico de Monterrey

MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA

MATERIA
Ciencia y Analítica de Datos

PROFESOR:
Dra. María de La Paz Rico

ALUMNO:

Carlos Enriquez Gorgonio
A01793102

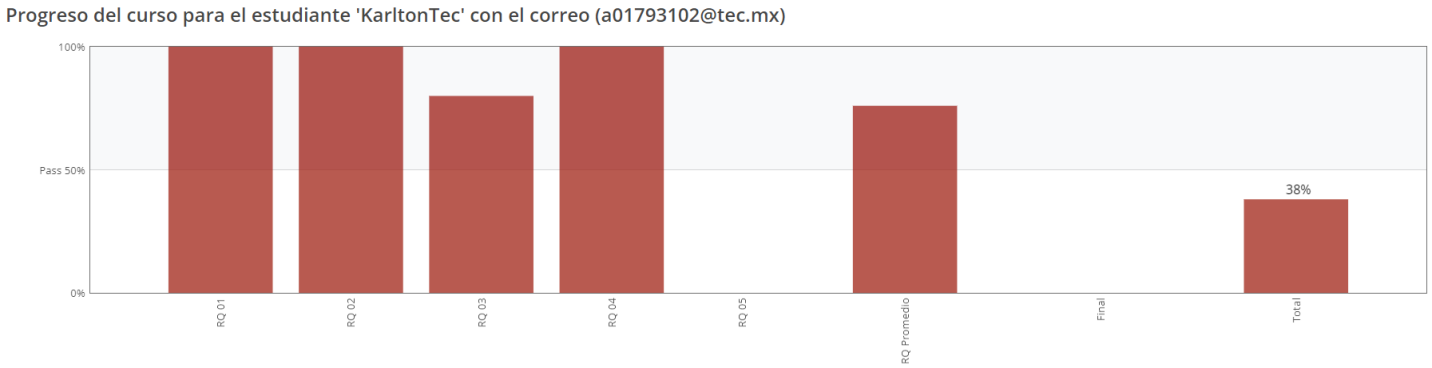
Curso IBM
Módulos 1 añ 4

Noviembre de 2022

El vínculo a la actividad en GitHub es:

https://github.com/PosgradoMNA/actividades-de-aprendizaje-KarltonBotics

}



Para leer un archivo csv usamos el método read, de la siguiente forma

pandas.read_csv()

Para descargar un archivo de internet

path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/auto.csv"

await download(path, "auto.csv")

path="auto.csv"

df = pd.read_csv(path, header=None)

Para imprimri los 10 primeros elementos del encabezado de nuestro data set, usamos head, y para los últimos tail

print("The last 10 rows of the dataframe\n")
df.tail(10)

Para reemplazar un valor, con un NAN, para que lfuncion dropna puede remover valores faltantes usamos

df1=df.replace('?',np.NaN)

Para borrar los valores de una columna usamos dropna, en el eje 0, en el eje 1 son filas horizontales

df=df1.dropna(subset=["price"], axis=0)

Para imprimr las columnas utilizamos:

print(df.columns)

Para guardar un dataset elaborado en un archivo CSV o coma separated values usamos

df.to_csv("automobile.csv", index=False)

Para ver el tipo de un dataset

print(df.dtypes) o solo df.dtypes

Par ver un resumen de datos de un dataset usamos

dataframe.describe()

Para la información usamos

dataframe.info()

Para seleccionar las columnas de un data frame las nombramos de la siguiente forma

dataframe[[' column 1 ',column 2', 'column 3']]

Para describir el tipo de cada columna usamos

dataframe[[' column 1 ',column 2', 'column 3']].describe()

Para ver el tipo de datos de una columna seleccionada usamos

df[['length', 'compression-ratio']].describe()

Para crear una variable que contenga los encabezados, los ponemos entre corchetes y los nombres de las columnas entre comillas

```
headers = ["symboling","normalized-losses","make","fuel-type","aspiration", "num-of-doors","body-style",
    "drive-wheels","engine-location","wheel-base", "length","width","height","curb-weight","engine-type",
    "num-of-cylinders", "engine-size","fuel-system","bore","stroke","compression-ratio","horsepower",
    "peak-rpm","city-mpg","highway-mpg","price"]
```

Para crear un nuevo data frem asignando el paremtro nombres igual a la lista encabezados usamos

```
df = pd.read_csv(filename, names = headers)
```

Para reemplazar los valores de que simblicen signo de interrogación con los valores NaN usamos

```
df.replace("?", np.nan, inplace = True)
```

isnull() y notnull(), son métodos que nos permiten identicar datos faltantes

```
issing_data = df.isnull()
```

Para imprimir sumatorias de datos falntes en cada columna usamos

```
for column in missing_data.columns.values.tolist():
    print(column)
    print (missing_data[column].value_counts())
    print("")
```

Para reemplazar valores NaN, con el promedio o mean usamos

```
df["normalized-losses"].replace(np.nan, avg_norm_loss, inplace=True)
```

#considerar que en ese nombre de la columna es avg_norm_loss

Para calcular el promedio de los valores de una columna, en este caso llamad #pak-rpm”

```
avg_peakrpm=df['peak-rpm'].astype('float').mean(axis=0)
```

Usar el método value_counts() para contar la cnidad de valores

```
df['num-of-doors'].value_counts()
```

Para usar el método calue_counts(), añadiendo el .idmax(), nos permite calcular el calor mas común de un tipo en una columna

```
df['num-of-doors'].value_counts().idxmax()
```

.dtype() para revisar el tipo de un dato

.astype() para cambiar el tipo de un dato

Ejemplo

```
df[["bore", "stroke"]] = df[["bore", "stroke"]].astype("float")

df[["normalized-losses"]] = df[["normalized-losses"]].astype("int")
```

En la estandarización de datos, para convertir de un valor a otro , podemos usar las formuals directamente de nuestras listas, por ejemplo

De una columna de litros por 100 kilometros, que queremos convertir desde una columna que tiene datos en millas por galon mpg, hacemos una nueva columna la cual obtendrá valores aplicando directamente la formula desde otra columna

L/100km = 235 / mpg

```
df['city-L/100km'] = 235/df["city-mpg"]
```

Después de crear una nueva columna, podemos renmobrarla, ya que tenga los contenidos

```
df.rename(columns={'highway-mpg':'highway-L/100km'}, inplace=True)
```

También se puede reemplazar el valor de una columna sin tener que crear una nueva, aplicandole una formula a la misma

```
df['length'] = df['length']/df['length'].max()
```

binning es el proceso de transofrma calore snumerocisocontios en categóricas discretas, llamadas bins

Por ejemplo si tenemos una lista de caballos de fuerza del 48 al 288 y lo queremos convertir a solo 3 tipos de valores, con la media, el mas algo y el menor, podemos re arreglar el arrenglo de la siguiente forma

Primero lo hacemos un solo tipo de valor entero

```
df["horsepower"]=df["horsepower"].astype(int, copy=True)
después vemos las graficas para ver ladistribición original de datos
%matplotlib inline
import matplotlib as plt
from matplotlib import pyplot
plt.pyplot.hist(df["horsepower"])

# set x/y labels and plot title
plt.pyplot.xlabel("horsepower")
plt.pyplot.ylabel("count")
plt.pyplot.title("horsepower bins")
```

Después defininmos nuestros bins, que serán nuestros nuevos elementos que agruparan

```
bins = np.linspace(min(df["horsepower"]), max(df["horsepower"]), 4)
bins
```

Le ponemos nombre a los grupos de nuestros bins

```
group_names = ['Low', 'Medium', 'High']
```

Exploratoy Data Analysus o EDA, es una etapa de analisis preliminar de analisis de datos

En esta etapa se realizan las sigueintes acciones:

Se resumen las principales caracteristicas de los datos

Se obtiene mejor entendimiento del data set

Se revelan o uncover relaciones entre variables

Se extraen variables importantes

La estadística descriptive

Nos permiten describe las características básicas de nuestro ata set y para tomar medidas y muestras de nuestros datos

Un ejemplo de esto es usando la función describe() de pandas

Dataframe.describe(), nos devuelve estadisticas para ese data set

Recordemos que los datos categóricos son aquellos que pueden ser subdivididos en diversas categorias o grupos, por ejemplo, en una columna, pude haber números del 1 al 7, pero estos s u vez son solo índices de otros datos o cifras, por ejemplo 1 puede ser hombre, 2 mujer, 3 niño, etc.

Para resumir datos categóricos témenos value_counts()

Sumatoria_de_columna= dataframe("columnadeseada").valuecounts()

Le ponemos nombre a la columna nueva que tiene las sumatorias

Sumatoria_de_columna.rename (columns={"tipo":"sumatoria", inplace =True})

Seleccionamos la columna que sera nuestro indice

Sumatoria_de_columna.index.nae="tipo"

La función de impresión o boxplot

Recordar que necesitamos la librería que jala el sns

Sns.boxplot(x="columnadeseada", y="otracolumnadeseada", data=dataframedesaeso)

Un ejemplo de x pueden ser tipo s de lalantas, y de y los precios de estas

En ocasiones cada observación representa un punto, por ejemplo el costo de una unidad el tamaño de su motor, pueden marcar un punto donde se interserctan

Para plotear la relación entre dos variables usamos scatter plot show

Nuestro valor en el eje de las x o x-axis es la variable predictora

El valor en el eje de las y es nuestra variable objetivo o target variable, y es la que intentamos predecir.

Por ejemplo si quier saber cuanto aumentaría en precio el costo de un carro de motor gra, de ponemos en el eje de las x la predictora, el tamaño del motor y en el eje de las y el valor del carro

Seria algo así:

y = ("tamaño_motor)

x = ("precio")

plt.scatter(x,y

ahora le ponemos títulos a nuestros ploteos

plt.title ("Scatterplot de tamaño de motor vs precios")

plt.xlabel("Tamaño del motor")

plt.ylabel("Precio del carro")

cuando vemos que las gracias van teniendo puntos que aumentan tanto en el eje de las x como el de las y, es un indicador que nuestras variables podrían tener una probable relación lineal

Un modelo o un estimador puede interpretarse como una ecuación matemática utilizada para predecir uno o varios valores o para relacionar una o más variables independientes. Entre mas relevante información se tiene, mas exactitud muestra el modelo.

Un ejemplo es un estimador de precios de carros, en donde le pones varias variables excepto el color, para predecir el precio de un carro, pero resulta que el color es un factor determinante de precios, entonces el modelo nunca encontrara la diferencia de precios basadas en color. Por eso es importante las variables significativas

La regresión lineal simple, se refiere a una variable independiente para hacer una prediccion

Se abrevia SLR y tiene dos elementos un predictor en el eje de las X y un objetivo en el eje de las y

Para usar la regresión lineal, primero imporatamos la librería

Después creamos una variable que contendrá el objeto de la regresión lienal

1. Import linear_model from scikit-learn

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

2. Create a Linear Regression Object using the constructor :

```
lm=LinearRegression()
```

Después definimos la variable predictora y la variable objetivo, x y y, y les asignamos una columna o una propiedad

Ahora si alimentamos el modelo, con el método fin aplicado a la función linear regresión

Después obtenemos y hat, que contiene un data frame con los valores de la predicción

- We define the predictor variable and target variable

```
X = df[['highway-mpg']]
Y = df['price']
```

- Then use lm.fit (X, Y) to fit the model , i.e fine the parameters b_0 and b_1

```
lm.fit(X, Y)
```

- We can obtain a prediction

```
Yhat=lm.predict(X)
```

Yhat	X
2	5
:	
3	4

COGNITIVE CLASS

Teniendo esto, también podemos obtener los valores de intercepción y el coeficiente

SLR – Estimated Linear Model

- We can view the intercept (b_0): `lm.intercept_` → `38423.305858`
- We can also view the slope (b_1): `lm.coef_` → `-821.73337832`
- The Relationship between Price and Highway MPG is given by:
- **Price = 38423.31 - 821.73 * highway-mpg**

$$\hat{Y} = b_0 + b_1 x$$

La regresión lineal múltiple, se refiere a múltiples variables independientes para hacer una predicción

Ocurre lo mismo con la regresión lineal multiple

Fitting a Multiple Linear Model Estimator

1. We can extract the for 4 predictor variables and store them in the variable Z

```
Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Then train the model as before:

```
lm.fit(Z, df['price'])
```

3. We can also obtain a prediction

```
Yhat=lm.predict(X)
```

x_1	x_2	x_3	x_4
3	5	-4	3
:	:	:	:
2	4	2	-4

Yhat
2
:
3

COGNITIVE

Los plots o graficas de regresión nos permiten estimar

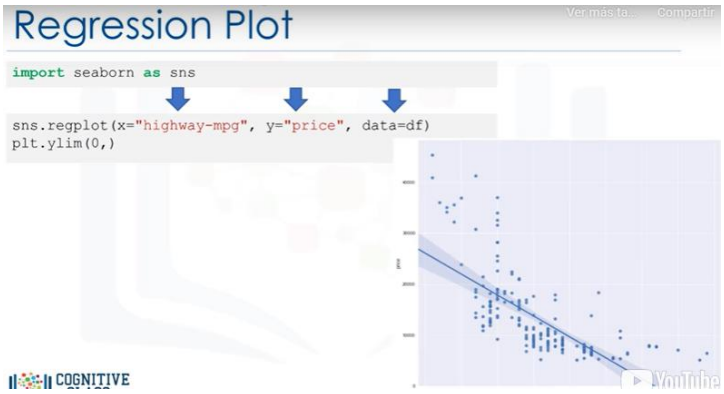
La relación entre dos variables

La fortaleza o dureza de la correlación

La dirección de la relación, si van en sentido positivo o negativo

Para plotear una regresión, usamos la librería seaborn, y la función regplot

Indicamos nuestra variable en el eje x y en el eje y y nuestra información

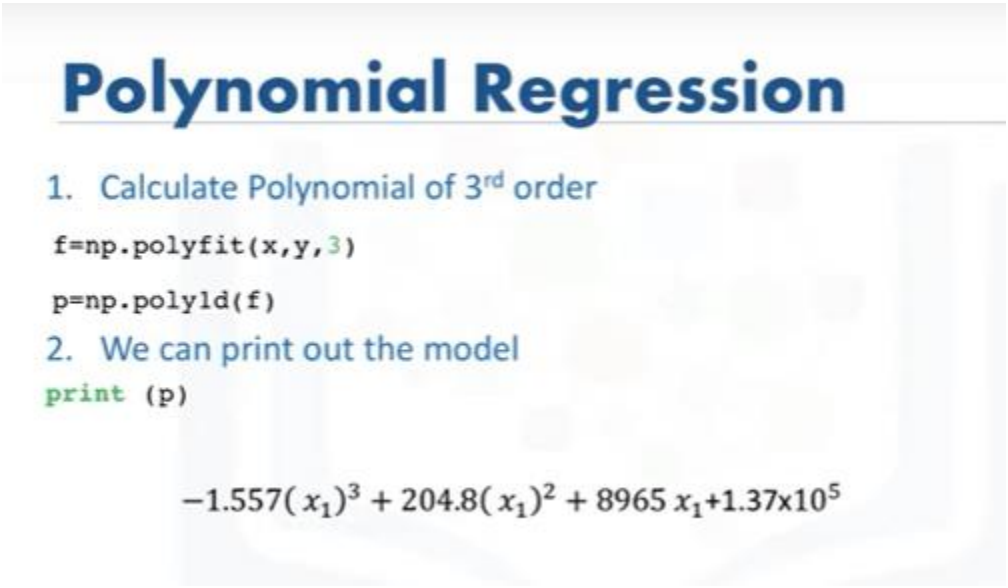


Los plot resuduales, nos permite ver la distribución de los datos, entre mas grande la varianza, el plot esta mal

Un plot de distribución grafica los valores predecidos conta los valores reales

La regresión polinomial nos permite usar los modelos de regresión lineal, para describir relaciones curvilíneas. La relación curvilínea se obtiene obteniendo los cuadrados o determinando los términos mayores de las predicciones variables

Para usar la regresión, usamo la función polyfit



También existe la regresión polinomial de mas deuna dimension

Antes de aplicar estas funciones, necesitamos hacer un pre procesamiento, para lo cual podemos usar el standarsd scales

Pre-processing

- For example we can Normalize the each feature simultaneously:

```
from sklearn.preprocessing import StandardScaler

SCALE=StandardScaler()
SCALE.fit(x_data[['horsepower', 'highway-mpg']])

x_scale=SCALE.transform(x_data[['horsepower', 'highway-mpg']])
```

Al final de nuestra normalización, la asignamos a un nuevo dataframe

Podemos usar los pipelines, para hacer muchas tranformaciones de forma simultanea

El primer elemento de los tuples de los pipelines, contiene el nombre del estimador en forma de etiqueta y el segundo elemento contiene el constructor del modelo

Pipeline Constructor

```
Input=[('scale',StandardScaler()),('polynomial',PolynomialFeatures(degree=2),…
('model',LinearRegression())]
```

- Pipeline constructor

```
pipe=Pipeline(Input)
```

pipeline object

- We can train the pipeline object

```
Pipe.train(X['horsepower', 'curb-weight', 'engine-size', 'highway-mpg'],y)

yhat=Pipe.predict(X[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
```