4-MicrosoftMalwarePrediction-RandomForest

May 21, 2020

1 Microsoft Malware Prediction

1.1 Random Forest

1.1.1 Importamos las librerías

[1]:
```
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
```

1.2 ——

Lectura de los datos

[ ]:
```
# 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)
```

[ ]:
```
# 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

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

[ ]:
```
# 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)
```

1.3 ——

1.3.1 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")
```

1.3.2 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 = True
                            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

Modelo 1

[3]:
```
# Configuración del algoritmo Random Forest
rf_model_01 = 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_01.fit(X_train, y_train)
time_rf_model_01 = time() - start_time

y_pred_01 = rf_model_01.predict(X_val)

print("Tiempo de entrenamiento: %.10f segundos" % time_rf_model_01)
print("Accuracy:", metrics.accuracy_score(y_val, y_pred_01))
```

```
/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: 2331.5107879639 segundos
Accuracy: 0.6352534630490332
```

[7]:
```
# Imprimimos algunas métricas
logloss = metrics.log_loss(y_val, y_pred_01)
accuracy = metrics.accuracy_score(y_val, y_pred_01)
F1 = metrics.f1_score(y_val, y_pred_01)
precision = precision_score(y_val, y_pred_01, average='binary')
recall = recall_score(y_val, y_pred_01, average='binary')
```

```
auc = metrics.roc_auc_score(y_val, y_pred_01)

metricas = [logloss, accuracy, F1, precision, recall, auc, time_rf_model_01]
nombre_metricas = ['Log loss', 'Accuracy', 'F1 Score', 'Precision', 'Recall',␣
↪'AUC', 'Tiempo de entrenamiento']

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

[7]:
```
                 Log loss  Accuracy  F1 Score  Precision    Recall       AUC  \
Random Forest   12.598056  0.635253  0.644785   0.628454  0.661989  0.635249

                Tiempo de entrenamiento
Random Forest                2331.510788
```

[8]:
```
# Guardamos el modelo
pkl_filename = "modelos/random_forest_01.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(rf_model_01, file)
```

Modelo 2

[9]:
```
# Configuración del algoritmo Random Forest
rf_model_02 = RandomForestClassifier(criterion = 'gini', max_depth = 6, n_jobs␣
↪= -1, oob_score = True,
                                     n_estimators = 200, max_features = "auto",␣
↪min_samples_leaf = 200)
```

[10]:
```
# Entrenamiento del modelo
start_time = time()
rf_model_02.fit(X_train, y_train)
time_rf_model_02 = time() - start_time

y_pred_02 = rf_model_02.predict(X_val)

print("Tiempo de entrenamiento: %.10f segundos" % time_rf_model_02)
print("Accuracy:", metrics.accuracy_score(y_val, y_pred_02))
```

```
/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: 2653.1309859753 segundos
Accuracy: 0.6240723222772582
```

[13]:
```
# Imprimimos algunas métricas
logloss = metrics.log_loss(y_val, y_pred_02)
accuracy = metrics.accuracy_score(y_val, y_pred_02)
```

```
F1 = metrics.f1_score(y_val, y_pred_02)
precision = precision_score(y_val, y_pred_02, average='binary')
recall = recall_score(y_val, y_pred_02, average='binary')
auc = metrics.roc_auc_score(y_val, y_pred_02)

metricas = [logloss, accuracy, F1, precision, recall, auc, time_rf_model_02]
nombre_metricas = ['Log loss', 'Accuracy', 'F1 Score', 'Precision', 'Recall',␣
 ↪'AUC', 'Tiempo de entrenamiento']

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

[13]:
```
                Log loss  Accuracy  F1 Score  Precision    Recall       AUC  \
Random Forest  12.984252  0.624072  0.641858   0.612946  0.673631  0.624065

               Tiempo de entrenamiento
Random Forest              2653.130986
```

[14]:
```
# Guardar el modelo
pkl_filename = "modelos/random_forest_02.pkl"
with open(pkl_filename, 'wb') as file:
    pickle.dump(rf_model_02, file)
```

1.3.3 Sacamos las variables más importantes del mejor modelo

[15]:
```
feature_importance = pd.DataFrame(sorted(zip(rf_model_01.
 ↪feature_importances_,X_train.columns)),
                                  columns=['Valor','Variable'])
```

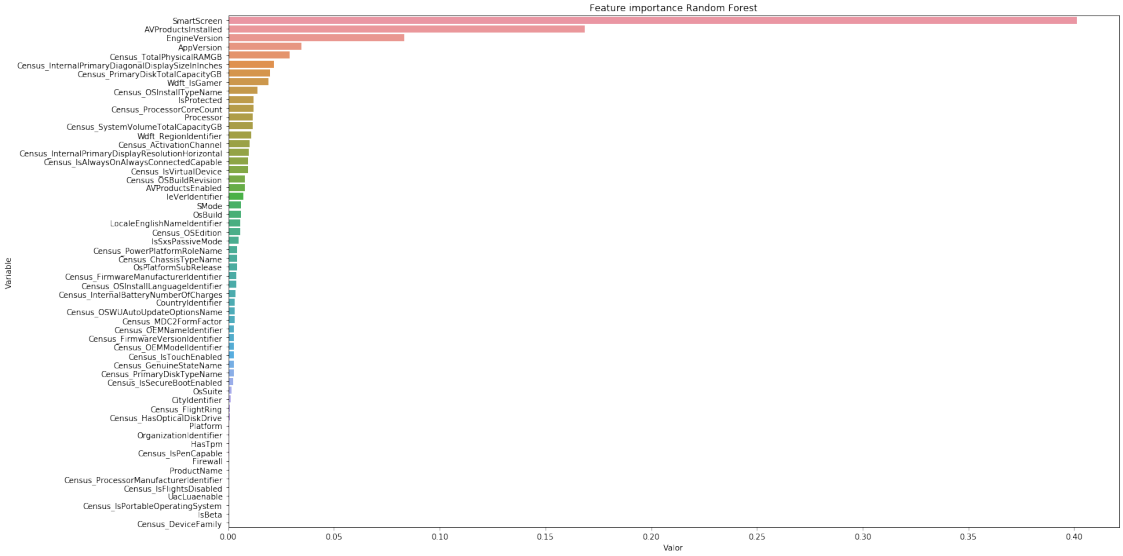
[17]:
```
feature_importance = feature_importance.sort_values('Valor', ascending=False)
feature_importance.head(10)
```

[17]:

	Valor	Variable
57	0.401149	SmartScreen
56	0.168516	AVProductsInstalled
55	0.083330	EngineVersion
54	0.034807	AppVersion
53	0.029097	Census_TotalPhysicalRAMGB
52	0.021547	Census_InternalPrimaryDiagonalDisplaySizeInInches
51	0.019983	Census_PrimaryDiskTotalCapacityGB
50	0.018912	Wdft_IsGamer
49	0.013920	Census_OSInstallTypeName
48	0.012147	IsProtected

[18]:
```
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()
```

[19]:
```
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

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

[ ]:
```
# 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()
```

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