Minería de datos G.571

Feature selection: Correlation coefficient y Fisher’s Score

Minería de Datos

Zavala Roman Irvin Eduardo 1270771

Correlation coefficient/Pearson's Correlation coefficient

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[n\Sigma x^2 - (\Sigma x)^2][n\Sigma y^2 - (\Sigma y)^2]}}$$

r = Pearson Coefficient

n= number of the pairs of the stock

$\sum xy$ = sum of products of the paired stocks

$\sum x$ = sum of the x scores

$\sum y$= sum of the y scores

$\sum x^2$ = sum of the squared x scores

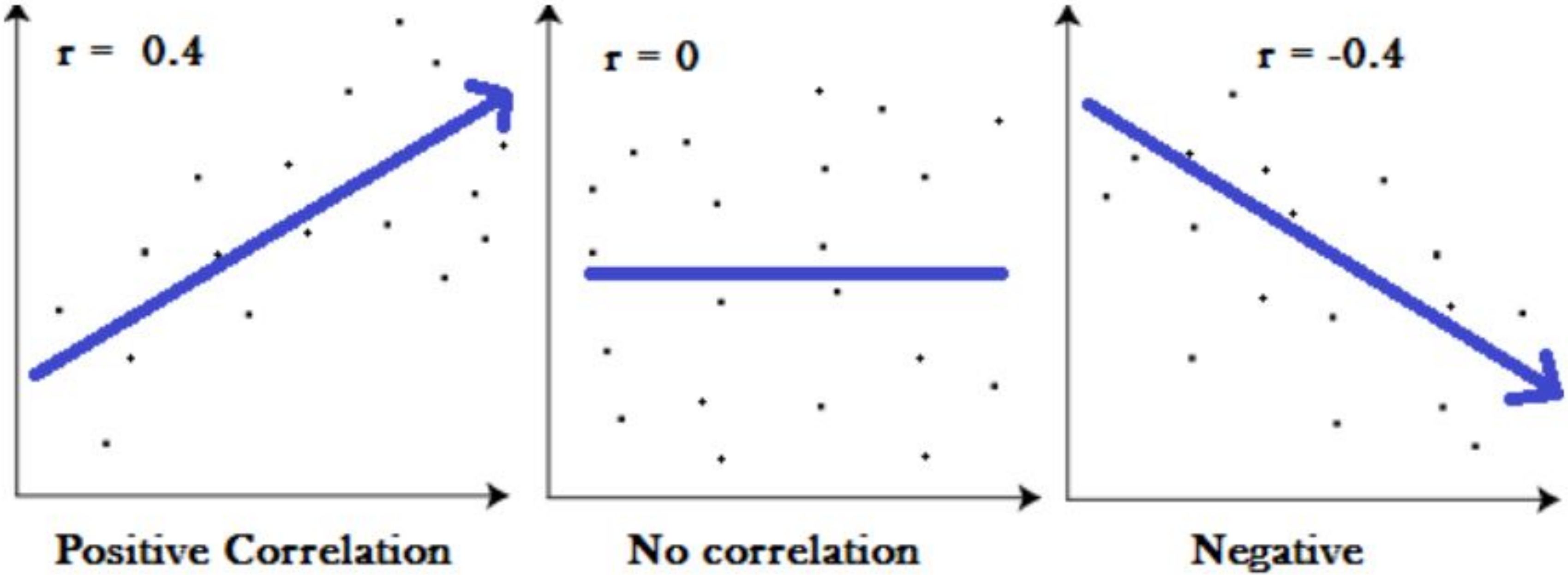
$\sum y^2$ = sum of the squared y scores

Correlation coefficient/Pearson's Correlation coefficient

	x	y	xy	x^2	y^2
	6	12	72	36	144
	8	10	80	64	100
	10	20	200	100	400
Σ	*24*	*42*	*352*	*200*	*644*

$$r = \frac{(3)(352) - (24)(42)}{\sqrt{(3 * 200 - (24)^2)(3 * 644 - (42)^2)}} = 0.7559$$

Correlation coefficient/Pearson's Correlation coefficient



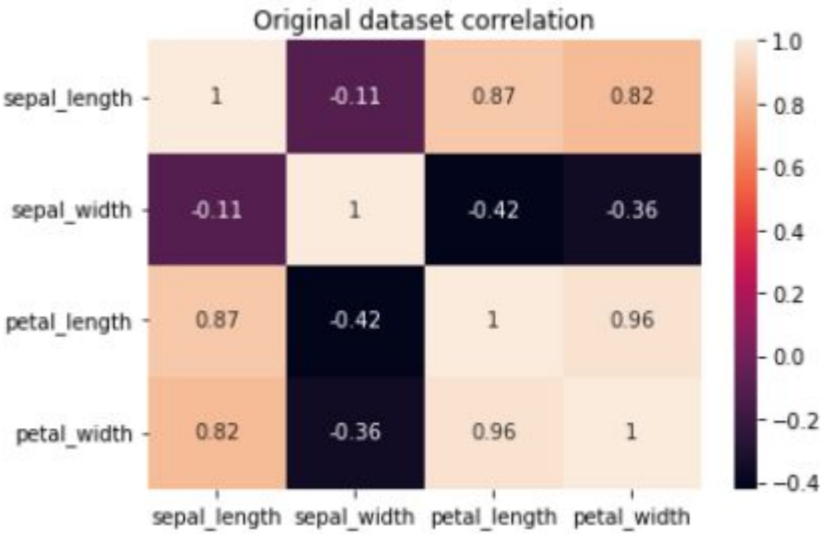
Fuente:
https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/

Correlation coefficient/Pearson's Correlation coefficient

Entonces, ¿cuál es la idea?

Buscar atributos altamente correlacionados en el dataset y si esto sucede se puede eliminar uno o varios ya que los consideramos “redundantes”.

Prueba Iris




```
ax = plt.axes()
sn.heatmap(corr_matrix,annot = True)
ax.set_title('Original dataset correlation')
plt.show()
```

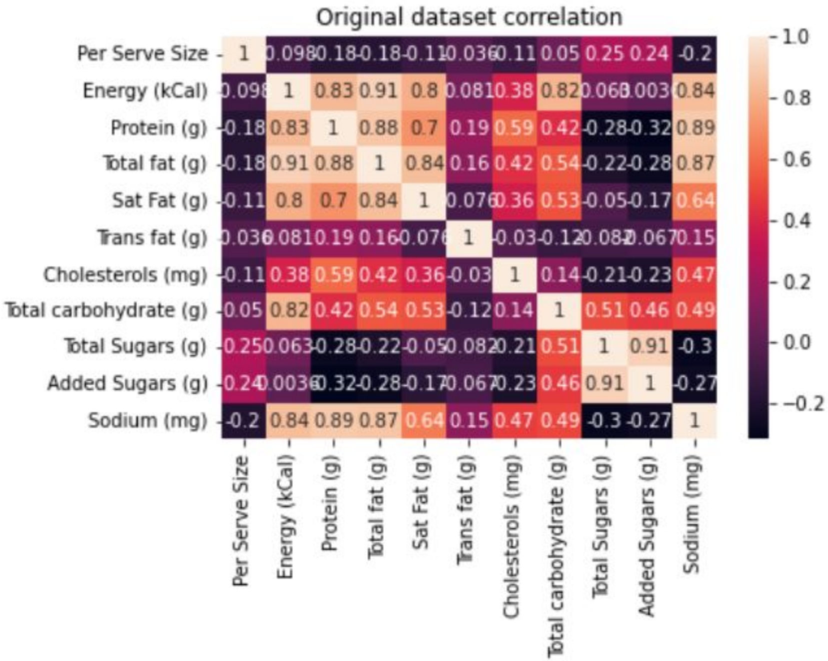
```python
#Algoritmo corr coef para eliminar atributos redundantes
def corr_coef(dataset, threshold):
    print(dataset)
    col_corr = set()
    corr_matrix = dataset.corr()
    ax = plt.axes()
    sn.heatmap(corr_matrix,annot = True)
    ax.set_title('Original dataset correlation')
    plt.show()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

```python
dataset = pd.read_csv('iris.csv')
label = 'species'
features = dataset.drop(label, axis = 1)
target = dataset[[label]].apply(lambda x: pd.factorize(x)[0])
corr_matrix = features.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))
to_drop = [column for column in upper.columns if any(upper[column] > 0.85)]
print(to_drop)
```

Resultado

```
____________DATASET IRIS____________
Metodo 1 decide eliminar: {'petal_length', 'petal_width'}
Metodo 2 decide eliminar: ['petal_length', 'petal_width']
Final dataset:
     sepal_length  sepal_width
0             5.1          3.5
1             4.9          3.0
2             4.7          3.2
3             4.6          3.1
4             5.0          3.6
..            ...          ...
145           6.7          3.0
146           6.3          2.5
147           6.5          3.0
148           6.2          3.4
149           5.9          3.0

[150 rows x 2 columns]
```

Correlation coefficient con dataset Starbucks




```
______________DATASET STARBUCKS_______________
Metodo 1 decide eliminar: {'Added Sugars (g)', 'Sodium (mg)', 'Total fat (g)'}
Metodo 2 decide eliminar: ['Total fat (g)', 'Added Sugars (g)', 'Sodium (mg)']
Final dataset:
                Menu Items  ...  Total Sugars (g)
0           McVeggie™ Burger  ...              7.90
1       McAloo Tikki Burger®  ...              7.05
2      McSpicy™ Paneer Burger  ...              8.35
3          Spicy Paneer Wrap  ...              3.50
4         American Veg Burger  ...              7.85
..                        ...  ...               ...
136    Tomato Ketchup Sachets  ...              2.33
137               Maple Syrup  ...             16.20
138              Cheese Slice  ...              0.54
139                Sweet Corn  ...              2.54
140      Mixed Fruit Beverage  ...             16.83

[141 rows x 9 columns]
```

Fisher's Score

Este método se basa en una función matemática que da como resultado un puntaje por atributo dando como resultado un ranking de todos los atributos.

$$F(\mathbf{x}^j) = \frac{\sum_{k=1}^{c} n_k(\mu_k^j - \mu^j)^2}{(\sigma^j)^2}$$

n_k Es el tamaño de la clase k-th

μ_k^j Media de la clase k-th respecto al feature j-th

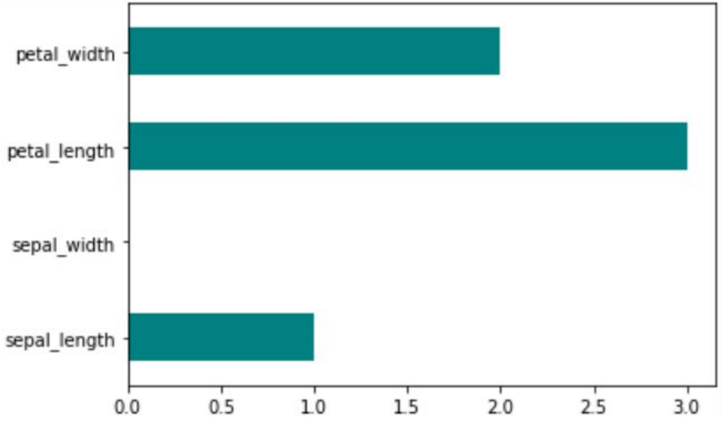
μ^j Media de la feature j-th

σ^j Desviación estándar de la feature j-th

Esta no es la mejor versión de este método,
pero es de los más óptimos computacionalmente.

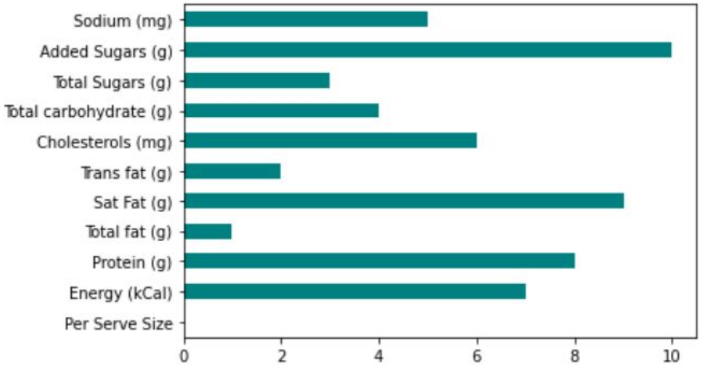
Fisher's Score prueba Iris

```
from skfeature.function.similarity_based import fisher_score
pd.options.mode.chained_assignment = None  # default='warn'

dataset = pd.read csv('iris.csv')
array = dataset.values
X = array[:,0:4]
Y = array[:,4]
score = fisher score.fisher score(X,Y)
feat importances = pd.Series(score, dataset.columns[0:len(dataset.columns)-1])
feat_importances.plot(kind = 'barh', color = 'teal')
plt.show()
```



Fisher’s Score prueba Starbucks

```
from skfeature.function.similarity_based import fisher_score
pd.options.mode.chained_assignment = None  # default='warn'

dataset = pd.read_csv('India_Menu.csv')
dataset1 = dataset.replace('[^0-9]+','', regex=True)
array = dataset.values
array1 = dataset1.values
X = array1[:,2:len(dataset.columns)].astype(float)
Y = array[:,0]
score = fisher_score.fisher_score(X,Y)
feat_importances = pd.Series(score, dataset.columns[2:len(dataset.columns)])
feat_importances.plot(kind = 'barh', color = 'teal')
plt.show()
```



¿Cuáles seleccionar?

Aunque se deben escoger los valores más altos y quitar los más bajos, ninguna fuente dice hasta qué punto recortar o salvar atributos. Por lo que queda a criterio del analizador mantener los primeros *m* lugares del ranking. En este caso decidí agarrar el top 50%.

```python
def fisher_score_selection(X, target, X_cols, dataset, porcentaje):
    score = fisher_score.fisher_score(X,Y)
    feat_importances = pd.Series(score, X_cols)

    feat_importances.plot(kind = 'barh', color = 'teal')
    plt.show()

    top = X.shape[1] * porcentaje//100
    feat_importances = feat_importances.sort_values(ascending=False)
    feat_importances.drop(feat_importances.tail(top).index,inplace=True)
    feat_importances = feat_importances.index.tolist()
    print("Selected: ", feat_importances)
    dataset = dataset[dataset.columns.intersection(feat_importances)]
    return dataset
dataset = pd.read_csv('iris.csv')
array = dataset.values
X = array[:,0:4]
X_cols = dataset.columns[0:4]
Y = array[:,4]
dataset = fisher_score_selection(X, Y, X_cols, dataset, 50)
print(dataset)
```

```
Selected:  ['petal_length', 'petal_width']
     petal_length  petal_width
0             1.4          0.2
1             1.4          0.2
2             1.3          0.2
3             1.5          0.2
4             1.4          0.2
..            ...          ...
145           5.2          2.3
146           5.0          1.9
147           5.2          2.0
148           5.4          2.3
149           5.1          1.8
```

```
dataset = pd.read_csv('India_Menu.csv')
dataset1 = dataset.replace('[^0-9]+','', regex=True)
array = dataset.values
array1 = dataset1.values
X = array1[:,2:len(dataset.columns)].astype(float)
X_cols = dataset.columns[2:len(dataset.columns)]
Y = array[:,0]
dataset = fisher_score_selection(X, Y, X_cols, dataset, 50)
print(dataset)
```

```
Selected:  ['Added Sugars (g)', 'Sat Fat (g)', 'Protein (g)', 'Energy (kCal)', 'Cholesterols (mg)', 'Sodium (mg)']
     Energy (kCal)  Protein (g)  ...  Added Sugars (g)  Sodium (mg)
0           402.05        10.24  ...              4.49       706.13
1           339.52         8.50  ...              4.07       545.34
2           652.76        20.29  ...              5.27      1074.58
3           674.68        20.96  ...              1.08      1087.46
4           512.17        15.30  ...              4.76      1051.24
..             ...          ...  ...               ...          ...
136          11.23         0.08  ...              1.64        71.05
137          86.40         0.00  ...              5.34        15.00
138          51.03         3.06  ...              0.00       178.95
139          45.08         1.47  ...              0.00         0.04
140          72.25         0.65  ...              0.00        10.80

[141 rows x 6 columns]
```

Bibliografía

Correlation coefficient: Simple definition, formula, easy steps. (2021, mayo 24). Statistics How To. https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/

Gu, Q., Li, Z., & Han, J. (2012). Generalized Fisher score for feature selection. En *arXiv [cs.LG]*. http://arxiv.org/abs/1202.3725

How to drop out highly correlated features in Python? (s/f). ProjectPro. https://www.projectpro.io/recipes/drop-out-highly-correlated-features-in-python

Naik, K. C. (s/f). *2-Feature Selection- Correlation.ipynb at master · krishnaik06/Complete-Feature-Selection*.

Singh, R. (2021, febrero 16). *Implementing Feature Selection Methods for Machine learning*. Medium. https://ranasinghiitkgp.medium.com/implementing-feature-selection-methods-for-machine-learning-bfa2e4b4e02

Thakur, M. (2019, septiembre 18). *Pearson correlation coefficient*. WallStreetMojo. https://www.wallstreetmojo.com/pearson-correlation-coefficient/