SAVE THE ATTACK!! ML-PROJECT

Problem Statement:

SecurityPlus.ai is an upcoming computer security software company.

The existing products of the company provide total malware protection for systems but a new research team of the company is focusing on a different problem statement of predicting the possibility of a system getting infected by malware.

You are a newly hired ML Engineer who has been assigned the task of working on this challenging big data problem.

The dataset for the above has been collected by monitoring users' systems who have availed the services of SecurityPlus.ai.

Each row in the data corresponds to one machine and includes the machine information along with other threat information which includes Engine Version, Processor, OS version, Firewall, etc.(to name a few).

The data collected ensures total user privacy.

The task is to save this virtual attack!

DataSet Details:

The dataset consists of a train_data.csv file of size 3.96 GB and a test_data.csv file of size 985.16 MB. Both files contain null values for some features which need to be handled accordingly.

Pre-processing Techniques:

Initial Pre-processing:

- We first read the training dataset in chunks of size 1,00,000.
- We then fetched 1,00,000 rows of the training data from the training chunk.
- We then identified the categorical and numerical features from the data frame and stored them in separate lists and data frames.
- Numerical features like `'DefaultBrowsersIdentifier','Census_IsWIMBootEnabled','Census_ThresholdOptIn','Census_IsFlightingInternal'` having null value percentage more than 60% were removed directly.

- The remaining numerical features containing null values were replaced by the mean of the column of that respective feature.

- Categorical features like `'PuaMode','Census_ProcessorClass','Census_InternalBatteryType'` having null value percentage more than 60% were removed directly.
- "Machine Identifier" feature which consists of all unique values was dropped from the categorical feature data frame.
- The remaining categorical features containing null values were replaced by the mode of the column of that respective feature.
- To check for a chunk of train data and test data we defined a new function that would segregate features based on a number of unique values to what encoding to be used on them. If the number of unique values was less than the threshold then they were listed for one hot encoding else for label encoding
- The threshold value was assumed as 100.

- Then we concatenated the numerical features data frame and categorical features data frame into "df_together" data frame.
- We then find the correlation matrix of the final data frame.
- (As an **experiment**) we also stored the features having correlation values greater than 0.89 (or 0.97) with other features in a different set. We also removed those features from the final data frame.
- We then scaled and normalize the features from the final data frame. Scaling was done using StandardScaler() function.

Final Pre-processing :

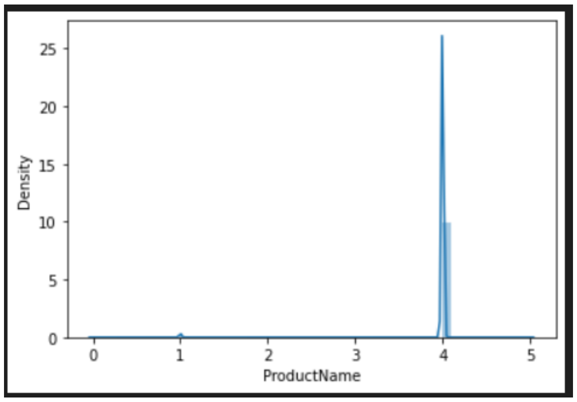
- Whole train and test datasets were loaded into the pandas data frame.
- To reduce the RAM usage, we defined a special function 'reduce_mem_usage(df)' which does as follows:
  For numerical features, It finds the min and max value of that feature column and then based on the data type it finds the appropriate range which would contain its min and max value. For eg, if a feature is of int64, but it's min and max values would fit in the range of int32, then we would replace int64 by int32.
  For categorical features, the data type "object" is replaced by "category".
  Thus this function reduces memory usage of the train test data frame by approximately 60%. Thus this gives us plenty of space for doing further preprocessing.
- Now we handle null values. We find the total percentage of null values present in each of the feature columns for both train and test data frames. If more than 50% of the null values are present in a column, then we dropped that column. For the rest null values, we do the following:

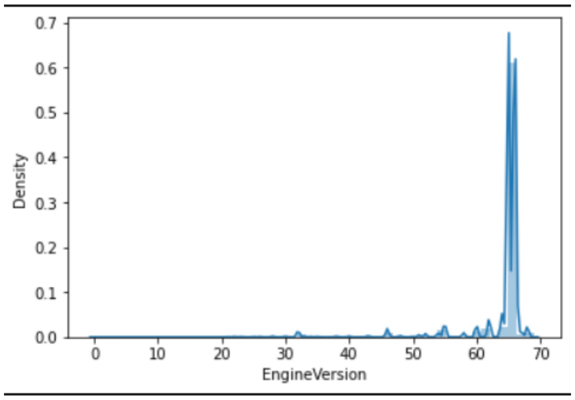
For numerical features, we replace it with the mean of that column.
For categorical features, we replace it with the mode of that column.

- Once we are done with handling null values, we then handle the categorical features.
- All categorical features were labeled encoded.
- Once the preprocessing was done for both train and test data frames we stored them in CSV file format for future use.

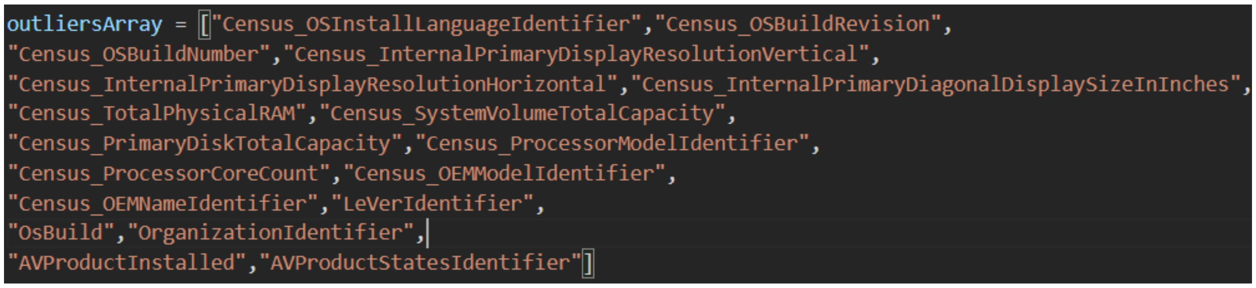
EDA:

- We plotted the distribution plot for some of the features. Some of them are attached below:





- Some of the numerical features contains outliers which needed to be taken care of. Those features are stored in separate list "outliersArray".

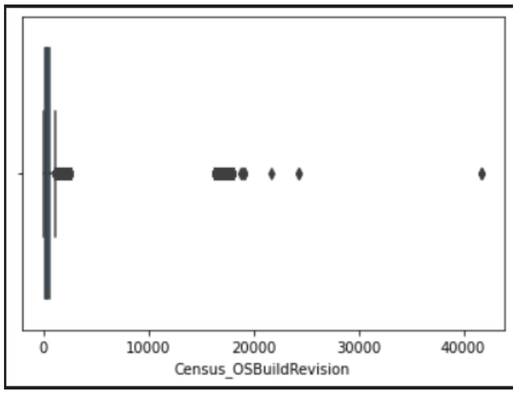


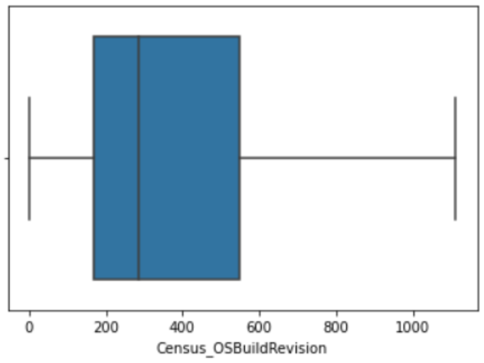
```
outliersArray = ["Census_OSInstallLanguageIdentifier","Census_OSBuildRevision",
"Census_OSBuildNumber","Census_InternalPrimaryDisplayResolutionVertical",
"Census_InternalPrimaryDisplayResolutionHorizontal","Census_InternalPrimaryDiagonalDisplaySizeInInches",
"Census_TotalPhysicalRAM","Census_SystemVolumeTotalCapacity",
"Census_PrimaryDiskTotalCapacity","Census_ProcessorModelIdentifier",
"Census_ProcessorCoreCount","Census_OEMModelIdentifier",
"Census_OEMNameIdentifier","LeVerIdentifier",
"OsBuild","OrganizationIdentifier",
"AVProductInstalled","AVProductStatesIdentifier"]
```

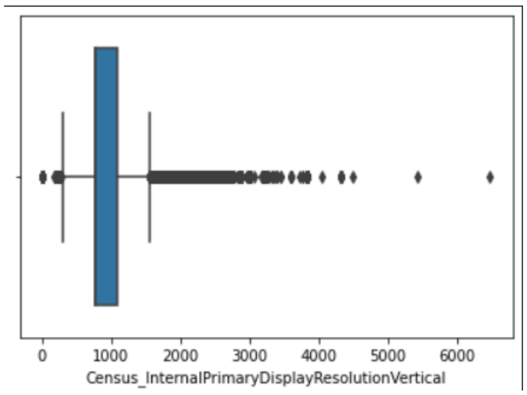
- We plotted the box plot for all these features. We used Z-Score method to remove outliers from these features.

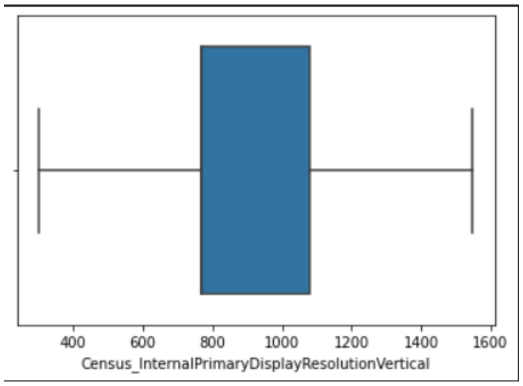
- For some features like `'Census_OSInstallLanguageIdentifier'`, outliers were replaced by max value of that particular feature.

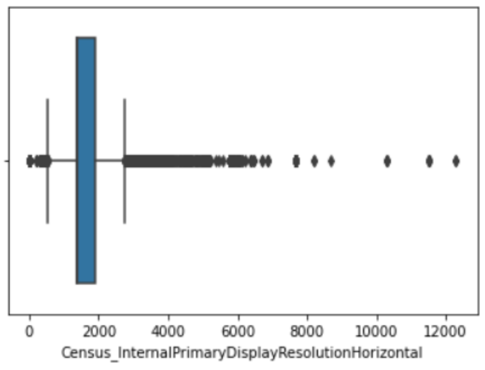
- For some features, outliers were replaced by the mean of that particular feature values. Some of them are attached below:

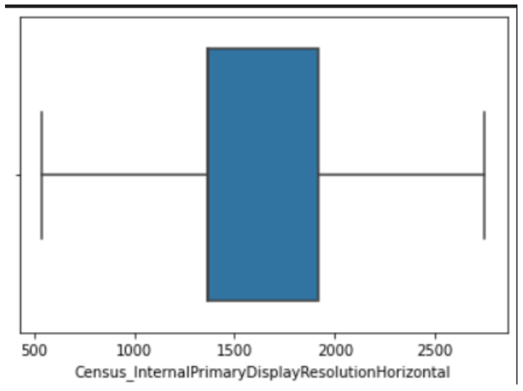






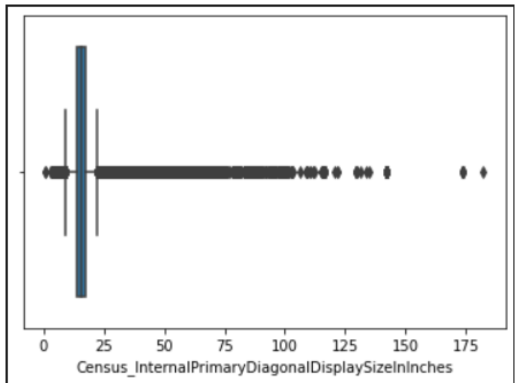


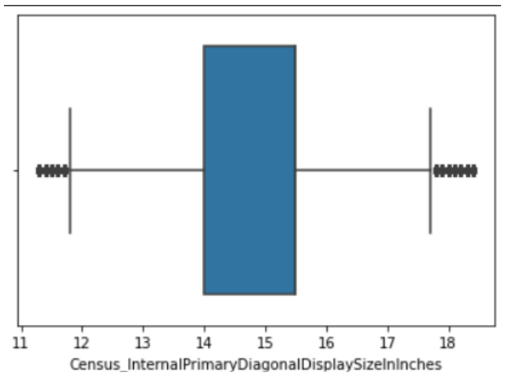


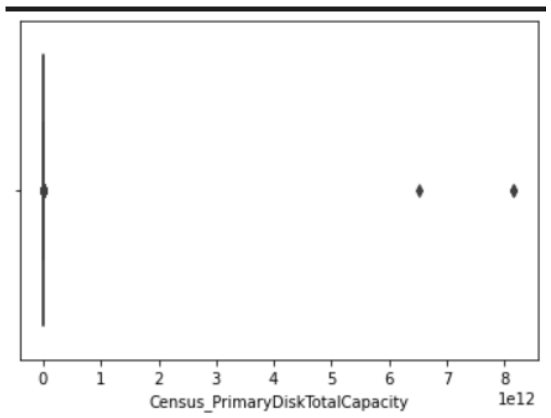


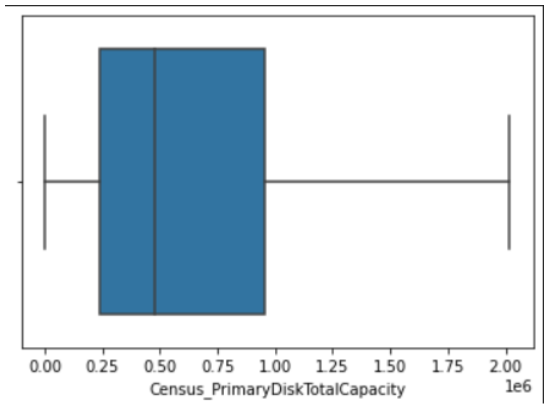
- Features like `'Census_TotalPhysicalRAM', 'Census_SystemVolumeTotalCapacity' 'Census_ProcessorCoreCount',OsBuild, 'OrganizationIdentifier'` containing outliers were also replaced with the mean of that respective feature values.

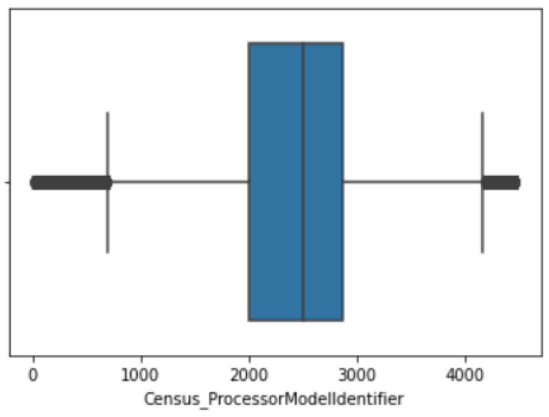
- For some features, outliers were replaced by the median of that particular feature values. Some of them are attached below:

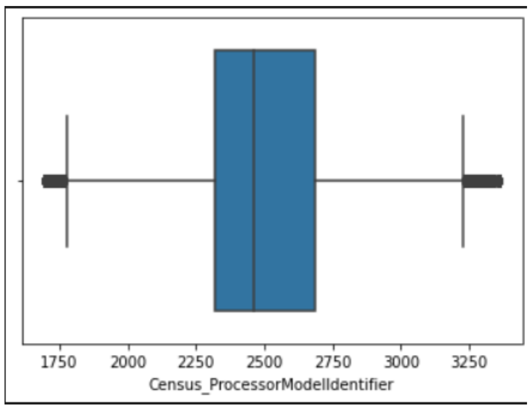








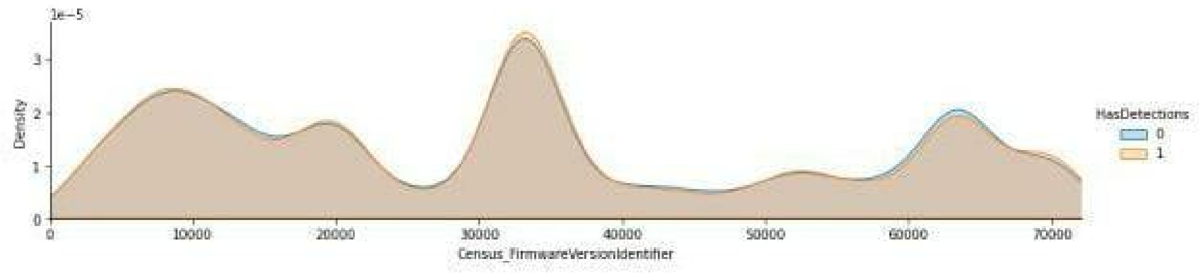


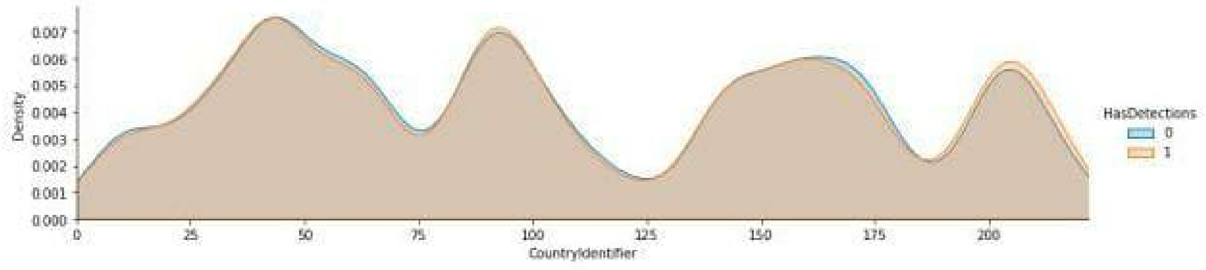


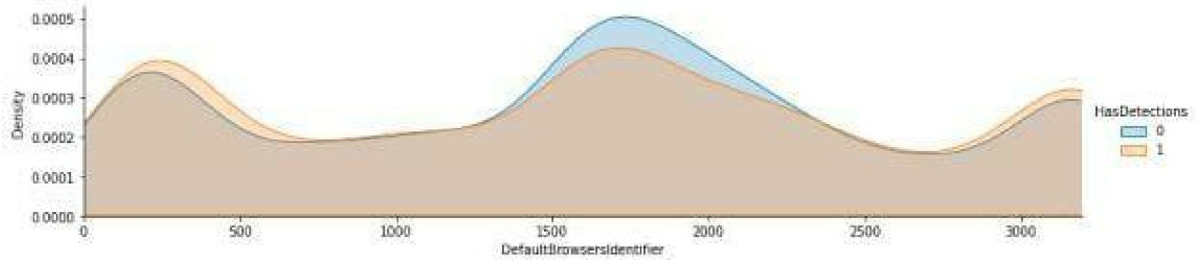
- Features like `'Census_OEMModelIdentifier', 'Census_OEMNameIdentifier' ,'AVProductStatesIdentifier'` containing outliers were also replaced with the median of that respective feature values.

- Atlast training data without outliers was stored in different CSV file. As an experiment we used this CSV file to train different models and get the roc_auc score.

- We plotted the Kernel Distribution Estimation plot for different features and analysed the subtle differences and similarities in the graphs for both labels of "HasDetection"







- For a chunk of data we found the correlation matrix(pearson) and plotted the corresponding heatmap.

- The pairs of features which were highly correlated, one of them was dropped from the dataframe.

Models and Experiments:

At the beginning we used "Logistic Regression" , "Naive Baiyes Classifier" , "SVC linear kernel" and "SVC rbf kernel" for a chunk of a dataset.
We made a validation dataset to make predictions using the above models.
In order to train the whole dataset we used dask and dask_ml for model fitting.
The preprocessed training and testing data that we had in the CSV format are loaded in the dask data frame. X and Y were found from it and converted to dask arrays.
Models like "Logistic Regression" , "Naive Baiyes Classifier", "XGBRegressor", "Random Forest" and "linear SVC" were used to train the whole training dataset. We then used parameter hypertuning with "XGBRegressor".

Results:

For a **chunk of data** we got the following results:

- We got a roc_auc score of 0.61778 in kaggle submission using Logistic Regression model with max_iterations taken as 100000 with class weights as {0:1, 1:1} and misclassification parameter C as 1.

- We experimented further by changing the parameters used in logistic regression model and found respective roc_auc score for them. All scores were nearly around 0.61-0.62.

- For naive baiyes , we got roc_auc score of 0.51984

- For SVC linear and rbf kernel, we got roc_auc score of nearly around 0.53.

For the **whole dataset**, we got the following results:

- By training logistic regression model in dask_ml, we got kaggle score of 0.59207. We used "lbfgs" solver as parameter.
- By training naive bayes model in dask_ml, we got kaggle score of 0.51984

- By training random forest model in sklearn, we got kaggle score of 0.63018. We have taken max_depth as 10, n_estimators as 30 and n_jobs as 4.

- By training linear SVC model in sklearn, we got kaggle score of 0.55983 having C parameter value as 0.1

- By training linear SVC model in sklearn, we got kaggle score of 0.56039 having C parameter value as 0.01

- As an experiment we splitted the training data further into training dataset and validation dataset with ratio 80:20.
- We then trained the obtained training dataset with XGBRegressor model in which we got kaggle score of 0.68695. We set the objective as "binary:logistic" and and n_estimators as 50.
- We tuned hyperparameters in XGB model , for this we created a dictionary of parameters added number of boosting rounds and tweaked some of the parameters like eta, depth etc. see the result:
  For the hyperparameters with:
  - max_depth = 5, eta = 0.3, Boosting_rounds = 500, rest default ,we obtained kaggle score of 0.70122
  - max_depth = 8, eta = 0.3, Boosting_rounds = 400, rest default, we obtained kaggle score of **0.70833**

- max_depth = 8, min_child_weight = 3,num_class = 2, eta = 0.4, subsample = 1, colsample_bytree = 1, objective = 'multi:softprob', we obtained kaggle score of 0.70795
- max_depth = 6, min_child_weight = 2, eta = 0.25, subsample = 1, colsample_bytree = 1, objective = 'binary:logistic', we obtained a kaggle(roc auc) score of 0.70463.
- max_depth = 12, min_child_weight = 1, eta = 0.3, subsample = 0.75, colsample_bytree = 1, objective = 'binary:logistic',num_boost_rounds =250 ,we obtained a kaggle(roc auc) score of 0.6991.2.

Conclusion:

We got our best kaggle score as **0.70833** using the XGBRegressor model with parameters as max_depth = 8, eta = 0.3, Boosting_rounds = 400 and rest as default.

Problems faced and there Solutions :

- The training data set was too large to deal with. On Kaggle, allotted RAM is 16GB, but reading the whole CSV file into the data frame itself consumes around 10.5 GB of RAM. The rest of it was not sufficient to carry out data filtering.
- Alternately, we tried to use vaex which is similar to pandas but with more improvements in processing. Though the RAM issue was resolved as CSV was converted to hdf5 format, it was very time-consuming.
- We had trained our model only for a chunk of preprocessed data for our 1st evaluation. Thereafter we decided to reduce the memory usage in a clever way. We replaced int64 by int32/, int16/int8 in whichever range the max and min value of the numerical feature column fitted. This is explained in detail in

the preprocessing step. For categorical features we replaced the 'object' type by 'category' type.

- For training the whole training dataset we used dask. Loaded the preprocessed csv to dask dataframe and converted it to dask array.

Notebooks Submitted:

Some notebooks may be messed up. Here is the brief idea about all notebooks:

"new-v-latest-0492b8.ipynb": Test and Train data is being preprocessed and the data is stored in csv files named "fdata_train.csv" and "fdata_test.csv".

"Model_fitting_369fc2.ipynb" : We load the preprocessed data and fit different xgb models

"data-without-outliers-final.ipynb" : In this notebook we mainly removed the outliers from the numerical features and also plotted the box plot.

"save-the-attack-v2-with-svm-poly-rgb-lr-with-saga.ipynb":
In this notebook models like SVC, logistic regression etc were trained for chunk of data

.
"save-the-attack-v2-final.ipynb" : In this notebook we mainly find kde plots, correlation matrix for a chunk of data and some linear model fitting on it.
'
"new-notebook-final.ipynb": In this notebook models like logistic regression , SVC etc were trained for the whole dataset.

TEAM MEMBERS:

IMT2019064 PRATIK RATNADEEP AHIRRAO

IMT2019066 PRATYUSH UPADHYAY