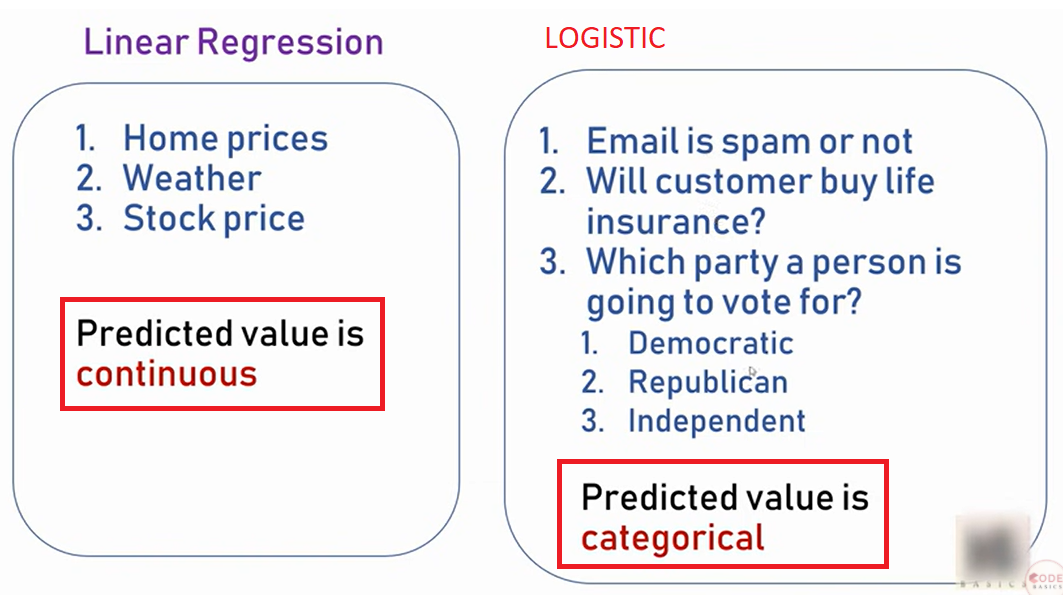
https://www.youtube.com/watch?v=zM4VZR0px8E&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw&index=8



Dataset is downloaded from Kaggle. Link: <https://www.kaggle.com/giripujar/hr-analytics>

import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv("HR\_comma\_sep.csv")

df.head()

**REGRESION LOGISTICA: OUTPUT DISCRETO, EN ESTE CASO SERA BINARIO**

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **Left** | **promotion\_last\_5years** | **Department** | **Salary** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | sales | Low |
| **1** | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | sales | Médium |
| **2** | 0.11 | 0.88 | 7 | 272 | 4 | 0 | 1 | 0 | sales | Médium |
| **3** | 0.72 | 0.87 | 5 | 223 | 5 | 0 | 1 | 0 | sales | Low |
| **4** | 0.37 | 0.52 | 2 | 159 | 3 | 0 | 1 | 0 | sales | Low |

**Data exploration and visualization**

Split data

left=1 employees leaving the company

left=0 employees retained

left **=** df[df**.**left**==**1]

retained **=** df[df**.**left**==**0]

**Average numbers for all columns**

df**.**groupby('left')**.**mean()

|  | **satisfaction\_level** | **last\_evaluation** | **number\_project** | **average\_montly\_hours** | **time\_spend\_company** | **Work\_accident** | **promotion\_last\_5years** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **left** |  |  |  |  |  |  |  |
| **0** | 0.666810 | 0.715473 | 3.786664 | 199.060203 | 3.380032 | 0.175009 | 0.026251 |
| **1** | 0.440098 | 0.718113 | 3.855503 | 207.419210 | 3.876505 | 0.047326 | 0.005321 |

Left = 0 employees retained

Left = 1 employees leaving

From above table we can draw following conclusions,

1. \*\*Satisfaction Level\*\*: Satisfaction level seems to be relatively low (0.44) in employees leaving the firm vs the retained ones (0.66)
2. \*\*Average Monthly Hours\*\*: Average monthly hours are higher in employees leaving the firm (199 vs 207)
3. \*\*Promotion Last 5 Years\*\*: Employees who are given promotion are likely to be retained at firm

|  |  |
| --- | --- |
| **Impact of salary on employee retention**  pd**.**crosstab(df**.**salary,df**.**left)**.**plot(kind**=**'bar') | **Department wise employee retention rate**  pd**.**crosstab(df**.**Department,df**.**left)**.**plot(kind**=**'bar') |
|  |  |
| **Show: High salaries are likely to stay** | **Some departemts retention is better** |

**From the data analysis so far we can conclude that we will use following variables as independant variables in our model**

1. \*\*Satisfaction Level\*\*
2. \*\*Average Monthly Hours\*\*
3. \*\*Promotion Last 5 Years\*\*
4. \*\*Salary\*\*

subdf **=** df[['satisfaction\_level','average\_montly\_hours','promotion\_last\_5years','salary']]

**Tackle salary dummy variable**

Salary has all text data. It needs to be converted to numbers and we will use dummy variable for that. Check my one hot encoding tutorial to understand purpose behind dummy variables.

salary\_dummies **=** pd**.**get\_dummies(subdf**.**salary, prefix**=**"salary")

df\_with\_dummies **=** pd**.**concat([subdf,salary\_dummies],axis**=**'columns')

df\_with\_dummies**.**head()

|  | **satisfaction\_level** | **average\_montly\_hours** | **promotion\_last\_5years** | **Salary** | **salary\_high** | **salary\_low** | **salary\_medium** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 157 | 0 | Low | 0 | 1 | 0 |
| **1** | 0.80 | 262 | 0 | Médium | 0 | 0 | 1 |
| **2** | 0.11 | 272 | 0 | Médium | 0 | 0 | 1 |
| **3** | 0.72 | 223 | 0 | Low | 0 | 1 | 0 |
| **4** | 0.37 | 159 | 0 | Low | 0 | 1 | 0 |

Now we need to remove salary column which is text data. It is already replaced by dummy variables so we can safely remove it

df\_with\_dummies**.**drop('salary',axis**=**'columns',inplace**=True**)

df\_with\_dummies**.**head()

|  | **satisfaction\_level** | **average\_montly\_hours** | **promotion\_last\_5years** | **salary\_high** | **salary\_low** | **salary\_medium** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.38 | 157 | 0 | 0 | 1 | 0 |
| **1** | 0.80 | 262 | 0 | 0 | 0 | 1 |
| **2** | 0.11 | 272 | 0 | 0 | 0 | 1 |
| **3** | 0.72 | 223 | 0 | 0 | 1 | 0 |
| **4** | 0.37 | 159 | 0 | 0 | 1 | 0 |

X **=** df\_with\_dummies

y **=** df**.**left

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,train\_size**=**0.3)

**from** sklearn.linear\_model **import** LogisticRegression

model **=** LogisticRegression()

model**.**fit(X\_train, y\_train)

model**.**predict(X\_test)

Out[89]:

array([0, 0, 0, ..., 0, 0, 1], dtype=int64)

**Accuracy of the model**

model**.**score(X\_test,y\_test)

0.78428571428571425

**SE PUEDE SIMPLIFICAR**

**X =** dfle**.drop(‘salary\_high**, inplace=True, axis=1)

X **=** df\_with\_dummies.drop

y **=** df**.**left

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X,y,train\_size**=**0.3)

**from** sklearn.linear\_model **import** LogisticRegression

model **=** LogisticRegression()

model**.**fit(X\_train, y\_train)

model**.**predict(X\_test)

Out[89]:

array([0, 0, 0, ..., 0, 0, 1], dtype=int64)

**Accuracy of the model**

model**.**score(X\_test,y\_test)

0.78428571428571425

= = = = =

import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv("HR\_comma\_sep.csv")

#data with employees leaving

left = df[df.left==1]

subdf = df[['satisfaction\_level','average\_montly\_hours','promotion\_last\_5years','salary']]

salary\_dummies = pd.get\_dummies(subdf.salary, prefix="salary")

df\_with\_dummies = pd.concat([subdf,salary\_dummies],axis='columns')

df\_with\_dummies.drop('salary',axis='columns',inplace=True)

X = df\_with\_dummies

y = df.left

left=1 employees leaving the company

left=0 employees retained

‘y’ tiene los valores left 0,1

from sklearn.model\_selection import train\_test\_split

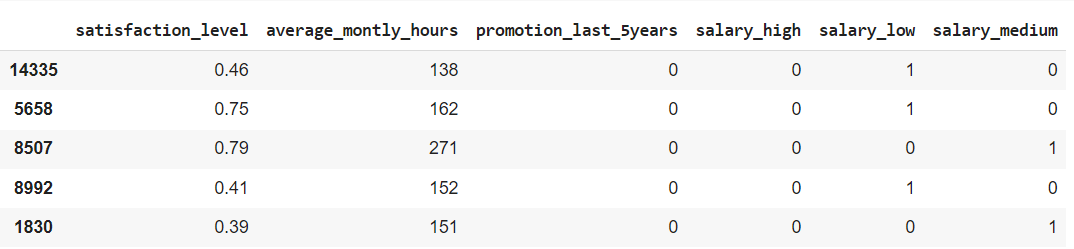
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.3)

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X\_train, y\_train)

X\_test



model.predict(X\_test)

ouput

array([0, 0, 0, ..., 0, 1, 0])

Lo que pronostica es que la mayoría se mantendrá, (0) y el que se ira es (1)

test = pd.DataFrame([[0.64, 188, 0,0,1,0]])

model.predict(test)

Si da ‘0’ se queda, si da ‘1’ se ira.

LA DATA PARA PROBAR ES QUE “Y” TARGET, ES 0,1 PERMANECERA O SE IRA

Y LA DATA ‘X’ ES LOS VALORES DE SATISFACCION, MONTHLY HOURS, PROMOTION, SALARY HIGH, LOW, MEDIUM

**ELIMINEMOS SALARY HIGH Y VERIFIQUEMOS EXACTITUD**

import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline

df = pd.read\_csv("HR\_comma\_sep.csv")

subdf = df[['satisfaction\_level','average\_montly\_hours','promotion\_last\_5years','salary']]

salary\_dummies = pd.get\_dummies(subdf.salary, prefix="salary")

df\_with\_dummies = pd.concat([subdf,salary\_dummies],axis='columns')

df\_with\_dummies.drop('salary',axis='columns',inplace=True)

X = df\_with\_dummies

X**.**drop('salary\_high',axis**=**'columns',inplace**=True**)

y = df.left #solo campo left

left=1 employees leaving the company

left=0 employees retained

‘y’ tiene los valores left 0,1

from sklearn.model\_selection import train\_test\_split

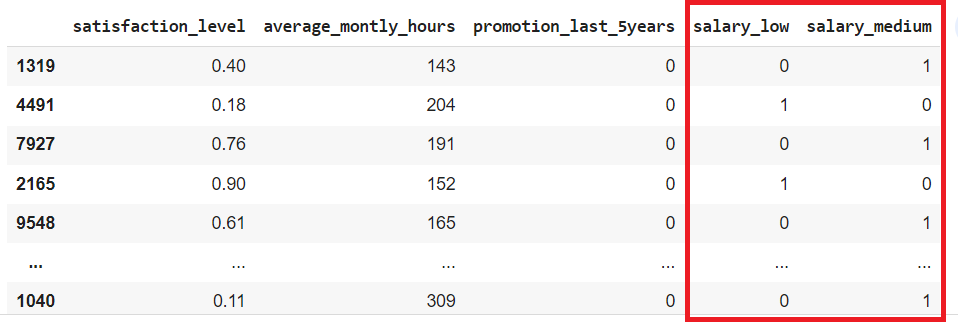
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.3)

from sklearn.linear\_model import LogisticRegression

model = LogisticRegression()

model.fit(X\_train, y\_train)

X\_test



model.predict(X\_test)

ouput

array([0, 0, 0, ..., 0, 1, 0])

model**.**score(X\_test,y\_test)

0.78019

Bajo de 0.784 a 0.780

Lo que pronostica es que la mayoría se mantendrá, (0) y el que se ira es (1)

test = pd.DataFrame([[0.64, 188, 0,1,0]])

model.predict(test)

Si da ‘0’ se queda, si da ‘1’ se ira.

LA DATA PARA PROBAR ES QUE “Y” TARGET, ES 0,1 PERMANECERA O SE IRA

Y LA DATA ‘X’ ES LOS VALORES DE SATISFACCION, MONTHLY HOURS, PROMOTION, SALARY HIGH, LOW, MEDIUM