



# Tecnológico de Monterrey

## Actividad 4 (Regresión Lineal y Logística)

Luis Gerardo Ramírez Lastra A01733093

27 de noviembre del 2022

Desarrollo de Proyectos de Análisis de datos

Profesores:

Alfredo García Suárez

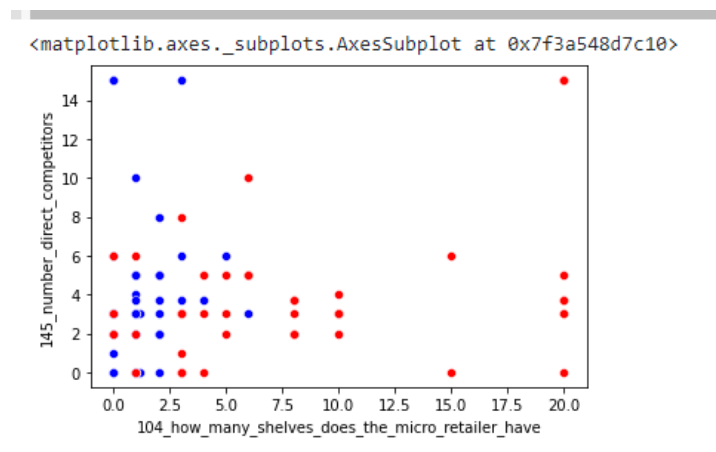
Andrés Esteban Acero López

Para dar solución a la actividad 4, aplicamos lo aprendido durante clase usando lo que es la regresión lineal y logística, para así poder predecir lo que podría pasar en el futuro con algunas variables dependientes, esto al observar cómo es su correlación con otras variables que ayuden a predecir algo. Para esta actividad utilizamos los datos recabados durante las últimas semanas, a los cuales les aplicamos regresiones lineales o logísticas dependiendo el caso.

## - Regresión lineal

1. Para el primer caso de regresión lineal, utilizamos como variable independiente el número de estantes y el número de productos perecederos que tiene una tienda de conveniencia, es importante recalcar que se utilizó la función de filtro para así tomar todos los datos que estén relacionados al tipo de tienda de conveniencia o comestibles, y como variable dependiente tomamos el número o la cantidad de competencia que rodea a esta microempresa.

A continuación se muestra una gráfica de dispersión que muestra cómo se ven las variables independientes elegidas en relación con la variable dependiente.



Se realizó un análisis de regresión lineal para así conocer cuál era la correlación que guardaban cada una de las variables tanto dependientes e independientes.

Gracias al código se obtuvieron los coeficientes de cada una de las variables independientes, pero para este caso lo que más nos importa es conocer el coeficiente de determinación que nos dirá qué tan confiable es el modelo creado y su correlación. El coeficiente de determinación es de 0.1493 y su correlación es igual a 0.3864; siendo este un modelo poco confiable

```
✓ [112] { 'fit_intercept': True,
  'normalize': 'deprecated',
  'copy_X': True,
  'n_jobs': None,
  'positive': False,
  'feature_names_in_': array(['104_how_many_shelves_does_the_micro_retailer_have',
                              '163_number_high_perishable_products'], dtype=object),
  'n_features_in_': 2,
  'coef_': array([0.24942108, 0.199453  ]),
  '_residues': 527.8557673250258,
  'rank_': 2,
  'singular_': array([46.91463951,  9.80241123]),
  'intercept_': 1.809260039812916}
```

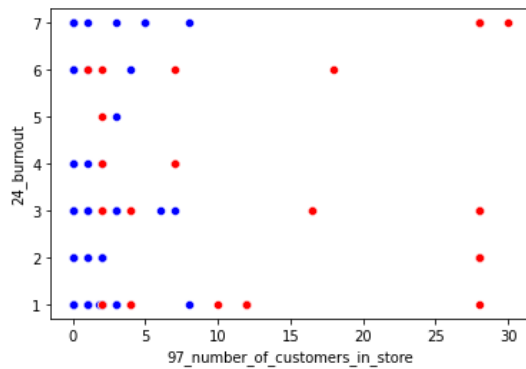
```
✓ [113] #Evaluamos la eficiencia del modelo obtenido por medio del coeficiente R2 Determinación
      coef_Deter=model.score(Vars_Indep,Var_Dep)
```

```
✓ [114] #Corroboramos cual es el coeficiente de Correlación de nuestro modelo
      coef_Correl=np.sqrt(coef_Deter)
      coef_Correl

0.38804377018657465
```

2. Para el segundo caso se realizó el análisis de regresión del número de clientes y el número de productos repartidos a las casas (variables independientes), para así determinar qué porcentaje es que el trabajador de una tienda se siente de algún modo cansado. En la siguiente gráfica se muestra la dispersión de las variables independientes con respecto a la dependiente (cantidad de cansancio).

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3a549d53d0>



```
{'fit_intercept': True,
'normalize': 'deprecated',
'copy_X': True,
'n_jobs': None,
'positive': False,
'feature_names_in_': array(['97_number_of_customers_in_store', '317_home_deliveries'],
dtype=object),
'n_features_in_': 2,
'coef_': array([0.08646894, 0.05834077]),
'_residues': 261.2358263895812,
'rank_': 2,
'singular_': array([78.97741253, 15.22107013]),
'intercept_': 1.903515067241456}
```

Con ayuda del modelo de regresión lineal se obtuvieron los coeficientes de cada una de nuestras variables y de igual forma se logró obtener el coeficiente de determinación igual a 0.08333, y así sabemos que su coeficiente de correlación es igual a 0.2887, por lo que podemos concluir que la correlación que tienen es muy bajo, siendo este un modelo poco confiable

```
✓ ▶ #Evaluamos la eficiencia del modelo obtenido por medio del coeficiente R2 Determinación
coef_Deter=model.score(Vars_Indep,Var_Dep)
coef_Deter
```

```
0.08332945883531984
```

```
✓ [120] #Corroboramos cual es el coeficiente de Correlación de nuestro modelo
coef_Correl=np.sqrt(coef_Deter)
coef_Correl
```


```
0.2886684236893946
```

3. Para el último análisis se tomó como variable independiente al número de estantes y al número de refrigeradores dentro de una tienda de conveniencia, esto para saber cómo estas afectan al número de clientes que se encuentran dentro de la tienda. En la siguiente gráfica se muestra la dispersión de datos de las variables elegidas.



Se realizó de igual forma, el modelo de regresión lineal el cual nos dio los coeficientes de las variables independientes y por consiguiente el valor de su coeficiente de determinación igual a 0.17 con un coeficiente de correlación de 0.4123.

```
{'fit_intercept': True,
 'normalize': 'deprecated',
 'copy_X': True,
 'n_jobs': None,
 'positive': False,
 'feature_names_in_': array(['104_how_many_shelves_does_the_micro_retailer_have',
 '268_number_fridges'], dtype=object),
 'n_features_in_': 2,
 'coef_': array([-0.08095999,  0.9007377 ]),
 '_residues': 193.12057730402466,
 'rank_': 2,
 'singular_': array([9.82731606,  6.95422119]),
 'intercept_': 0.466230951870491}
```

✓ 0s  #Evaluamos la eficiencia del modelo obtenido por medio del coeficiente R2 Determinación

```
coef_Deter=model.score(Vars_Indep,Var_Dep)
coef_Deter
```

0.17001405093694888

✓ [124] 0s #Corroboramos cual es el coeficiente de Correlación de nuestro modelo

```
coef_Correl=np.sqrt(coef_Deter)
coef_Correl
```

0.41232760147357206

- Tabla comparativa

Modelo	Variables independientes	Variable dependiente	Coeficiente de determinación	Coeficiente de correlación
1	Número de estantes Número de productos perecederos	Competencia alrededor del área	0.1493	0.3864
2	Número de clientes Número de productos repartidos	Cansancio en los trabajadores	0.0833	0.2887
3	Número de estantes Número de refrigeradores	Número de clientes	0.17	0.4123

Dentro de la tabla comparativa, se muestra cómo es que el cambiar a las variables independientes e independientes hará que el coeficiente de determinación y correlación cambie haciendo que esté aumente o disminuya. Para nuestro caso, es importante procurar hacer que estos coeficientes sean lo mayor posible y que el coeficiente de correlación tienda a 1 para que el modelo creado sea confiable.

## - Regresión logística

Una regresión logística nos ayuda a poder predecir con valores numéricos algún dato de tipo binario, es decir que sus resultados solo tengan la respuesta sí o no. Para poder realizar este análisis se deberán seleccionar variables independientes de tipo numérica y una variable dependiente que tenga strings sí y no.

1. Para el primer modelo de regresión logística se utiliza como variables independientes al número de refrigeradores y número de estantes con los que cuenta una tienda de conveniencia, y como variable dependiente se definiría si la tienda vende o no productos frescos. A continuación se muestra todo el código que se realizó, para así dividir los datos, escalar los y hasta entrenarlos para así lograr predecir algunos valores en términos de sí o no

```
✓ 0s #Declaramos las variables dependientes e independientes para la regresión Logística
Vars_Indep= filtros[['268_number_fridges', '104_how_many_shelves_does_the_micro_retailer_have']]
Var_Dep= filtros['102_does_the_micro_retailer_sells_fresh_products_']

#Redefinimos las variables
X= Vars_Indep
y= Var_Dep

#Dividimos el conjunto de datos en la parte de entrenamiento y prueba:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=None)

#Se escalan todos los datos
escalar = StandardScaler()

#Para realizar el escalamiento de las variables "X" tanto de entrenamiento como de prueba, utilizaremos
X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

#Definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo = LogisticRegression()
```

```
#Entrenamos el modelo
algoritmo.fit(X_train, y_train)

#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

array(['no', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
       'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes'],
      dtype=object)
```

Para saber si el modelo realizado es bueno, se procedió a obtener la precisión, exactitud y sensibilidad del modelo, y así conocer qué tan bueno es prediciendo los valores entre sí o no. Teniendo un valor de precisión igual a 0.9375, una exactitud de 0.888 y una sensibilidad del 0.9375.

```
#Cálculo de precisión, exactitud y sensibilidad

#Calculo la precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred, average="binary", pos_label="yes")
print('Precisión del modelo:')
print(precision)

#Calculo la exactitud del modelo
from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)

#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="yes")
print('Sensibilidad del modelo:')
print(sensibilidad)
```



```
Precisión del modelo:
0.9375
Exactitud del modelo:
0.8888888888888888
Sensibilidad del modelo:
0.9375
```

2. Para el segundo modelo de regresión logística, se tomó cómo variable independiente el número de clientes en la tienda y el número de empleados que se han tenido en el último año, cómo variable dependiente se tomó si es que dentro de la tienda hay o no una conexión a internet dependiendo de la cantidad de personas que están dentro de la misma. De igual forma, se utilizó el mismo código, únicamente cambiando las variables a analizar para así poder tener un modelo que nos pueda decir si hay o no una conexión a internet dentro del establecimiento.

[illegible]

Para saber si este modelo de regresión logística logra ser confiable se debe calcular la precisión, exactitud y sensibilidad del modelo creado. Para este caso tenemos que la exactitud = 0.66, la precisión = 0.7058 con una sensibilidad igual a 0.9231.

```
#Cálculo de precisión, exactitud y sensibilidad

#Calculo la precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred, average="binary", pos_label="yes")
print('Precisión del modelo:')
print(precision)

#Calculo la exactitud del modelo
from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)

#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="yes")
print('Sensibilidad del modelo:')
print(sensibilidad)
```

```
Precisión del modelo:
0.7058823529411765
Exactitud del modelo:
0.6666666666666666
Sensibilidad del modelo:
0.9230769230769231
```

3. Para este último modelo de regresión logística se utiliza como variable independiente al número de clientes dentro de la tienda y a la cantidad de productos perecederos que esta misma debe de vender, con ayuda de estas variables se tendrá que definir si es que el trabajador se siente cansado o no.

```
#Declaramos las variables dependientes e independientes para la regresión Logística
Vars_Indep= filtros[['97_number_of_customers_in_store', '163_number_high_perishable_products']]
Var_Dep= filtros['310_burnout']

#Redefinimos las variables
X= Vars_Indep
y= Var_Dep

#Dividimos el conjunto de datos en la parte de entrenamiento y prueba:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state =None)

#Se escalan todos los datos
escalar = StandardScaler()

#Para realizar el escalamiento de las variables "X" tanto de entrenamiento como de prueba, utilizaremos fit_transform
X_train = escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

#Definimos el algoritmo a utilizar
from sklearn.linear_model import LogisticRegression
algoritmo = LogisticRegression()

#Entrenamos el modelo
algoritmo.fit(X_train, y_train)

#Realizamos una predicción
y_pred = algoritmo.predict(X_test)
y_pred

array(['no', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no',
       'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes'],
      dtype=object)
```

Al igual que en las regresiones pasadas, se realizó el mismo código para así determinar qué tan bueno es la regresión logística y que tan bien predice el valor 'sí o no'. Sin dejar de lado el querer calcular la precisión, exactitud y sensibilidad del modelo. teniendo entonces para este modelo una precisión = 0.4285, una exactitud de 0.444 con una sensibilidad = 0.75.

```
#Cálculo de precisión, exactitud y sensibilidad

#Calculo la precisión del modelo
from sklearn.metrics import precision_score

precision = precision_score(y_test, y_pred, average="binary", pos_label="yes")
print('Precisión del modelo:')
print(precision)

#Calculo la exactitud del modelo
from sklearn.metrics import accuracy_score

exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)

#Calculo la sensibilidad del modelo
from sklearn.metrics import recall_score

sensibilidad = recall_score(y_test, y_pred, average="binary", pos_label="yes")
print('Sensibilidad del modelo:')
print(sensibilidad)
```

```
✖ Precisión del modelo:
0.42857142857142855
Exactitud del modelo:
0.4444444444444444
Sensibilidad del modelo:
0.75
```

#### - Tabla comparativa

En la tabla, se muestra una comparación entre todos los modelos que se realizaron en la regresión logística, mostrando así sus variables independientes e independientes y junto a estos la precisión, exactitud y sensibilidad con la que cuentan. Teniendo todos estos datos en una misma tabla, podemos observar quién tiene una mejor precisión al predecir la variable dependiente seleccionada.

Modelo	Variables independientes	Variable dependiente	Precisión	Exactitud	Sensibilidad
1	Número de estantes Número de refrigeradores	Venta de productos frescos	0.9375	0.8888	0.9375

2	Número de clientes Número de empleados	Conexión a internet	0.7059	0.6667	0.9239
3	Número de clientes Número de productos perecederos	Cansancio en los trabajadores	0.4285	0.4444	0.75