7-MicrosoftMalwarePrediction-MLflow

May 23, 2020

1 Microsoft Malware Prediction

1.1 MLflow

1.1.1 Importamos las librerías

[2]:
```
import mlflow
import mlflow.sklearn
import pandas as pd
import numpy as np
import pickle

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

1.1.2 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")
```

Definimos las métricas a evaluar

[4]:
```
def eval_metricas(y_val, y_pred):
    rmse = np.sqrt(mean_squared_error(y_val, y_pred))
    mae = mean_absolute_error(y_val, y_pred)
    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)
```

```
    return (rmse, mae, logloss, accuracy, F1, precision, recall, auc)
```

1.1.3 Random Forest con MLflow

Modelo 1

```
RandomForestClassifier(criterion = 'entropy', max_depth = 12, n_jobs = -1, oob_score = True,
                       n_estimators = 100, max_features = "auto", min_samples_leaf = 50)
```

[5]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    # Cargamos el modelo 01
    pkl_filename_01 = "modelos/random_forest_01.pkl"
    with open(pkl_filename_01, 'rb') as file:
        rf_model_01 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 01
    predict_01 = rf_model_01.predict(X_val)

    # Calculamos diversas medidas
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict_01)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(rf_model_01, "RFmodel01")
```

```
/Users/gema/anaconda3/lib/python3.7/site-packages/py4j/java_collections.py:13:
DeprecationWarning: Using or importing the ABCs from 'collections' instead of
from 'collections.abc' is deprecated, and in 3.8 it will stop working
  from collections import (
```

[6]:
```
print("Random Forest 01")
print(" RMSE: %s" % rmse)
print(" MAE: %s" % mae)
```

```
print(" Log Loss: %s" % logloss)
print(" Accuracy: %s" % accuracy)
print(" F1 Score: %s" % F1)
print(" Precision: %s" % precision)
print(" Recall: %s" % recall)
print(" AUC: %s" % auc)
```

```
Random Forest 01
 RMSE: 0.603942494738503
 MAE: 0.3647465369509668
 Log Loss: 12.598055574220245
 Accuracy: 0.6352534630490332
 F1 Score: 0.644785486621467
 Precision: 0.6284537260213509
 Recall: 0.6619887284601822
 AUC: 0.6352494524898191
```

Modelo 2

```
RandomForestClassifier(criterion = 'gini', max_depth = 6, n_jobs = -1, oob_score = True,
                       n_estimators = 200, max_features = "auto", min_samples_leaf = 200)
```

[7]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    # Cargamos el modelo 02
    pkl_filename_02 = "modelos/random_forest_02.pkl"
    with open(pkl_filename_02, 'rb') as file:
        rf_model_02 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 01
    predict_02 = rf_model_02.predict(X_val)

    # Calculamos diversas medidas
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict_02)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
```

```
    mlflow.sklearn.log_model(rf_model_02, "RFmodel02")
```

[8]:
```
print("Random Forest 02")
print(" RMSE: %s" % rmse)
print(" MAE: %s" % mae)
print(" Log Loss: %s" % logloss)
print(" Accuracy: %s" % accuracy)
print(" F1 Score: %s" % F1)
print(" Precision: %s" % precision)
print(" Recall: %s" % recall)
print(" AUC: %s" % auc)
```

```
Random Forest 02
 RMSE: 0.6131294135194802
 MAE: 0.3759276777227418
 Log Loss: 12.984252090973918
 Accuracy: 0.6240723222772582
 F1 Score: 0.6418576773820872
 Precision: 0.6129464141289714
 Recall: 0.6736313065103992
 AUC: 0.624064887930234
```

1.1.4 Regresión Logística con MLflow

```
# Modelo 1
LogisticRegression()

# Modelo 2
LogisticRegression(n_jobs=-1, max_iter=300)
```

[9]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    ### Modelo 01 ###

    # Cargamos el modelo 01
    pkl_filename_01 = "modelos/regresion_logistica_01.pkl"
    with open(pkl_filename_01, 'rb') as file:
        rl_model_01 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 01
    predict_01 = rl_model_01.predict(X_val)

    # Calculamos diversas medidas para modelo 01
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict_01)
```

```
    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(rl_model_01, "RLmodel01")

    # ---------------------------------------------------- #

    ### Modelo 02 ###

    # Cargamos el modelo 02
    pkl_filename_02 = "modelos/regresion_logistica_02.pkl"
    with open(pkl_filename_02, 'rb') as file:
        rl_model_02 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 02
    predict_02 = rl_model_02.predict(X_val)

    # Calculamos diversas medidas para modelo 02
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
↪eval_metricas(y_val, predict_02)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(rl_model_02, "RLmodel02")
```

Modelo 1

```
LogisticRegression()
```

[12]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    ### Modelo 01 ###

    # Cargamos el modelo 01
    pkl_filename_01 = "modelos/regresion_logistica_01.pkl"
    with open(pkl_filename_01, 'rb') as file:
        rl_model_01 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 01
    predict_01 = rl_model_01.predict(X_val)

    # Calculamos diversas medidas para modelo 01
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict_01)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(rl_model_01, "RLmodel01")
```

[13]:
```
print("Regresión Logística 01")
print(" RMSE: %s" % rmse)
print(" MAE: %s" % mae)
print(" Log Loss: %s" % logloss)
print(" Accuracy: %s" % accuracy)
print(" F1 Score: %s" % F1)
print(" Precision: %s" % precision)
print(" Recall: %s" % recall)
print(" AUC: %s" % auc)
```

```
Regresión Logística 01
 RMSE: 0.6313192130219785
 MAE: 0.39856394873069023
 Log Loss: 13.766043833266938
 Accuracy: 0.6014360512693098
 F1 Score: 0.573077997394292
 Precision: 0.617082943347811
 Recall: 0.5349313807180456
```

```
AUC: 0.6014460276400377
```

Modelo 2

```
LogisticRegression(n_jobs=-1, max_iter=300)
```

[14]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    ### Modelo 02 ###

    # Cargamos el modelo 02
    pkl_filename_02 = "modelos/regresion_logistica_02.pkl"
    with open(pkl_filename_02, 'rb') as file:
        rl_model_02 = pickle.load(file)

    # Obtenemos las predicciones con el modelo 02
    predict_02 = rl_model_02.predict(X_val)

    # Calculamos diversas medidas para modelo 02
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict_02)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(rl_model_02, "RLmodel02")
```

[15]:
```
print("Regresión Logística 02")
print(" RMSE: %s" % rmse)
print(" MAE: %s" % mae)
print(" Log Loss: %s" % logloss)
print(" Accuracy: %s" % accuracy)
print(" F1 Score: %s" % F1)
print(" Precision: %s" % precision)
print(" Recall: %s" % recall)
print(" AUC: %s" % auc)
```

```
Regresión Logística 02
 RMSE: 0.6281548889735611
```

```
MAE: 0.3945785645413868
Log Loss: 13.628411310275684
Accuracy: 0.6054214354586133
F1 Score: 0.5981926155914526
Precision: 0.6094510608912811
Recall: 0.5873425822437693
AUC: 0.6054241474687699
```

1.1.5 Gradient Boosting con MLflow

```
GradientBoostingClassifier(n_estimators=50, learning_rate=0.75,
                           max_depth=4, validation_fraction=0.2, random_state=9)
```

[10]:
```
# Iniciamos 'mlflow'
with mlflow.start_run():

    # Cargamos el modelo (solo hay uno)
    pkl_filename = "modelos/gradient_boosting.pkl"
    with open(pkl_filename, 'rb') as file:
        gb_model = pickle.load(file)

    # Obtenemos las predicciones con el modelo 01
    predict = gb_model.predict(X_val)

    # Calculamos diversas medidas
    (rmse, mae, logloss, accuracy, F1, precision, recall, auc) =␣
 ↪eval_metricas(y_val, predict)

    # Log mlflow parámetros
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("mae", mae)
    mlflow.log_metric("logloss", logloss)
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1", F1)
    mlflow.log_metric("precision", precision)
    mlflow.log_metric("recall", recall)
    mlflow.log_metric("auc", auc)

    # Log modelo generado
    mlflow.sklearn.log_model(gb_model, "GBmodel")
```

[11]:
```
print("Gradient Boosting")
print(" RMSE: %s" % rmse)
print(" MAE: %s" % mae)
print(" Log Loss: %s" % logloss)
print(" Accuracy: %s" % accuracy)
print(" F1 Score: %s" % F1)
```

```
print(" Precision: %s" % precision)
print(" Recall: %s" % recall)
print(" AUC: %s" % auc)
```

```
Gradient Boosting
 RMSE: 0.5936602658835763
 MAE: 0.3524325112889586
 Log Loss: 12.172721138968884
 Accuracy: 0.6475674887110414
 F1 Score: 0.6409167592554746
 Precision: 0.6533458463624776
 Recall: 0.6289517395038116
 AUC: 0.6475702812610071
```
