5-MicrosoftMalwarePrediction-RegresionLogistica

May 21, 2020

1 Microsoft Malware Prediction

1.0.1 Regresión Logística

https://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/

Importamos las librerías

[13]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import pickle

from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score, precision_score, recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
from time import time
```

Lectura de los datos

[2]:

```
# Leemos los datos originales (para el submission necesitamos la variable␣
 ↪identificadora de test)

# Leemos el fichero json
import json

with open('datos/datatype.json', 'r') as myfile:
    data = myfile.read()

# Obtenemos los tipos de datos para el train
dtypes_train = json.loads(data) # Parse file
```

```
# Hacemos una copia de los tipos de datos a modificar para test
dtypes_test = dtypes_train.copy()

# Eliminamos la variable 'target'
del dtypes_test['HasDetections']

# Lectura de nuevo del conjunto de train y test, con los tipos de datos que␣
 ↪hemos definido
train = pd.read_csv("./datos/train_malware.csv", dtype = dtypes_train)
test = pd.read_csv("./datos/test_malware.csv", dtype = dtypes_test)
```

[3]:

```
# Leemos los datos con label encoding
train_label_encoding = pd.read_csv("./datos/train_filtrado_encoding.csv")
test_label_encoding = pd.read_csv("./datos/test_filtrado_encoding.csv")
```

Partición

[4]:

```
# Dividimos la variable target de
x = train_label_encoding.drop('HasDetections', axis=1)
y = train_label_encoding['HasDetections']
```

[5]:

```
# Creamos el conjunto de validación
X_train, X_val, y_train, y_val = train_test_split(x, y, test_size=0.25,␣
 ↪random_state = 3)
print(X_train.shape, y_train.shape, X_val.shape, y_val.shape)
```

```
(6580545, 58) (6580545,) (2193515, 58) (2193515,)
```

Lectura del conjunto de datos particionados

[2]:

```
# Lectura del conjunto de datos particionado
X_train = pd.read_csv("./datos/X_train.csv")
X_val = pd.read_csv("./datos/X_val.csv")
y_train = pd.read_csv("./datos/y_train.csv")
y_val = pd.read_csv("./datos/y_val.csv")
```

Algoritmo de Regresión Logística

[5]:

```
# Configuración del algoritmo de Regresión Logística
rl_model = linear_model.LogisticRegression()
```

[6]:

```
# Vemos los hiperparámetros del modelo
rl_model
```

[6]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

[7]:

```
# Entrenamiento del modelo
start_time = time()
```

```
rl_model.fit(X_train,y_train)
elapsed_time = time() - start_time

y_pred = rl_model.predict(X_val)

print("Tiempo de entrenamiento: %.10f segundos" % elapsed_time)
print("Accuracy: ", metrics.accuracy_score(y_val, y_pred))
```

```
/Users/gema/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/gema/anaconda3/lib/python3.7/site-
packages/sklearn/utils/validation.py:761: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/Users/gema/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:931:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  "the number of iterations.", ConvergenceWarning)
```

```
Tiempo de entrenamiento: 5095.5392849445 segundos
Accuracy:  0.6014360512693098
```

[10]:

```
# Imprimimos algunas métricas
logloss = metrics.log_loss(y_val, y_pred)
accuracy = metrics.accuracy_score(y_val, y_pred)
F1 = metrics.f1_score(y_val, y_pred)
precision = precision_score(y_val, y_pred, average='binary')
recall = recall_score(y_val, y_pred, average='binary')
auc = metrics.roc_auc_score(y_val, y_pred)

metricas = [logloss, accuracy, F1, precision, recall, auc, elapsed_time]
nombre_metricas = ['Log loss', 'Accuracy', 'F1 Score', 'Precision', 'Recall',␣
 ↪'AUC', 'Tiempo de entrenamiento']
```

[11]:

```
pd.DataFrame(metricas, nombre_metricas, columns = ['Regresión Logística']).T
```

[11]:

```
                      Log loss  Accuracy  F1 Score  Precision    Recall  \
Regresión Logística  13.766044  0.601436  0.573078   0.617083  0.534931

                          AUC  Tiempo de entrenamiento
Regresión Logística  0.601446              5095.539285
```

Guardamos el modelo

[14]:

```
# Guardar el modelo
pkl_filename = "modelos/regresion_logistica.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(rl_model, file)
```

Vamos a sacar las variables más importantes

[8]:

```
# Con Regresión Logística no funciona lo mismo que para Random Forest
```

Submission en Kaggle

[15]:

```
pred_rl_model = rl_model.predict(test_label_encoding)
(pred_rl_model, len(y_pred))
```

[15]:

```
(array([1, 1, 1, ..., 1, 1, 1]), 2193515)
```

[16]:

```
# Cogemos los identificadores del conjunto test
id_test = test['MachineIdentifier']

# Leemos el CSV para realizar el submission
submission = pd.read_csv("./datos/Submissions/RegresionLogistica/
 ↪sample_submission.csv")
# Vemos que 'submission.head()' coincide con 'id_test' de manera ordenada

# Pegamos la lista de los identificadores a la columna␣
 ↪submission['HasDetections']
submission['HasDetections'] = pred_rl_model
submission.head()
```

[16]:

```
                  MachineIdentifier  HasDetections
0  0000010489e3af074adeac69c53e555e              1
1  00000176ac758d54827acd545b6315a5              1
2  0000019dcefc128c2d4387c1273dae1d              1
3  0000055553dc51b1295785415f1a224d              1
4  00000574cefffeca83ec8adf9285b2bf              1
```

[17]:

```
# Guardamos el fichero CSV
submission.to_csv('./datos/Submissions/RegresionLogistica/sample_submission.
 ↪csv', index = False, header = True)
```

Validación cruzada de nuestro modelo

[9]:

```
name='Logistic Regression'
seed = 7
kfold = model_selection.KFold(n_splits=3, random_state=seed)
cv_results = model_selection.cross_val_score(rl_model, X_train, y_train,␣
 ↪cv=kfold, scoring='accuracy')
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
```

```
/Users/gema/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/gema/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
```

```
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/gema/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)

Logistic Regression: 0.597678 (0.005764)
```

[12]:

```
predictions = rl_model.predict(X_val)
print(accuracy_score(y_val, predictions))
```

```
0.6014360512693098
```