4-MicrosoftMalwarePrediction-RandomForest

May 21, 2020

1 Microsoft Malware Prediction

1.1 Random Forest

Importamos las librerías

[27]:
```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import pickle

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import f1_score, precision_score, recall_score
from time import time
```

Lectura de los datos

[13]:
```
# 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)
```

[2]:
```
# 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

[3]:
```
# Dividimos la variable target de
x = train_label_encoding.drop('HasDetections', axis=1)
y = train_label_encoding['HasDetections']
```

[4]:
```
# 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

[3]:
```
# 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 Random Forest

Partición 80-20

```
rf = RandomForestClassifier(criterion = 'entropy', max_depth = d, min_samples_split = 2,
                            oob_score=True, n_estimators=100)
```

	max_depth	n_estimators	tiempo (seg.)	tiempo	accuracy
1	2	100	885.1690599918	14 minutos	0.6197803525391894
2	3	100	1381.4913179874	23 minutos	0.6198293606380626
3	None	100	10573.8996510506	2.93 horas	0.6500776151519365
4	2	300	5432.5211529732	1.50 horas	0.619139828084148
5	3	300	33088.3806273937	9.19 horas	0.6200863682263399
6	2	700	6232.4242198467	1.73 horas	0.6193934165027365

Partición 75-25, `max_features = "auto"` y `min_samples_leaf = 50`

```
rf = RandomForestClassifier(criterion = 'entropy', max_depth = d, n_jobs = -1, oob_score = Tru
n_estimators = 100, max_features = "auto", min_samples_leaf = 50)`
```

	max_depth	n_estimators	tiempo (seg.)	tiempo	accuracy
1	3	100	503.1005158424	9 minutos	0.6197803525391894
2	5	100	711.5991830826	12 minutos	0.623109028203591
3	9	100	1596.7300050259	27 minutos	0.6290889280447136
4	12	100	1967.6013000011	33 minutos	0.6345728203363096
5	7	150	1880.0360009670	32 minutos	0.6274044171113486
6	4	500	3667.4405992031	1 hora	0.6211163361089393
7	6	500	5644.0747678280	1.56 horas	0.625054307811891
8	8	500	7002.1717000008	1.95 horas	0.6281078542886646

[4]:
```
# Configuración del algoritmo Random Forest
rf_model = RandomForestClassifier(criterion = 'entropy', max_depth = 12, n_jobs␣
 ↪= -1, oob_score = True,
                                   n_estimators = 100, max_features = "auto",␣
 ↪min_samples_leaf = 50)
```

[5]:
```
# Entrenamiento del modelo
start_time = time()
rf_model.fit(X_train, y_train)
elapsed_time = time() - start_time

y_pred = rf_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/ipykernel_launcher.py:3:
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().


Tiempo de entrenamiento: 1854.0278778076 segundos
Accuracy:  0.6349548555628751
```

[159]:
```
# 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']
```

[163]:

```
pd.DataFrame(metricas, nombre_metricas, columns = ['Random Forest']).T
```

[163]:

```
                Log loss  Accuracy  F1 Score  Precision    Recall       AUC  \
Random Forest  12.608369  0.634955  0.644334   0.628283  0.661227  0.634951

               Tiempo de entrenamiento
Random Forest              1854.027878
```

Guardamos el modelo

[18]:

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

Vamos a sacar las variables más importantes

[6]:

```
feature_importance = pd.DataFrame(sorted(zip(rf_model.
 ↪feature_importances_,X_train.columns)),
                                  columns=['Valor','Variable'])
```

[7]:

```
feature_importance = feature_importance.sort_values('Valor', ascending=False)
feature_importance.head()
```

[7]:

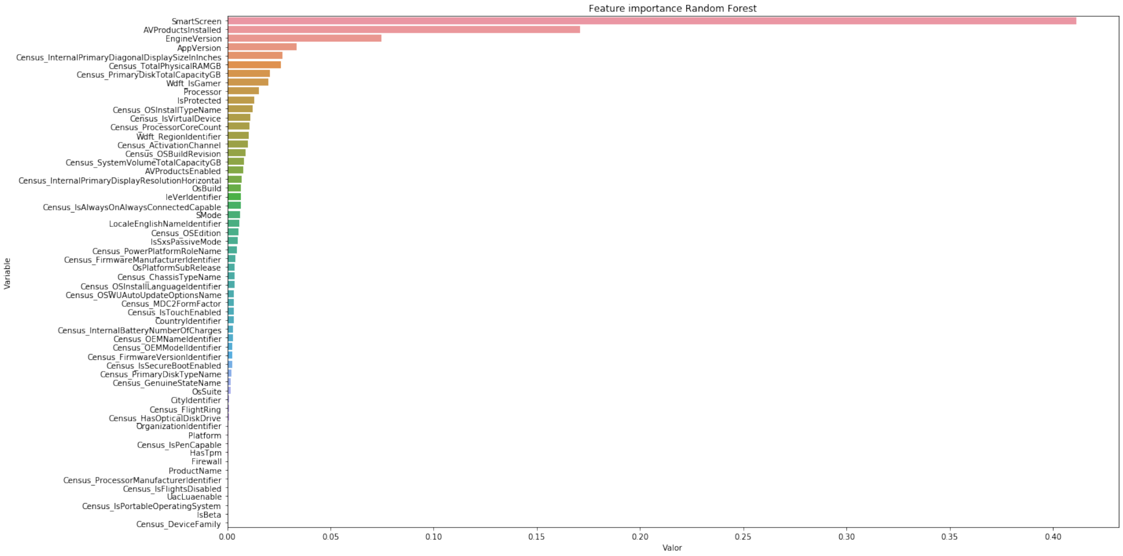
```
       Valor                                          Variable
57  0.411256                                       SmartScreen
56  0.170796                                AVProductsInstalled
55  0.074633                                     EngineVersion
54  0.033565                                        AppVersion
53  0.026912  Census_InternalPrimaryDiagonalDisplaySizeInInches
```

[8]:

```
fig = px.bar(feature_importance, x='Valor', y='Variable', orientation='h')
fig.update_layout(title_text='Feature importance Random Forest', title_x=0,␣
 ↪xaxis=dict(title='Valor'),
                 margin=dict(l=10, r=10, t=100, b=0), template='seaborn',
                  uniformtext_minsize=6,)
fig.show()
```

[9]:

```
plt.figure(figsize=(20, 10))
sns.barplot(x="Valor", y="Variable",
            data=feature_importance.sort_values(by="Valor", ascending=False))
plt.title('Feature importance Random Forest')
plt.tight_layout()
plt.show()
```



Submission en Kaggle

[15]:
```
pred_rf_model = rf_model.predict(test_label_encoding)
(pred_rf_model, len(y_pred))
```

[15]:
```
(array([1, 1, 0, ..., 1, 0, 0]), 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/RandomForest/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_rf_model
submission.head()
```

[16]:
```
                  MachineIdentifier  HasDetections
0  0000010489e3af074adeac69c53e555e              1
1  00000176ac758d54827acd545b6315a5              1
2  0000019dcefc128c2d4387c1273dae1d              0
3  0000055553dc51b1295785415f1a224d              1
4  00000574cefffeca83ec8adf9285b2bf              1
```

[17]:
```
# Guardamos el fichero CSV
submission.to_csv('./datos/Submissions/RandomForest/sample_submission.csv',␣
 ↪index = False, header = True)
```