Laboratorio de Datos

Primer Cuatrimestre 2024

Práctica N° 5: Modelo lineal multivariado. Entrenamiento y testeo.

1. Queremos estudiar la relación entre la longitud de la aleta de un pingüino y el peso del pingüino. Como en una esfera, el peso es proporcional a la longitud del radio elevada al cubo, podemos conjeturar que un polinomio de grado 3 es apropiado para ajustar el peso en función de la longitud de la aleta. Queremos verificar si nuestra conjetura tiene sustento en los datos.

(a) Datos faltantes. Ejecutar el siguiente código y observar si hay filas con datos faltantes (NaN).

```
penguins = sns.load_dataset("penguins")
penguins.head()
```

Para hacerlo en forma más sistemática (en lugar de mirar solo algunas filas) podes usar el siguiente código

```
penguins.isnull().values.any()
```

Para eliminar las filas con valores faltantes aplicamos al DataFrame el método `dropna()`. Eliminar las filas con datos faltantes del DataFrame de pingüinos y verificar que el DataFrame resultante no contiene valores faltantes.

(b) Conjuntos de entrenamiento y testeo. Dividir el dataset resultante en un grupo de entrenamiento y uno de testeo (80% - 20%). Hacerlo de las siguientes dos formas distintas:

A. Utilizando un array de Numpy para filtrar:

i. Crear un array de Numpy booleano de longitud igual a la cantidad de filas del DataFrame que tome el valor True en el primer 80% de las coordenadas y False en el restante 20%.

ii. Si se quiere seleccionar una muestra al azar, se puede utilizar `numpy.random.shuffle` para "mezclar" el vector.

iii. Utilizar el vector generado para filtrar el Dataframe.

Siguiendo esos pasos, completar el siguiente código:

```
train_ind = np.full(???, False)
train_ind[0:???] = True
np.random.shuffle(train_ind)  # Lo guarda en el mismo vector.
penguins_train = penguins[???]
penguins_test = penguins[???]
```

B. Utilizando la función `train_test_split` de `sklearn`:

```
from sklearn.model_selection import train_test_split
penguins_train, penguins_test = train_test_split(penguins,
   test_size=0.2, random_state=42)
```

En este codigo, `random_state` se conoce también como semilla aleatoria. Si corremos varias veces el código con el mismo número, obtendremos siempre la misma muestra. Esto permite que el experimento sea reproducible. Si modificamos el número (o no lo indicamos), obtendremos distitnas muestras aleatorias.

Podemos también separar primero las variables explicativas y variable respuesta:

```
from sklearn.model_selection import train_test_split
X = ???
y = ???
X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
```

(c) Crear y ajustar 3 modelos utilizando polinomios de grados 1, 2 y 3.

(d) Calcular para cada uno el error predicción en el grupo de entrenamiento y en el grupo de test.

(e) ¿Cuál modelo tiene el menor error (ECM) en el ajuste? ¿Cuál el menor error (ECM) de predicción?

(f) En base a los resultados obtenidos, ¿cuál de los tres modelos utilizaría?

2. En el archivo `50_startups.csv` tenemos los siguientes datos de 50 compañías: gastos en investigación y desarrollo, gastos administrativos, gastos en marketing y ganancias. Queremos estimar las ganancias a partir de los gastos en las distintas áreas.

(a) Leer el archivo, y realizar un gráfico de dispersión para cada par de variables. Se pueden generar todos los gráficos automáticamente con el `pairplot`.

```
startups = pd.read_csv(???)
sns.pairplot(
    data=startups, aspect = .8)
```

En base a estos gráficos, si quisiéramos predecir la ganancia mediante un modelo lineal utilizando una sola variable predictora, ¿cuál variable utilizaría? Diseñar un experimento para verificar su respuesta.

(b) En este ejemplo, ¿considera que un modelo lineal multivariado ayudaría a predecir mejor la ganancia que el modelo lineal univariado del ítem anterior? Realizar un experimento para verificar su respuesta.

3. En el **Ejercicio 1** no tuvimos en cuenta el sexo del pingüino para predecir el peso, y puede ser una variable importante. Se quiere predecir ahora el peso de un pingüino usando como variables predictoras el largo de la aleta y el sexo del pingüino (utilizar el DataFrame sin datos faltantes, como vimos en el **Ejercicio 1 (a)**).

(a) ¿Cuáles son todos los valores que toma la variable "sex"? ¿Qué tipo de variable es: numérica o categórica, ordinal o nominal? ¿Es una variable binaria?

(b) Escribir (en lápiz y papel) la ecuación de un modelo lineal para este caso. ¿Qué unidades tienen las variables y cómo se codifica la variable "sexo del pinguino"?

(c) **Codificación de variables binarias.** Si utilizamos `Formulaic` para generar la matriz de datos e incluimos la variable sexo en el modelo, automáticamente va a crear una columna con el sexo codificado con 0's y 1's.

Alternativamente, podemos utilizar el siguiente código:

```
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
sex01 = encoder.fit_transform(penguins[["sex"]])
penguins["sex01"] = sex01
```

(d) Ajustar el modelo usando todos los datos disponibles. Reportar los coeficientes encontrados y calcular el error de predicción (ECM). ¿Considera que agregar la variable "sex" mejoró el modelo?

(e) Realizar una visualización apropiada para ver de los datos junto con las predicciones del modelo.

(f) Dos pingüinos que tienen igual largo de aleta, uno macho y otro hembra, ¿qué diferencia de peso predice el modelo que tendrán?

4. Ahora se quiere predecir el peso de un pinguino usando como variables predictoras el largo de la aleta y la especie del pingüino.

(a) Trabajamos con la base de pingüinos sin datos faltantes. ¿Cuáles son todos los valores que toma la variable "species"? ¿Qué tipo de variable es: numérica o categórica, ordinal o nominal? ¿Es una variable binaria?

(b) Escribir (en lápiz y papel) la ecuación de un modelo lineal para este caso. ¿Cómo se codifica la variable "especie"?

(c) Explicar qué diferencia tiene este modelo respecto al propuesto en el ejercicio 1.

(d) **Codificación de variables categóricas.** Si utilizamos `Formulaic` para generar la matriz de datos e incluimos variables categóricas, automáticamente va a crear las columnas indicadoras con 0's y 1's necesarias (variables dummies).

Alternativamente podemos usar el siguiente código que utiliza la función `OneHotEncoder`:

```
from sklearn.preprocessing import OneHotEncoder
penguins = sns.load_dataset("penguins").dropna()
encoderOHE = OneHotEncoder(sparse_output = False)
species3 = encoderOHE.fit_transform(penguins[["species"]])
species3_df = pd.DataFrame(species3,
   columns=encoderOHE.get_feature_names_out(),
   index=penguins.index)
penguins3 = pd.concat([penguins, species3_df], axis = 1)
penguins3.head()
```

Si utilizan este código, verifiquen que los tamaños de `species3` y `penguins3` sean los esperados, y que el DataFrame resultante no tenga datos faltantes (como el DataFrame original no tiene faltantes, este tampoco debería tenerlos, pero nunca está mal verificarlo).

(e) Ajustar el modelo usando todos los datos disponibles. Reportar los coeficientes encontrados y calcular el error de predicción.

(f) Realizar una visualización apropiada para ver de los datos junto con las predicciones del modelo.

5. En este ejercicio trabajaremos con el dataset de inmuebles (`inmuebles.csv`). El objetivo de este ejercicio es comparar tres modelos lineales para predecir el precio de un inmueble:

Modelo 1: $precio \sim superficie$
Modelo 2: $precio \sim superficie + zona$
Modelo 3: un modelo propuesto por usted

Como medida de comparación utilizaremos la raíz cuadrada del ECM, que notaremos RECM. En cada ítem, indicar qué modelo tiene el mejor desempeño.

(a) *Nivel 1.* Entrenar cada modelo sobre la totalidad de los datos disponibles. Graficar, en una misma figura, los datos y las predicciones del modelo 2 y del modelo 3. Las predicciones deben estar señaladas en la leyenda (recuerde el uso del argumento `label`) y las lineas del modelo 3 deben estar punteadas (investigue agregar `linestyle='--'` donde corresponda). Puede adaptar el siguiente código:

```
(
    so.Plot(data=data, x='superficie', y='precio')
    .add(so.Dot(), color='zona')
    .add(so.Line(), y=modelo.predict(X).flatten(), color='zona')
)
```

(b) *Nivel 2.* Separar los datos en conjuntos de entrenamiento (*train*) y prueba (*test*). Entrenar cada modelo sobre el mismo conjunto de entrenamiento y comparar su desempeño sobre el conjunto de prueba.

(c) *Nivel 3.* Separar los datos en dos conjuntos: entrenamiento y testeo. Luego, dividir el conjunto de entrenamiento en dos conjuntos: entrenamiento y validación. Entrenar cada modelo sobre el conjunto de entrenamiento y comparar su desempeño en el conjunto de validación. Elija el que usted considere el más apropiado, entrénelo sobre la unión de los datos de testeo y validación y calcule la RECM y el R^2 sobre el conjunto de testeo.

(d) *Nivel 4 - k-Fold Cross-Validation.* Aplicar validación cruzada en k pliegos con $k = 6$. Para esto, separar el conjunto de datos en entrenamiento y testeo. Aplicar la técnica de k-Fold CV para cada modelo con el conjunto de testeo. Elegir el modelo con menor promedio de RECM y calcular la RECM y el R^2 sobre el conjunto de testeo.