Alumno:	García González Aarón Antonio
Grupo:	3CV9
Unidad de Aprendizaje:	Procesamiento de lenguaje natural
Profesora:	Kolesnikova Olga
Tarea 3.	Algoritmo de aprendizaje automático por modelo de regresión lineal y logística con descenso gradiente mediante scikit-learn.
Fecha:	Martes 21 de abril de 2020

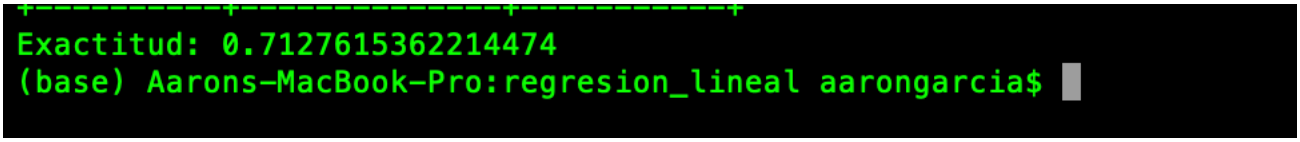
Algoritmo de aprendizaje automático por modelo de regresión lineal

Valores de predicción de algunos elementos del conjunto de pruebas

Real	Prediccion	Error
687500	349042	49.2302
671500	556826	17.0773
330000	558064	69.1102
500000	700749	40.1497
635000	1.26425e+06	99.095
380000	201910	46.8658
310000	618175	99.4114
735000	810668	10.2949
331000	354575	7.12238
627500	681364	8.58385
320000	211705	33.8422
198000	334445	68.9117
476500	416287	12.6364
375000	686336	83.0228
429950	419839	2.35175
725000	741268	2.2438
390000	475573	21.9419
185000	93120.7	49.6645
300000	198498	33.8339
286300	141224	50.6727
324950	300760	7.4443
699900	569113	18.6865
330000	460679	39.5996
215000	193002	10.2318
557500	506231	9.19622

760000	890264	17.14
199000	229200	15.1759
208000	125376	39.723
280005	220651	21.1974
1.7e+06	1.1655e+06	31.4412
395000	437815	10.8391
325000	316705	2.55219
750000	678921	9.47722
230000	217593	5.39432
845000	1.13272e+06	34.0492
765000	832689	8.84817
455000	359128	21.0708
825000	775144	6.04312
460000	465206	1.13172
170000	53903.6	68.292
674250	832676	23.4966
637800	642132	0.67919
402000	651926	62.1706
1e+06	472238	52.7762
436000	572976	31.4166
553000	360633	34.7862
825000	1.30798e+06	58.5427
700000	718845	2.69214
538000	547551	1.77532
489000	415202	15.0916
207000	277811	34.2084

Valor de error general del programa



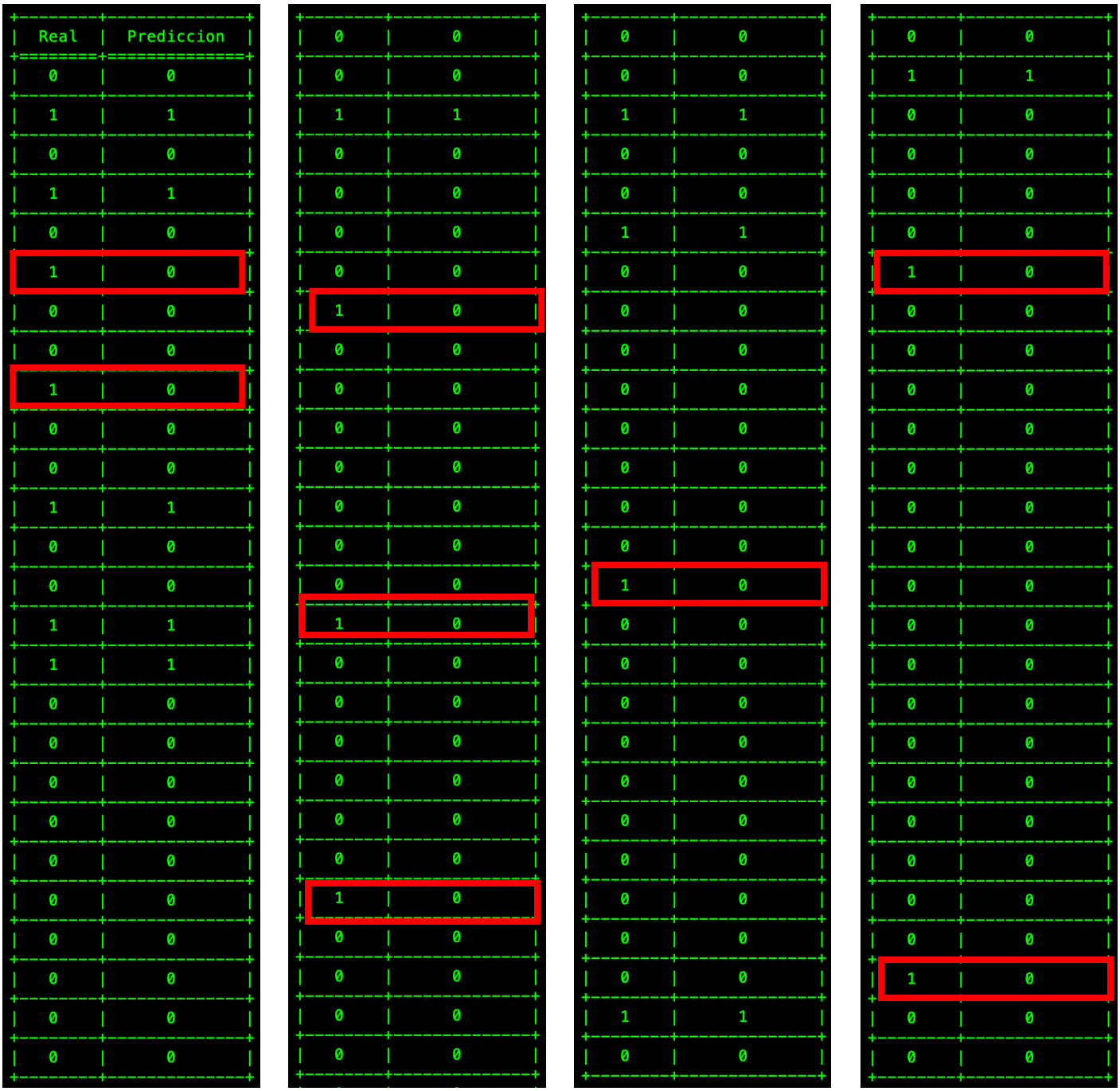
Código

```python
from datetime import datetime
import pandas as pd
import numpy as np
import random
import math
from tabulate import tabulate
from sklearn import datasets, linear_model, metrics
from sklearn import preprocessing
from sklearn.model_selection import train_test_split


# funcion que obtiene los datos a partir de un documento .csv
# el campo fecha no lo desprecia
def getData(name_file):
    global θs

    datos = pd.read_csv(name_file)

    matriz = np.array(datos)
    matriz = matriz[:,1:] # quitamos la columna id

    for f in matriz:
        f[0] = datetime.strptime(f[0], '%M/%d/%Y').date() # convertimos a fecha

    minimo = min(matriz[:, 0]) # fecha mas antigua

    for f in matriz:
        f[0] = int(abs(minimo - f[0]).days) # favorecemos en numero a las fechas mas cercanas

    random.shuffle(matriz) # revolvemos los datos

    Y = matriz[:, 1]
    X = matriz[:, [0]+[f for f in range(2,len(matriz[0]))]] # quitamos la columna de Y

    θs = np.array( [0]*(len(X[0])+1) )

    return X,Y


# funcion que a partir de la prediccion y el dato objetivo es decir el precio de las casas,
# imprime una tabla comparativa
def printPrediction(Y_predicciones, Y_test):
    tabla = []
    for i in range(0,len(Y_test)):
        eror_particular = abs(100-(100/(Y_test[i])*Y_predicciones[i]))
        tabla.append([Y_test[i],Y_predicciones[i],eror_particular])

    print(tabulate(tabla,headers=['Real','Prediccion', 'Error' ],tablefmt="grid", numalign="center"))


def main():
    name_file = "data.csv"
    X, Y = getData(name_file)
```

```
    X = preprocessing.scale(X) ## feature scalling

    X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.3)

    reg = linear_model.LinearRegression() # creamos el objeto de tipo regresion linear
    reg.fit(X_train, Y_train) # entrenamos el modelo

    Y_predicciones = reg.predict(X_test)

    printPrediction(Y_predicciones, Y_test)
    print('Exactitud: {}'.format(reg.score(X_test, Y_test)))


if __name__ == "__main__":
    main()
```

Algoritmo de aprendizaje automático por modelo de regresión logística

Valores de predicción de algunos elementos del conjunto de pruebas



Real	Prediccion
0	0
1	1
0	0
1	1
0	0
1	0
0	0
0	0
1	0
0	0
0	0
1	1
0	0
0	0
1	1
1	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

0	0
0	0
1	1
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0

0	0
0	0
1	1
0	0
0	0
0	0
0	0
1	1
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

0	0
1	1
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
0	0
0	0
0	0

Valor de error general del programa

```
0.9855072463768116
(base) Aarons-MacBook-Pro:regresion_logistica aarongarcia$
```

Código

```python
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_score


if __name__ == "__main__":

    df=pd.read_csv('SMSSpamCollection.txt',sep='\t',names=['Status','Message'])

    #print("Spam: ", len(df[df.Status=='spam']))
    #print("Ham: ", len(df[df.Status=='ham']))

    df.loc[df["Status"]=='ham',"Status",]=0
    df.loc[df["Status"]=='spam',"Status",]=1

    df_x=df["Message"]
    df_y=df["Status"]

    cv = TfidfVectorizer(min_df=1,stop_words='english') # escalar los datos

    df_x = cv.fit_transform(df_x).toarray()
    df_y = np.array(df_y.astype('int'))

    x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.3)

    logreg = LogisticRegression()
    logreg.fit(x_train,y_train)

    y_predicciones = logreg.predict(x_test)

    precision = precision_score(y_test,y_predicciones)
    print(precision)
```