<u>MALWARE CLASSIFICATION</u>

1. EDA evaluation.

- We do have both numerical and categorical features. Out of the last, many of them are boolean or quasi-boolean.
- The target distribution is almost balanced to 50% for both values. No nulls in it.
- Imputation of -1 for numerical features nulls, and Other for categorical features nulls.
- Converting all features to numerical: first we reduced some of the features labels, then used One Hot Encoding (OHE) function for converting categorical features values into numerical features. The reason behind the label reduction is to reduce the potential features that could be created with OHE, and do not create features with minimal or small explicative value for mode construction. This will be provide time and computacional efficiency.
- Visualize features correlation, eliminating those with high correlation since those will provide similar information to the future model. Also a good way to reduce the final number of features to be passed to the model construction.

2. Functions used:

- Treating nulls for numerical features: nulos_numericas (df, column, fillna_value).
- Group features labels: SetOthers(dataframe, column, num_values)
- Plotting categorical features: plot_cat_values(dataframe, column, target_col)
- Converting categorical features into numerical: OHE (data, column_name)

3. Model construction.

- Validation split was done using AppVersion. This feature provides Windows version, and helped us to use updated info (last version) as validation information.
- RandomHoldout was used for developmet partition (train and test).
- Execute and visualize the Decision Tree. We observe how the model chooses SmartScreen (Windows Cloud solution for malware detection) as the feature for starting the analysis.

4. Model evaluation:

- Using results, Accuracy of 60.64%
- Confusion Matrix, Accuracy of 60.64%

There is a 13% of positive malware cases that are not detected by the model. This is a crucial metrics to be improved, since we are talking about false negatives. Those are the cases where probably Windows has the most of the interest to detect and work on them, since are the wholes for malware to get into its operating system.

- Module metrics and model score, same accuracy, 60.64%
- Probabilistic evaluation with scoring, mean of 67% correct identification.

5. Validation strategies:

- Our DecisionTree could have more depth (branches), and still gaining accuracy at the train level without reducing the one at the test level. We could go as depth as 13 branches. Validation score still increases, same for training and test.
- We can play around with different validation strategies, so that we can find the most efficient set up for the model. That includes parameters alteration, such as min_samples_split size. Once we increase this, we will gain more depth and a better accuracy, but this will increase our computacional needs, and eventually will get to undesired overfitting scenarios.
- Other metrics accuracy results are: K-fold as validation strategy: 0.61, F1: 0.649, AUC: en principio 0.659.

6. Alternative models evaluation:

RandomForest and GradientBoosting:

Altenative models are not significantly improving model success, measures for all three models (includent Decison Tree) by ROC AUC Score. This could be explained because both RandomForets and GradientBoosting work better with small data samples. In this case we do have a good sample, 500000 cases. Even that being the case GradientBoosting achieved to increase AUC from 0.65 (DecisionTree model) to 0.69.

It is interesting to notice the top features with Feature Importance function. This list provides information about the features that are considered by the model the ones that have more impact to detect malware. Considering this, Windows could focus on improving the performance of such features, and reduce the malware infection cases.