REPRODUCIBILITY

In this section we report additional information regarding our experimental evaluation of Section 7. We are taking the following steps to ensure reproducibility of our experiments:

Datasets: As mentioned in Section 7, we used a large corpus of **publicly** available Python scripts (26*K*). This dataset is available at [16].

Experiments with Labeled Datasets: In Section 7.2, we presented experiments with scripts for which we have manually extracted the provenance information (ground truth). The ground truth for the scripts that were used in these experiments is also available at [16].

Algorithms: The pseudocode for all the algorithms used by Vamsa is either presented in this paper, or is available in our technical report [35].

Knowledge Base: A file with the data stored in the knowledge base that we used for our experiments is available at [16].

Real Use Case: In section 7.5, we presented a real scenario encountered in production that highlights the significance of using tools such as Vamsa to collect provenance information from ML scripts. Figure 8 presents a simplified but representative version of the script that is used in production.

```
import warnings

warnings.filterwarnings("ignore", category=UserWarning)

import pandas as pd
import lightgbm as lgb
from sklearn import metrics
import numpy as np

data_train = pd.read_csv("global_train.csv")
data_test = pd.read_csv("global_test.csv")

# Feature Engineering
data_train["SuccessfulVertices"] = (data_train["TotalNumberOfVertices"] - data_train[
    "RevocationCount"] - data_train["FailedCount"]) / data_train["TotalNumberOfVertices"]

data_test["SuccessfulVertices"] = (data_test["TotalNumberOfVertices"] - data_test[
    "RevocationCount"] - data_test["FailedCount"]) / data_test["TotalNumberOfVertices"]

train_x = data_train.drop(columns=[
    "reason",
    "TotalNumberOfVertices",
])

train_y = data_train["reason"]

test_x = data_test.drop(columns=[
    "reason",
    "TotalNumberOfVertices",
])

test_y = data_test["reason"]

# Train/Test the model
n_leaves = 8
n_trees = 100
clf = lgb.LGBMClassifier(num_leaves=n_leaves, n_estimators=n_trees)
clf.fit(np.array(train_x), train_y)
score = metrics.precision_score(test_y, clf.predict(test_x), average='macro')
print("Precision Score on Test Data: " + str(score))
```

Figure 8: Simplified data science script used in our real example