Tecnológico de Monterrey

ITESM Campus Puebla

Integración de seguridad informática en redes y sistemas de software
TC2007B.1

Reporte Módulo 2

Alejandro Castro Reus A01731065

Fecha: 09/09/22

Introducción

Para esta actividad se decidió utilizar un dataset que contiene los datos de los gastos médicos de las personas (variable dependiente) junto con datos personales como edad, sexo, índice de masa corporal, entre otros (variables independientes). Se decidió este modelo porque tiene una cantidad adecuada de filas y pocas columnas.

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

Primer modelo

Como primer acercamiento, se decidió usar un modelo de regresión lineal simple. Como variable independiente se decidió usar bmi (desde ahora x1). Lo primero que se hizo fue dividir el dataset en train y test:

```
#Linear regression with one independent variable
#Separation in train/test
x_train, x_test, y_train, y_test = train_test_split(df["x1"], df["y"], test_size= 0.2, random_state=0)
```

Esto nos permitirá entrenar un modelo y al mismo tiempo saber cómo se comporta en el mundo real. Posteriormente se entrena el modelo:

```
reg = LinearRegression().fit(x_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1))
y_train_pred = reg.predict(x_train.values.reshape(-1,1))
y_test_pred = reg.predict(x_test.values.reshape(-1,1))
```

Se obtienen los coeficientes y el error:

```
print(f"Coeffs: {reg.coef_}, Intercept: {reg.intercept_}")

mse_train = mean_absolute_error(y_train, y_train_pred)
mse_test = mean_absolute_error(y_test, y_test_pred)

print(f"mse_train: {mse_train}, mse_test: {mse_test}")
```

```
Coeffs: [[357.09343567]], Intercept: [2228.39494088]
mse_train: 9082.324056295754, mse_test: 9276.717029711594
```

Y se realizan gráficas relevantes para determinar la varianza y el sesgo:

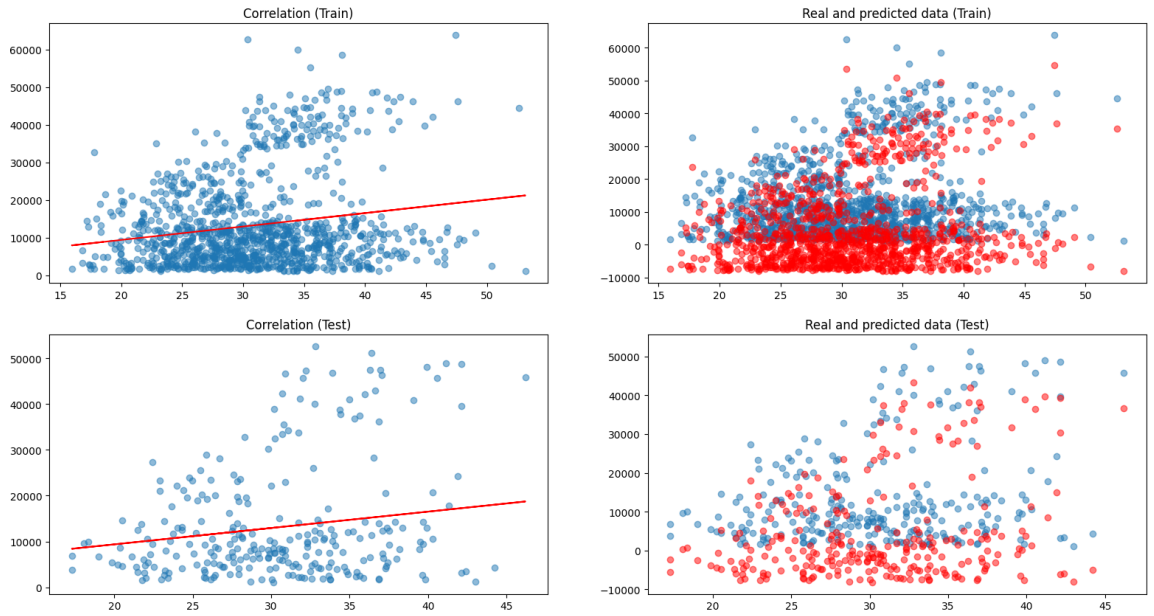
```python
#Bias in models
axis[0, 0].scatter(x_train, y_train, alpha=0.5)
axis[0, 0].plot(x_train, y_train_pred, color="red")
axis[0, 0].set_title("Correlation (Train)")

axis[1, 0].scatter(x_test, y_test, alpha=0.5)
axis[1, 0].plot(x_test, y_test_pred, color="red")
axis[1, 0].set_title("Correlation (Test)")

#Variation in model
axis[0, 1].scatter(x_train, y_train, alpha=0.5)
axis[0, 1].scatter(x_train, y_train - mse_train, color="red", alpha=0.5)
axis[0, 1].set_title("Real and predicted data (Train)")

axis[1, 1].scatter(x_test, y_test, alpha=0.5)
axis[1, 1].scatter(x_test, y_test - mse_test, color="red", alpha=0.5)
axis[1, 1].set_title("Real and predicted data (Test)")

plt.show()
```



En las dos primeras gráficas se hace un scatter plot y se plotea la regresión, en las últimas dos gráficas se grafican los datos reales y los datos reales menos el error medio absoluto.

En las gráficas de la primera columna se puede notar que la mayoría de los datos se encuentran bastante alejados de la línea de regresión, igualmente se puede notar que el error absoluto medio es bastante elevado (esto se cumple tanto en train como en test). Esto nos puede hacer concluir que el modelo cuenta con un sesgo bastante elevado.

Para el sesgo se pueden comparar las dos gráficas de la segunda columna. Se puede ver que las dos tienen un error pero este es bastante parecido tanto en test como en train.

Con lo que se ha mencionado se puede concluir que el modelo está teniendo un underfitting (sesgo alto y varianza baja). Que este sea el caso invalida métodos de regularización como Lasso o Ridge ya que estos se deben usar cuando existe overfitting. Para aumentar el sesgo se debe aumentar la complejidad del modelo, por esta razón se optó por agregar una variable independiente extra.

Segundo modelo

La variable independiente extra que se le agregó al modelo fue la de edad (desde ahora x2). Se realizaron los mismos métodos que en el modelo anterior:

```
#Linear regression with two independent variables
x_train, x_test, y_train, y_test = train_test_split(df[["x1", "x2"]], df["y"], test_size= 0.2, random_state=0)
mult_reg = LinearRegression().fit(x_train, y_train)

y_mult_train_pred = mult_reg.predict(x_train)
y_mult_test_pred = mult_reg.predict(x_test)

mse_mult_train = mean_absolute_error(y_train, y_mult_train_pred)
mse_mult_test = mean_absolute_error(y_test, y_mult_test_pred)

print(f"mse_mult_train: {mse_mult_train}, mse_mult_test: {mse_mult_test}")
```

En esta ocasión no se pueden hacer gráficas como las que se realizaron en la sección de arriba por lo que nos basaremos únicamente en el error:

```
mse_mult_train: 9009.913723184343, mse_mult_test: 8967.270539213538
```

Se puede ver que sigue teniendo un error bastante elevado (sesgo elevado) pero lo hemos mejorado, igualmente se puede ver que sigue teniendo una baja varianza al tener los dos errores magnitudes parecidas. Por estas razones se puede prever lograr un buen fit (sesgo bajo y magnitud baja) si se sigue este camino.