

## Task 1: Data Preparation

Here we have done all the required data preparation tasks. Mainly removing columns with more than 60% null values and those which are skewed more than 98%.

I have not removed Census\_InternalBatteryType alone though it has many Null values as this column is needed for EDA.

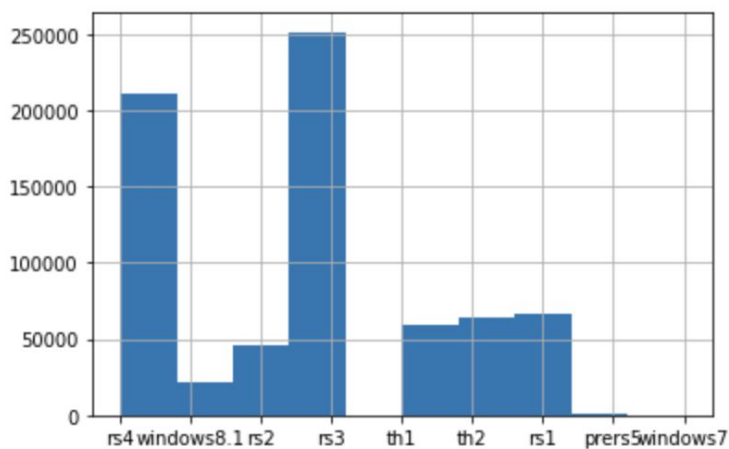
The categorical columns are checked for the available list of values and reduced as much as possible so that the computation effort required by the model is less if we are using one hot encoding. Also have eliminated junk and duplicate case values.

## Task 2: Data Exploration (EDA)

### Target variable distribution and bias:

From the count of values and histogram we can conclude that the target variable is equally distributed (almost) and the data is a balanced dataset.

**Which OsPlatformSubRelease has the maximum number of malware detections?**



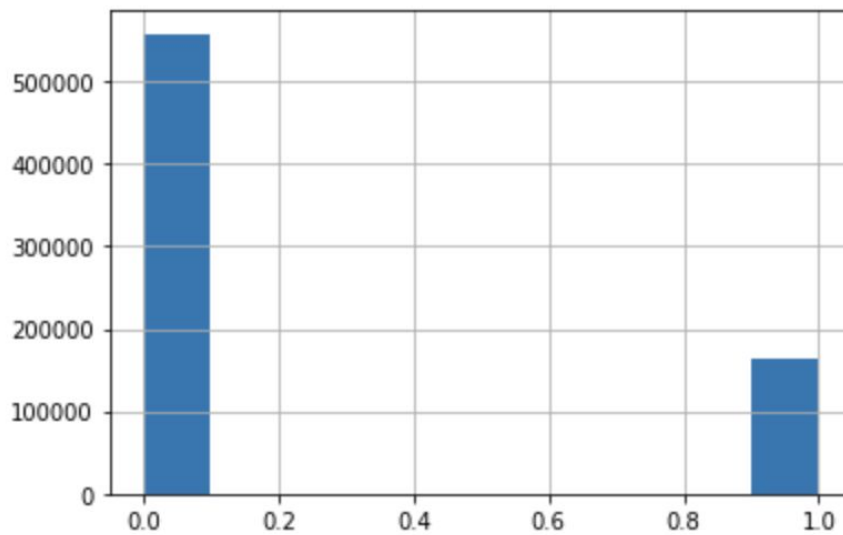
rs3 is the OsPlatformSubRelease which has maximum number of malware detections followed by rs4

Is a gamer more prone to malware attacks than a normal user?

```
Wdft_IsGamer
0.0      557494.0
1.0      163251.0
Name: HasDetections, dtype: float64
```

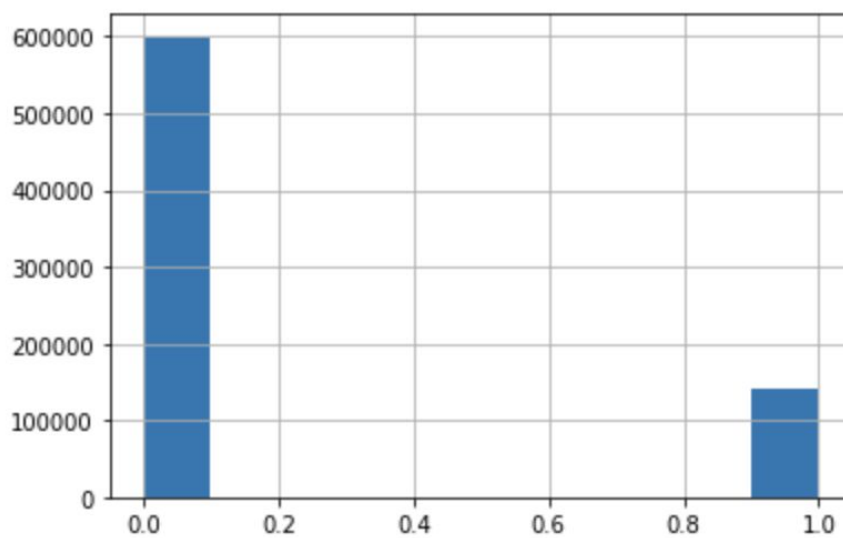
```
df_malware['Wdft_IsGamer'].hist()
```

<AxesSubplot:>



```
df[df['HasDetections'] == 0]['Wdft_IsGamer'].hist()
```

<AxesSubplot:>



From the groupby and graphs result we can conclude that only a non-gamer is more prone to malware attacks. This might be because a non-gamer is the one who tries to surf the web and explore various things which might cause malware. A gamer would only be playing a game which is trusted.

### **How does malware detection vary with different number of antivirus products?**

```
AVProductsEnabled
0.0      1644.0
1.0    705582.0
2.0    13183.0
3.0      310.0
4.0       26.0
Name: HasDetections, dtype: float64
```

Yes. As expected as the number of antivirus products enabled increases the malware detected decreases. It is also interesting that for systems which have no antivirus installed, malware is also less. But this can be due to less data about systems without any antivirus.

### **Does the presence of TPM result in better detection of malware?**

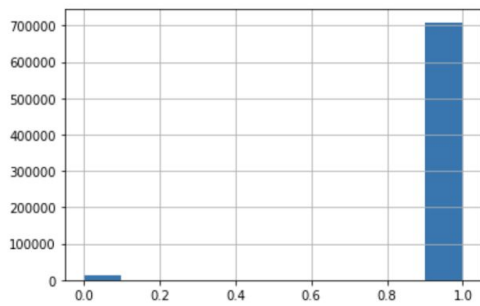
```
HasTpm
0      771.0
1   719974.0
Name: HasDetections, dtype: float64
```

Yes. The presence of TPM results in better detection of malware.

### **Which RAM type is present in most computers? Is there any RAM type which is very prone to malware attacks?**

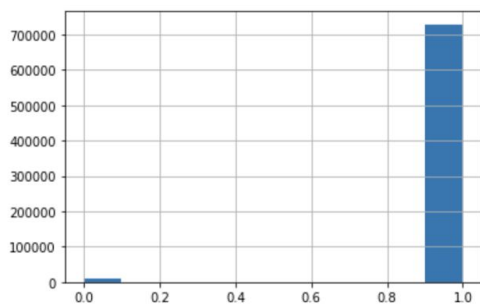
4096 MB is the RAM type in most computers and also prone to malware

Show how malware frequency is impacted by Firewall?



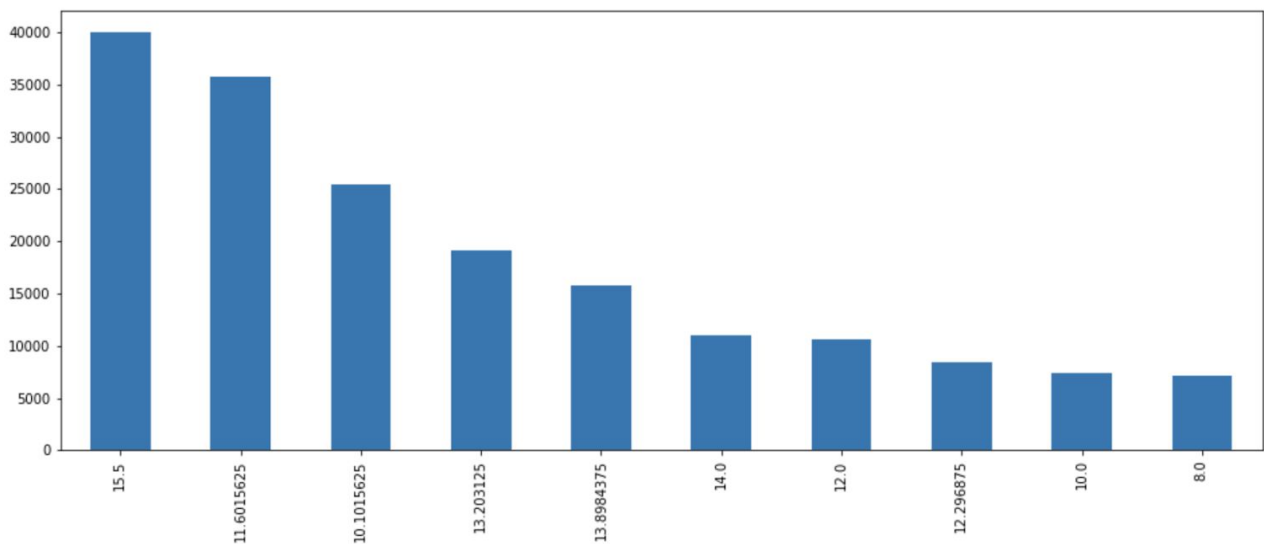
```
df[df['HasDetections'] == 0]['Firewall'].hist()
```

<AxesSubplot:>



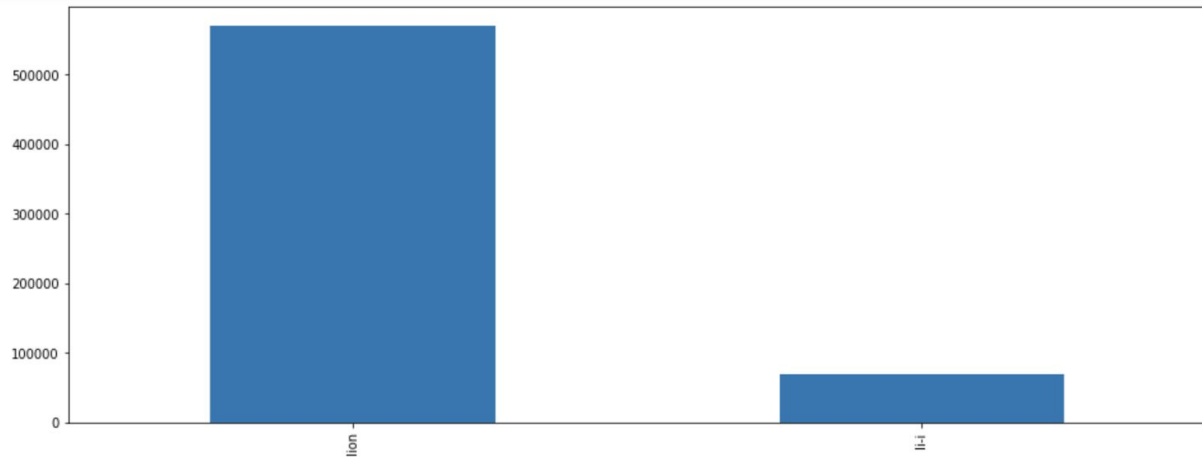
In both cases where malware is affected or not the number of systems with firewall is more. This is because we have data skewed towards systems having firewall

Counts of Census\_InternalPrimaryDiagonalDisplaySizeInInches by top 10 categories for touch devices



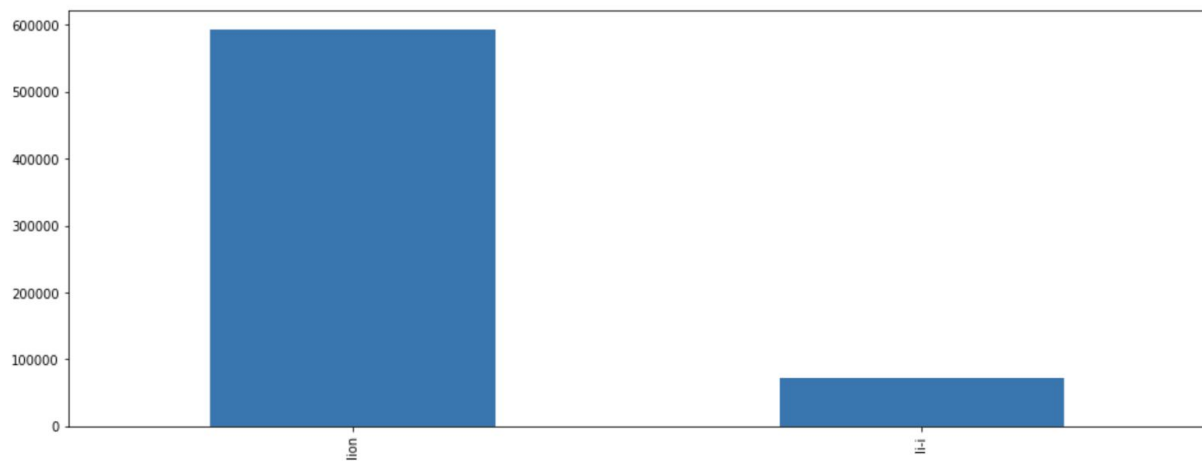
The count is highest for 15.5 inch followed by 11

## Counts of Census\_InternalBatteryType by top-2 categories with respect to malware detection %



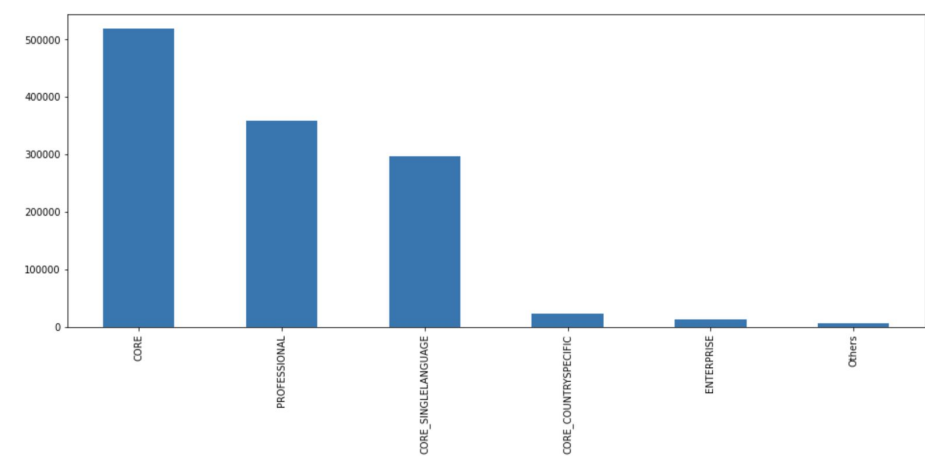
```
plt.figure(figsize=(16, 6))  
df[df['HasDetections'] == 0]['Census_InternalBatteryTypeReduced'].value_counts().head(2).plot(kind='bar')
```

<AxesSubplot:>



In aboth cases the top is lion followed by li-i category. However the amount of systems with malware detected is less than those with not detected.

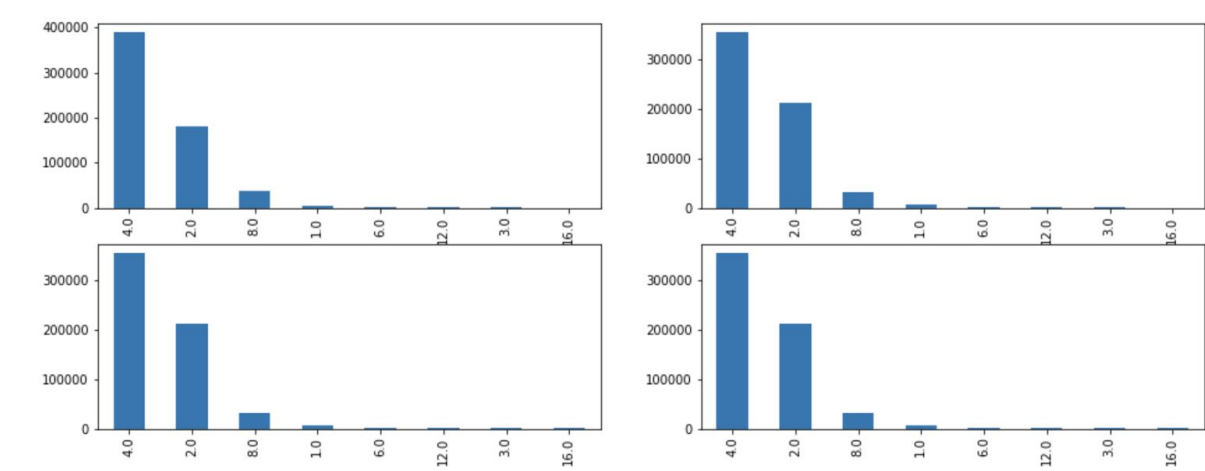
Count of Census\_OSSkuName by top 10 categories for non-touch devices



Here we have Core followed by Professional and Core\_SingleLanguage

Since we filtered many which were junk we have only 6 above and rest is grouped in Others

Count of Census\_ProcessorCoreCount by top 8 categories for touch-enabled and non touch enabled devices w.r.t malware detection %



For both malware detected and not detected, Census\_ProcessorCoreCount top categories are 4, 2, 8. Rest is very minimal. This trend is same for touch and non touch devices. Since most of the columns are categorical we could not draw much variety of plots.

## Task 3: Data Preparation for SageMaker

Since there are a lot of categorical values, we need to do either one hot or label encoding.

We choose label encoding for categorical values which has an order associated with them. Because then the encoding values would convey that information to the ML model. Here there is a weight associated with each category value.

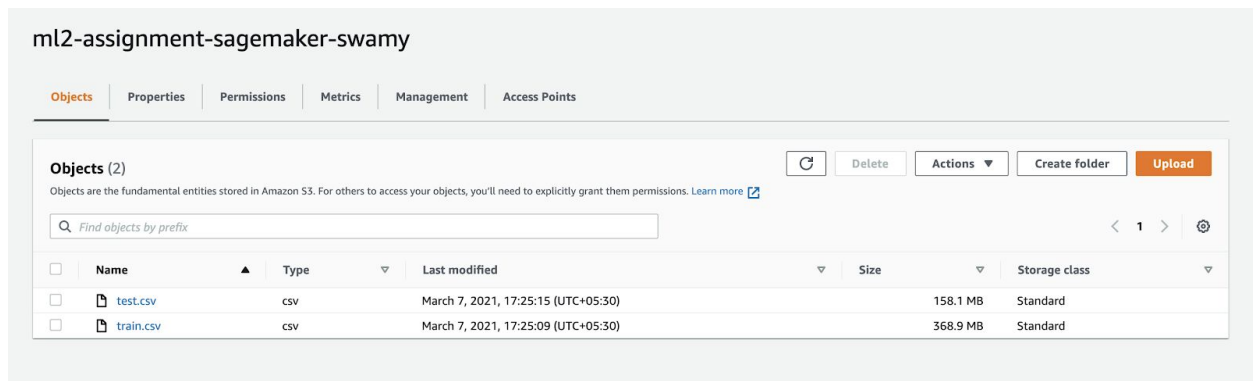
We choose one hot encoding for categorical values which does not have any order associated with them and all have equal weights.

However we should also note the fact that with one hot encoding the size of column multiplies to the number of values. Since we already have 66 independent variables, we also need to consider the size factor of the dataset. For this purpose only, we have reduced the number of values in all the columns as part of data cleaning.

There are 2 changes we need to do for the data to be SageMaker ready other than above encoding.

1. Have the target variable in the first column
2. Not have a header row
3. Remove any identifier column (here it is MachineIdentifier) and index columns

In order to connect to S3 we need to create an IAM role to access S3 and attach it to our EC2 instance. Only after that we can use boto3 to upload the files to S3



The screenshot shows the Amazon S3 console interface for a bucket named 'ml2-assignment-sagemaker-swamy'. The 'Objects' tab is active, showing a list of two objects: 'test.csv' and 'train.csv'. Both are CSV files, uploaded on March 7, 2021, at 17:25:15 (UTC+05:30) and 17:25:09 (UTC+05:30) respectively. The sizes are 158.1 MB and 368.9 MB, and both are stored in the 'Standard' storage class. The interface includes a search bar, a table with columns for Name, Type, Last modified, Size, and Storage class, and action buttons like 'Delete', 'Actions', 'Create folder', and 'Upload'.

	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	test.csv	csv	March 7, 2021, 17:25:15 (UTC+05:30)	158.1 MB	Standard
<input type="checkbox"/>	train.csv	csv	March 7, 2021, 17:25:09 (UTC+05:30)	368.9 MB	Standard

## Task 4: Model Training in SageMaker

Xgboost\_malware\_detection.ipynb file is attached. Below is screenshot of model being trained

```
In [5]: sess = sagemaker.Session()

xgb = sagemaker.estimator.Estimator(container,
                                     role,
                                     instance_count=1,
                                     instance_type='ml.m4.xlarge',
                                     output_path='s3://{}/{}'.format(bucket, prefix),
                                     sagemaker_session=sess)

xgb.set_hyperparameters(max_depth=8,
                        eta=0.36,
                        gamma=6.5,
                        min_child_weight=14,
                        subsample=0.85,
                        silent=0,
                        objective='binary:logistic',
                        num_round=800,
                        colsample_bytree=0.15,
                        eval_metric='auc')

xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})

[04:54:50] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 54 extra nodes, 136 pruned nodes, max_depth=8
[793]#011train-auc:0.772949#011validation-auc:0.738359
[04:54:51] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 180 pruned nodes, max_depth=8
[794]#011train-auc:0.772958#011validation-auc:0.738365
[04:54:52] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 78 extra nodes, 170 pruned nodes, max_depth=8
[795]#011train-auc:0.773019#011validation-auc:0.738363
[04:54:54] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 58 extra nodes, 162 pruned nodes, max_depth=8
[796]#011train-auc:0.773066#011validation-auc:0.73834
[04:54:54] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 76 pruned nodes, max_depth=0
[797]#011train-auc:0.773066#011validation-auc:0.73834
[04:54:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 76 extra nodes, 220 pruned nodes, max_depth=8
[798]#011train-auc:0.773138#011validation-auc:0.738298
[04:54:57] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46 extra nodes, 62 pruned nodes, max_depth=8
[799]#011train-auc:0.773175#011validation-auc:0.738298

2021-03-08 04:55:54 Uploading - Uploading generated training model
2021-03-08 04:55:54 Completed - Training job completed
Training seconds: 958
Billable seconds: 958
```

## Task 5: Model Inference and Performance

### AUC:

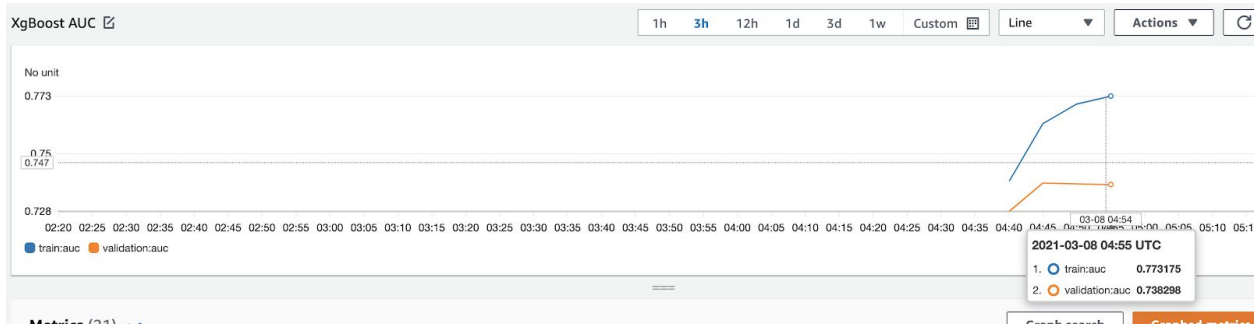
The model performed well on train and test data with an auc of 0.77 for train and 0.73 for test (here test data is the validation data). PFB screenshots from logs.

```
2021-03-08T10:24:52.891+05:30 [04:54:51] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 180 pruned nodes, max_depth=8
2021-03-08T10:24:52.891+05:30 [794]#011train-auc:0.772958#011validation-auc:0.738365
2021-03-08T10:24:53.892+05:30 [04:54:52] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 78 extra nodes, 170 pruned nodes, max_depth=8
2021-03-08T10:24:53.892+05:30 [795]#011train-auc:0.773019#011validation-auc:0.738363
2021-03-08T10:24:54.893+05:30 [04:54:54] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 58 extra nodes, 162 pruned nodes, max_depth=8
2021-03-08T10:24:54.893+05:30 [796]#011train-auc:0.773066#011validation-auc:0.73834
2021-03-08T10:24:55.893+05:30 [04:54:54] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra nodes, 76 pruned nodes, max_depth=0
2021-03-08T10:24:55.893+05:30 [797]#011train-auc:0.773066#011validation-auc:0.73834
2021-03-08T10:24:56.894+05:30 [04:54:55] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 76 extra nodes, 220 pruned nodes, max_depth=8
2021-03-08T10:24:56.894+05:30 [798]#011train-auc:0.773138#011validation-auc:0.738298
2021-03-08T10:24:57.894+05:30 [04:54:57] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46 extra nodes, 62 pruned nodes, max_depth=8
2021-03-08T10:24:57.894+05:30 [799]#011train-auc:0.773175#011validation-auc:0.738298

No newer events at this moment. Auto retrv naused. Resume
```



## ROC Curve:



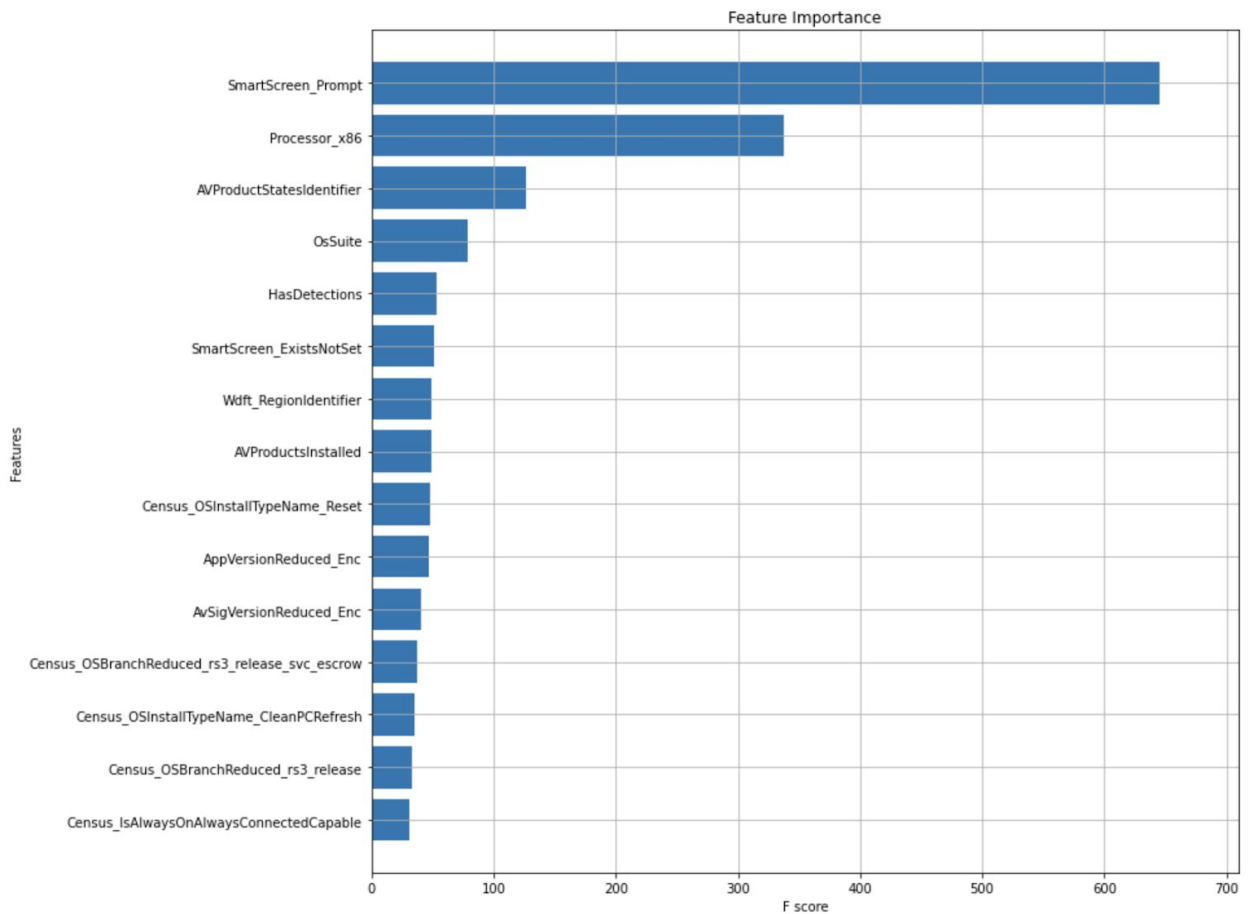
## Accuracy & F1 Score:

The Accuracy of the model is 0.67 and F1 score is 0.66 on the test data after making predictions with the model deployed on Sagemaker.

The above metrics are really good for a model with such a huge dataset and minimal resources.

## Feature Importance:

With the given code the top 15 features identified are as given below:



1. 'SmartScreen\_Prompt',
2. 'Processor\_x86',
3. 'AVProductStatesIdentifier',
4. 'OsSuite',
5. 'HasDetections',
6. 'SmartScreen\_ExistsNotSet',
7. 'Wdft\_RegionIdentifier',
8. 'AVProductsInstalled',
9. 'Census\_OSInstallTypeName\_Reset',
10. 'AppVersionReduced\_Enc',
11. 'AvSigVersionReduced\_Enc',
12. 'Census\_OSBranchReduced\_rs3\_release\_svc\_escrow',
13. 'Census\_OSInstallTypeName\_CleanPCRefresh',
14. 'Census\_OSBranchReduced\_rs3\_release',
15. 'Census\_IsAlwaysOnAlwaysConnectedCapable'

In the above list HasDetections is the target variable itself and hence we can leave it.

Looks like SmartScreen column with value Prompt has most influence on malware attacks followed by if the Processor is of type x86 and then followed by the antivirus product state.

### **Pandas Profiling:**

Profiling is done for 10 features. Note that some are different categorical values of the same feature. Profiling has reported its findings and also reported there are a number of duplicate rows.

For the next run we should validate if these are really duplicates and how they were added to the data. However for this assignment we can stop with this.

We got a real feel of the feature importance and they make sense since they point out the important features of the dataset like Processor, antivirus product states, number of antivirus products installed, and signature of the product and so forth. Looks like the availability of SmartScreen contributes to malware to a great extent. It can be further explored on how.