1.Problem Description

Among all operating systems Microsoft takes the 77% share in the market. Considering the inevitable growth of the online world security risk to customer data has increased. With increasing information sharing through the internet customer data is risked. These concerns include frauds online money stealing and information gathering etc... to address these concerns Malware industry has been working to find out the solutions to malware and protection from it for the computer. The best protection is to stay away from it.

The malware industry continues to be a well-organized, well-funded market dedicated to evading traditional security measures. Once a computer is infected by malware, criminals can hurt consumers and enterprises in many ways.

To enhance the security of almost 1billion users Microsoft is challenging the data science community for its customer security to bring up their best models to predict potential malware hits. Like our previous challenges we now again enable the Kaggle's data scientist to show off their skills and play a role for security of one billion users.

We must find out the patterns in malware signals and generate a possible predictor that would help save our customers from malware before even getting infected so it will be less harmful for the computer and customer`s data. we have got sample data based on which you have to design a model. Remember about data preparation steps.

The main goal of this competition is to predict a Windows machine's probability of getting infected by various families of malware, based on different properties of that machine

2.Data Description

The data set consists of mainly training and testing files and a sample submission. To stay in line with our goals the data was collected by combining heartbeat and threat reports from windows defender.

Each row in the test data is defined as properties of machine and ‘hasdetection’ column identifies if there were any detection on a machine with certain properties (columns). You must predict the ‘has detection’ column during testing.

Columns (83 total) as copied from original description from Kaggle are described as follows .

- **MachineIdentifier -** Individual machine ID
- **ProductName -** Defender state information e.g., win8defender
- **EngineVersion -** Defender state information e.g., 1.1.12603.0
- **AppVersion -** Defender state information e.g., 4.9.10586.0
- **AvSigVersion -** Defender state information e.g., 1.217.1014.0
- **IsBeta -** Defender state information e.g., false
- **RtpStateBitfield -** NA
- **IsSxsPassiveMode -** NA
- **DefaultBrowsersIdentifier -** ID for the machine's default browser
- **AVProductStatesIdentifier -** ID for the specific configuration of a user's antivirus software
- **AVProductsInstalled -** NA
- **AVProductsEnabled -** NA
- **HasTpm -** True if machine has tpm
- **CountryIdentifier -** ID for the country the machine is in
- **CityIdentifier -** ID for the city the machine is in
- **OrganizationIdentifier -** ID for the organization the machine belongs in, organization ID is mapped to both specific companies and broad industries
- **GeoNameIdentifier -** ID for the geographic region a machine is in
- **LocaleEnglishNameIdentifier -** English name of Locale ID of the current user
- **Platform -** Calculates platform name (of OS related properties and processor properties)
- **Processor -** This is the process architecture of the installed operating system

- **OsVer -** Version of the current operating system
- **OsBuild -** Build of the current operating system
- **OsSuite -** Product suite mask for the current operating system.
- **OsPlatformSubRelease -** Returns the OS Platform sub-release (Windows Vista, Windows 7, Windows 8, TH1, TH2)
- **OsBuildLab -** Build lab that generated the current OS. Example: 9600.17630.amd64fre.winblue_r7.150109-2022
- **SkuEdition -** The goal of this feature is to use the Product Type defined in the MSDN to map to a 'SKU-Edition' name that is useful in population reporting. The valid Product Type are defined in %sdxroot%\data\windowseditions.xml. This API has been used since Vista and Server 2008, so there are many Product Types that do not apply to Windows 10. The 'SKU-Edition' is a string value that is in one of three classes of results. The design must hand each class.
- **IsProtected -** This is a calculated field derived from the Spynet Report's AV Products field. Returns: a. TRUE if there is at least one active and up-to-date antivirus product running on this machine. b. FALSE if there is no active AV product on this machine, or if the AV is active, but is not receiving the latest updates. c. null if there are no Anti-Virus Products in the report. Returns: Whether a machine is protected.
- **AutoSampleOptIn -** This is the SubmitSamplesConsent value passed in from the service, available on CAMP 9+
- **PuaMode -** Pua Enabled mode from the service
- **SMode -** This field is set to true when the device is known to be in 'S Mode', as in, Windows 10 S mode, where only Microsoft Store apps can be installed
- **IeVerIdentifier -** NA
- **SmartScreen -** This is the SmartScreen enabled string value from registry. This is obtained by checking in order, HKLM\SOFTWARE\Policies\Microsoft\Windows\System\SmartScreenEnabled and HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SmartScreenEnabled. If the value exists but is blank, the value "ExistsNotSet" is sent in telemetry.
- **Firewall -** This attribute is true (1) for Windows 8.1 and above if windows firewall is enabled, as reported by the service.
- **UacLuaenable -** This attribute reports whether the "administrator in Admin Approval Mode" user type is disabled or enabled in UAC. The value reported is obtained by reading the regkey

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA.

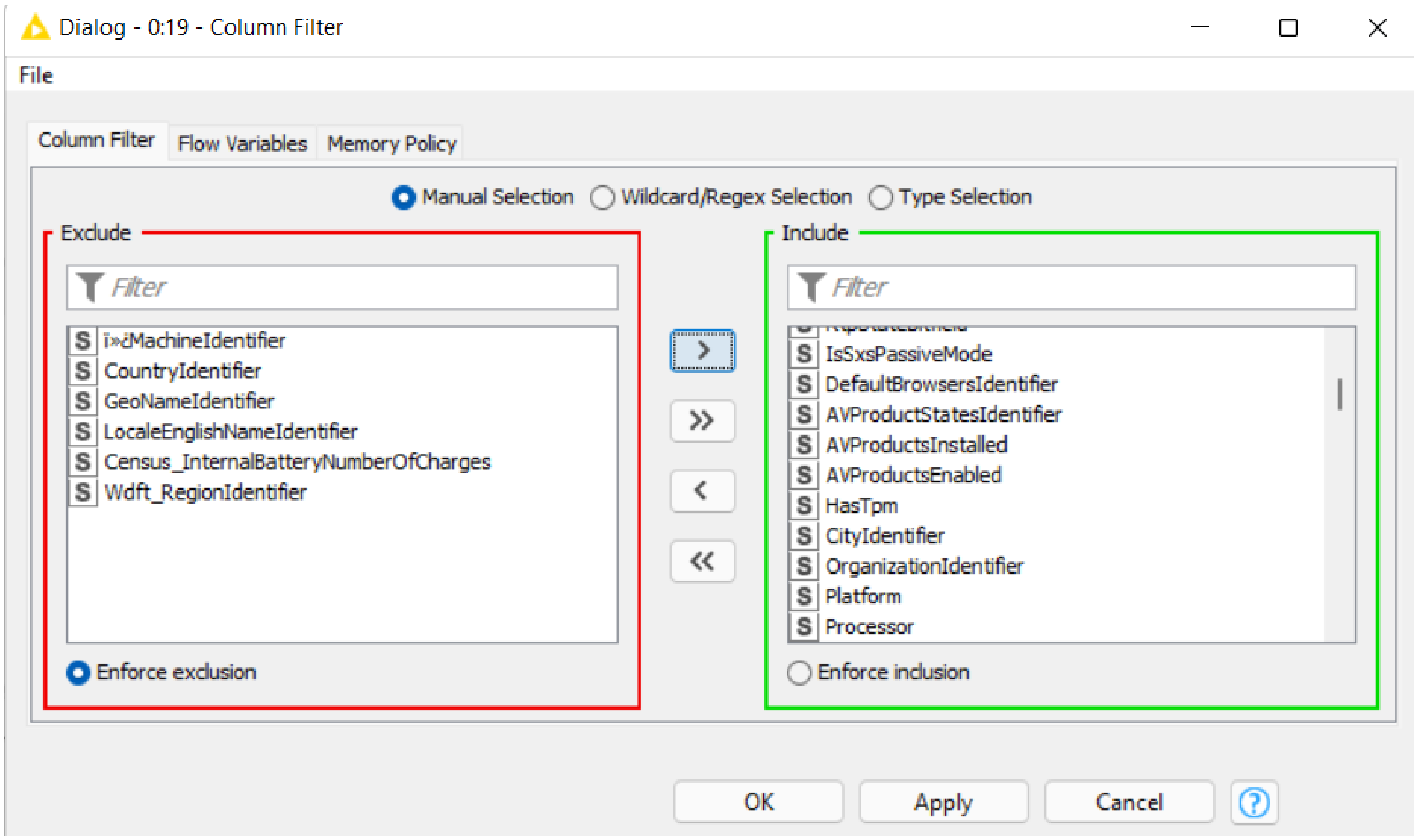
- **Census_MDC2FormFactor -** A grouping based on a combination of Device Census level hardware characteristics. The logic used to define Form Factor is rooted in business and industry standards and aligns with how people think about their device. (Examples: Smartphone, Small Tablet, All in One, Convertible...)
- **Census_DeviceFamily** - AKA DeviceClass. Indicates the type of device that an edition of the OS is intended for. Example values: Windows.Desktop, Windows.Mobile, and iOS.Phone
- **Census_OEMNameIdentifier** - NA
- **Census_OEMModelIdentifier** - NA
- **Census_ProcessorCoreCount -** Number of logical cores in the processor
- **Census_ProcessorManufacturerIdentifier** - NA
- **Census_ProcessorModelIdentifier** - NA
- **Census_ProcessorClass -** A classification of processors into high/medium/low. Initially used for Pricing Level SKU. No longer maintained and updated
- **Census_PrimaryDiskTotalCapacity** - Amount of disk space on primary disk of the machine in MB
- **Census_PrimaryDiskTypeName -** Friendly name of Primary Disk Type - HDD or SSD
- **Census_SystemVolumeTotalCapacity** - The size of the partition that the System volume is installed on in MB
- **Census_HasOpticalDiskDrive -** True indicates that the machine has an optical disk drive (CD/DVD)
- **Census_TotalPhysicalRAM -** Retrieves the physical RAM in MB
- **Census_ChassisTypeName -** Retrieves a numeric representation of what type of chassis the machine has. A value of 0 means xx
- **Census_InternalPrimaryDiagonalDisplaySizeInInches** - Retrieves the physical diagonal length in inches of the primary display
- **Census_InternalPrimaryDisplayResolutionHorizontal -** Retrieves the number of pixels in the horizontal direction of the internal display.
- **Census_InternalPrimaryDisplayResolutionVertical -** Retrieves the number of pixels in the vertical direction of the internal display
- **Census_PowerPlatformRoleName -** Indicates the OEM preferred power management profile. This value helps identify the basic form factor of the device
- **Census_InternalBatteryType -** NA

- **Census_InternalBatteryNumberOfCharges** - NA
- **Census_OSVersion -** Numeric OS version Example - 10.0.10130.0
- **Census_OSArchitecture** - Architecture on which the OS is based. Derived from OSVersionFull. Example - amd64
- **Census_OSBranch -** Branch of the OS extracted from the OsVersionFull. Example - OsBranch = fbl_partner_eeap where OsVersion = 6.4.9813.0.amd64fre.fbl_partner_eeap.140810-0005
- **Census_OSBuildNumber** - OS Build number extracted from the OsVersionFull. Example - OsBuildNumber = 10512 or 10240
- **Census_OSBuildRevision -** OS Build revision extracted from the OsVersionFull. Example - OsBuildRevision = 1000 or 16458
- **Census_OSEdition -** Edition of the current OS. Sourced from HKLM\Software\Microsoft\Windows NT\CurrentVersion@EditionID in registry. Example: Enterprise
- **Census_OSSkuName -** OS edition friendly name (currently Windows only)
- **Census_OSInstallTypeName -** Friendly description of what install was used on the machine i.e., clean
- **Census_OSInstallLanguageIdentifier** - NA
- **Census_OSUILocaleIdentifier** - NA
- **Census_OSWUAutoUpdateOptionsName -** Friendly name of the WindowsUpdate auto-update settings on the machine.
- **Census_IsPortableOperatingSystem -** Indicates whether OS is booted up and running via Windows-To-Go on a USB stick.
- **Census_GenuineStateName** - Friendly name of OSGenuineStateID. 0 = Genuine
- **Census_ActivationChannel -** Retail license key or Volume license key for a machine.
- **Census_IsFlightingInternal** - NA
- **Census_IsFlightsDisabled -** Indicates if the machine is participating in flighting.
- **Census_FlightRing -** The ring that the device user would like to receive flights for. This might be different from the ring of the OS which is currently installed if the user changes the ring after getting a flight from a different ring.
- **Census_ThresholdOptIn -** NA
- **Census_FirmwareManufacturerIdentifier** - NA
- **Census_FirmwareVersionIdentifier** - NA
- **Census_IsSecureBootEnabled -** Indicates if Secure Boot mode is enabled.
- **Census_IsWIMBootEnabled** - NA

- **Census_IsVirtualDevice -** Identifies a Virtual Machine (machine learning model)
- **Census_IsTouchEnabled -** Is this a touch device ?
- **Census_IsPenCapable -** Is the device capable of pen input ?
- **Census_IsAlwaysOnAlwaysConnectedCapable -** Retreives information about whether the battery enables the device to be AlwaysOnAlwaysConnected .
- **Wdft_IsGamer -** Indicates whether the device is a gamer device or not based on its hardware combination.
- **Wdft_RegionIdentifier -** NA

3.DATA PRE-PROCESSING AND ANALYZATION

The given data is the details properties of machine , On Kaggle data is divided into train(4.3gb)having 89 million rows and test(3.8gb) with 83 columns. Our machines were not capable of processing that much of big data therefore we must prepare data. From the given dataset of 89 million rows of train data we extract 200k rows to train our model and for test we take 50k rows from the given test data on Kaggle. Details of every column of the dataset which is on Kaggle.
Data contains the 83 property of machines which can be helpful in prediction of potential future malware. But we haven't used every column for model training, there are some columns which we think are not very much playing part in predictions with the help of **column filter**.
Firstly, with the help of column filter and by analyzing data we remove the above 7 columns which reduce the column size from 83 to 76 .Machine identifier is unique machine id which does not have any impact on **accuracy** .next is the country and genome identifier which is the country location of the machine more local English name of the machine and the region does not have any significant effect on the explanation or the accuracy of the model.
Correlation graphs help us understand the relation between variables and if two variables are strongly correlated, we need to remove them



Dialog - 0:19 - Column Filter

File

Column Filter | Flow Variables | Memory Policy

[✓ Manual Selection] Wildcard/Regex Selection Type Selection

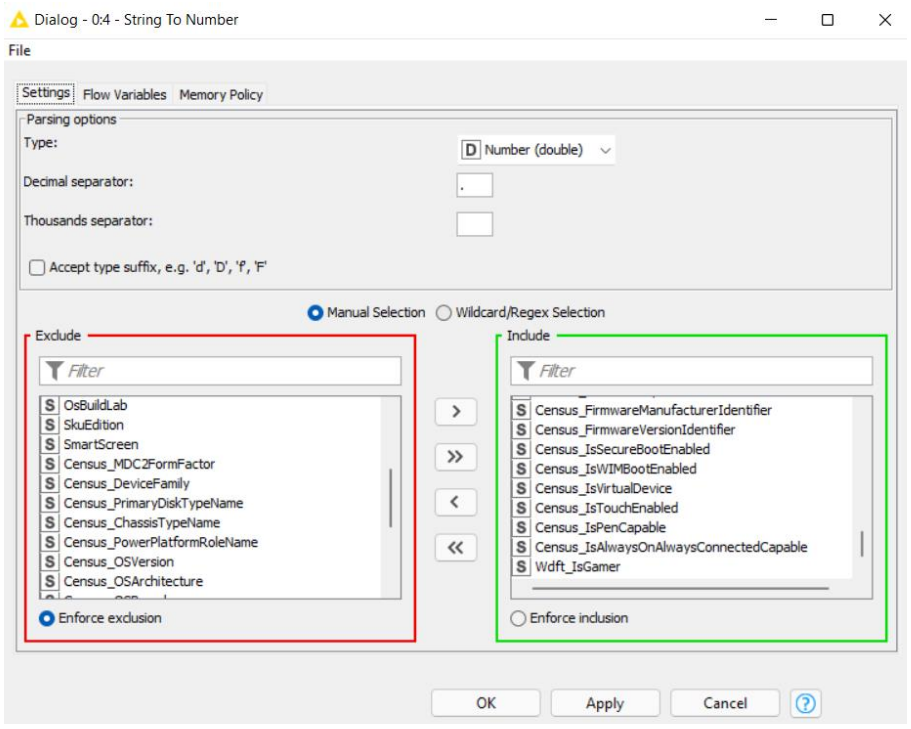
Exclude:
- ï»¿MachineIdentifier
- CountryIdentifier
- GeoNameIdentifier
- LocaleEnglishNameIdentifier
- Census_InternalBatteryNumberOfCharges
- Wdft_RegionIdentifier

[✓ Enforce exclusion]

Include:
- IsSxsPassiveMode
- DefaultBrowsersIdentifier
- AVProductStatesIdentifier
- AVProductsInstalled
- AVProductsEnabled
- HasTpm
- CityIdentifier
- OrganizationIdentifier
- Platform
- Processor

Enforce inclusion

OK | Apply | Cancel



Dialog - 0:4 - String To Number

File

Settings | Flow Variables | Memory Policy

Parsing options

Type: Number (double)

Decimal separator: .

Thousands separator:

Accept type suffix, e.g. 'd', 'D', 'f', 'F'

[✓ Manual Selection] Wildcard/Regex Selection

Exclude:
- OsBuildLab
- SkuEdition
- SmartScreen
- Census_MDC2FormFactor
- Census_DeviceFamily
- Census_PrimaryDiskTypeName
- Census_ChassisTypeName
- Census_PowerPlatformRoleName
- Census_OSVersion
- Census_OSArchitecture

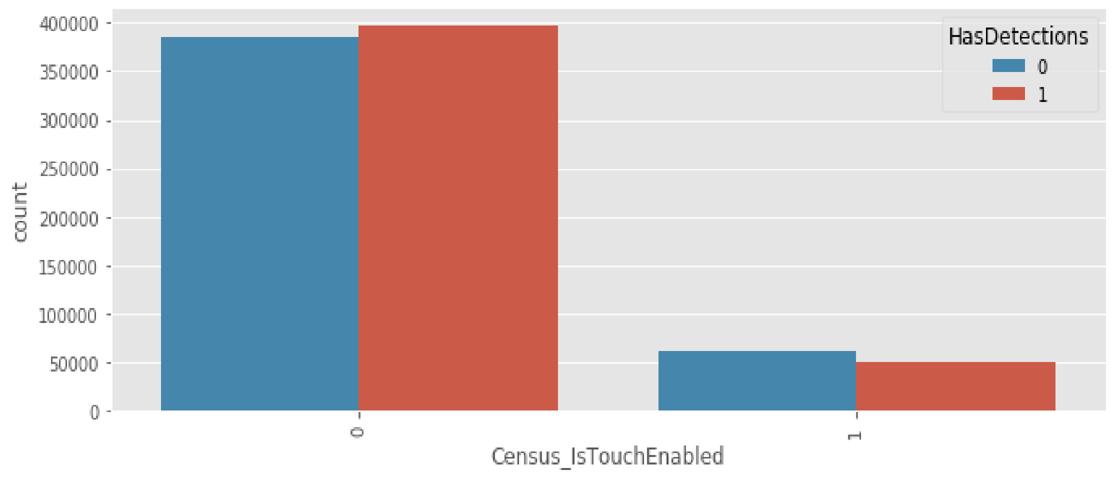
[✓ Enforce exclusion]

Include:
- Census_FirmwareManufacturerIdentifier
- Census_FirmwareVersionIdentifier
- Census_IsSecureBootEnabled
- Census_IsWIMBootEnabled
- Census_IsVirtualDevice
- Census_IsTouchEnabled
- Census_IsPenCapable
- Census_IsAlwaysOnAlwaysConnectedCapable
- Wdft_IsGamer

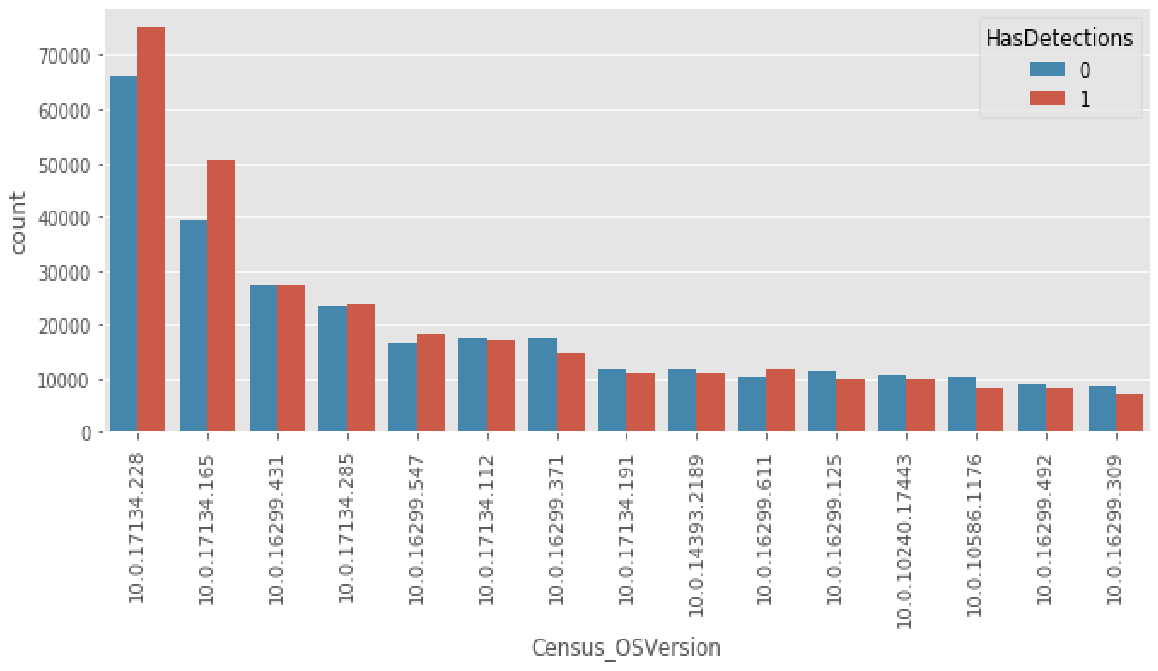
Enforce inclusion

OK | Apply | Cancel

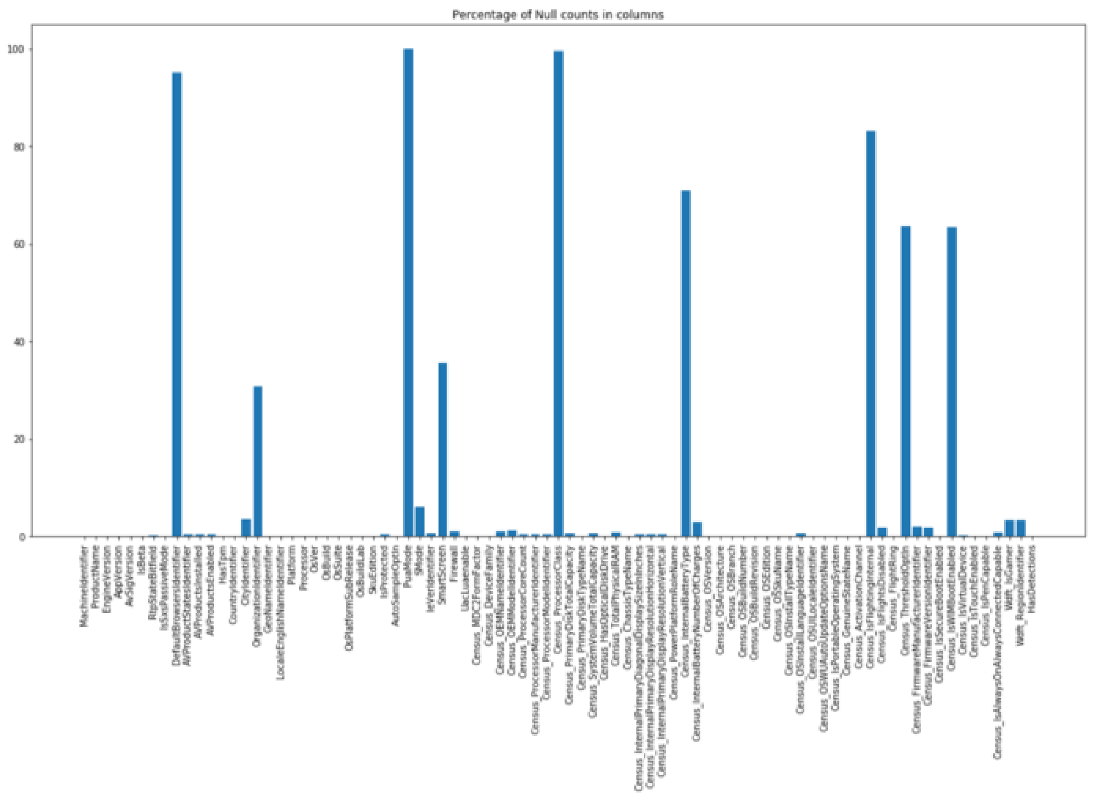
Another thing which we notice in dataset is the all the categories that is column are of type string however there are values are numeric therefore we use string to number node and carefully converted many columns to numeric datatype



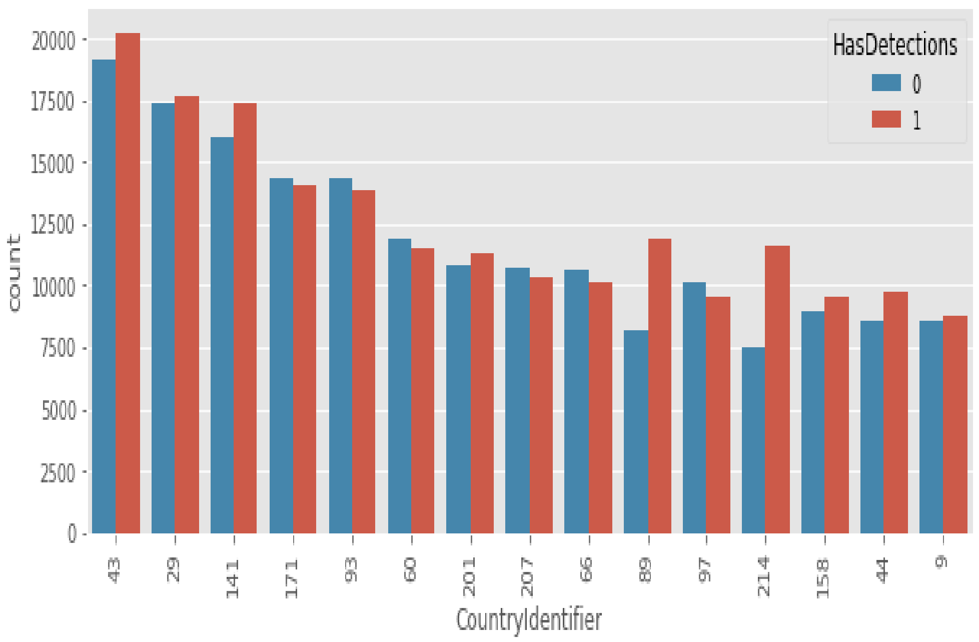
This shows that the device which have touch enabled have lower risk of infection to malware



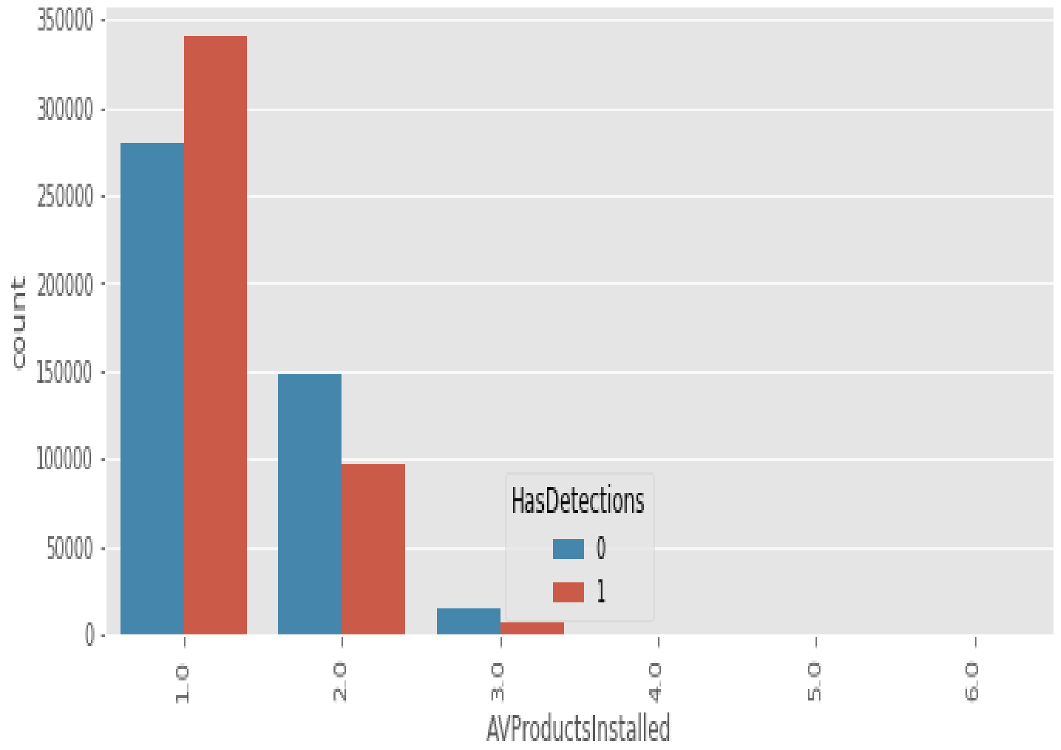
From the histogram we can see that machine’s malware detection defers largely based on operating system version



Here we find out the percentage of null values in our columns we removed those with the pre-decided percentages



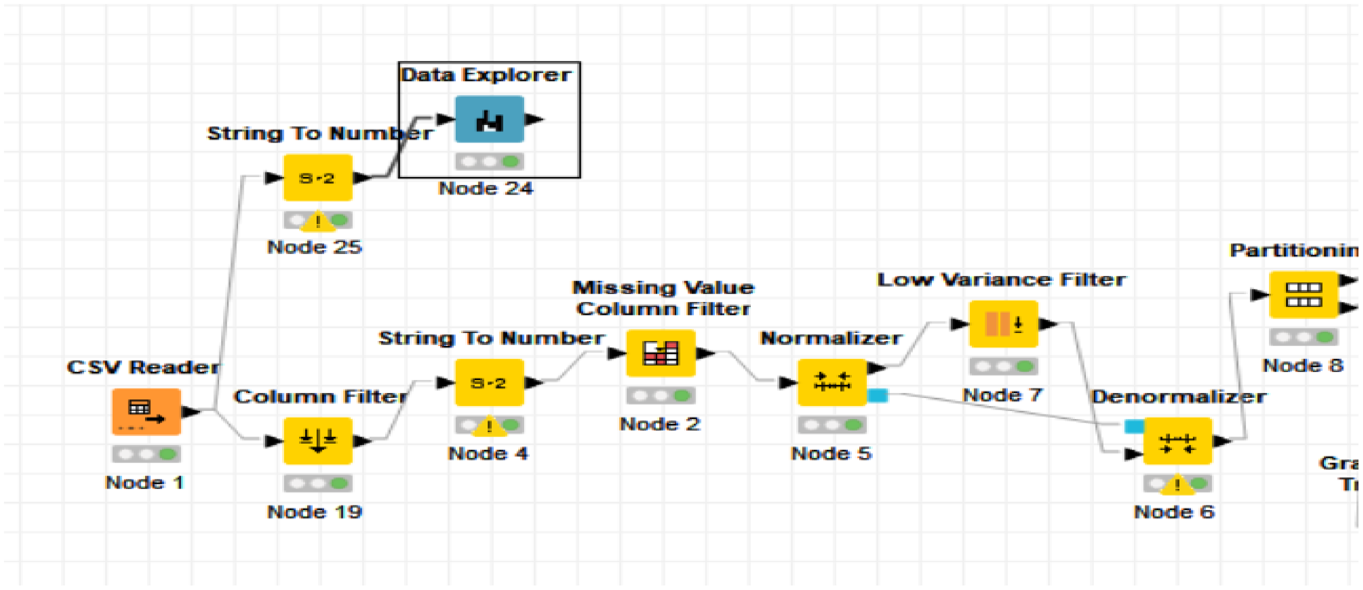
This shows that the location at which machine reside is also important in malware detection because country with code 43 have highest detections



Similarly, installation of any type of antivirus is also play an important role. Graph shows that machines with 1 antivirus software shows high count of deductions as compared to machines with 3 installed antivirus software

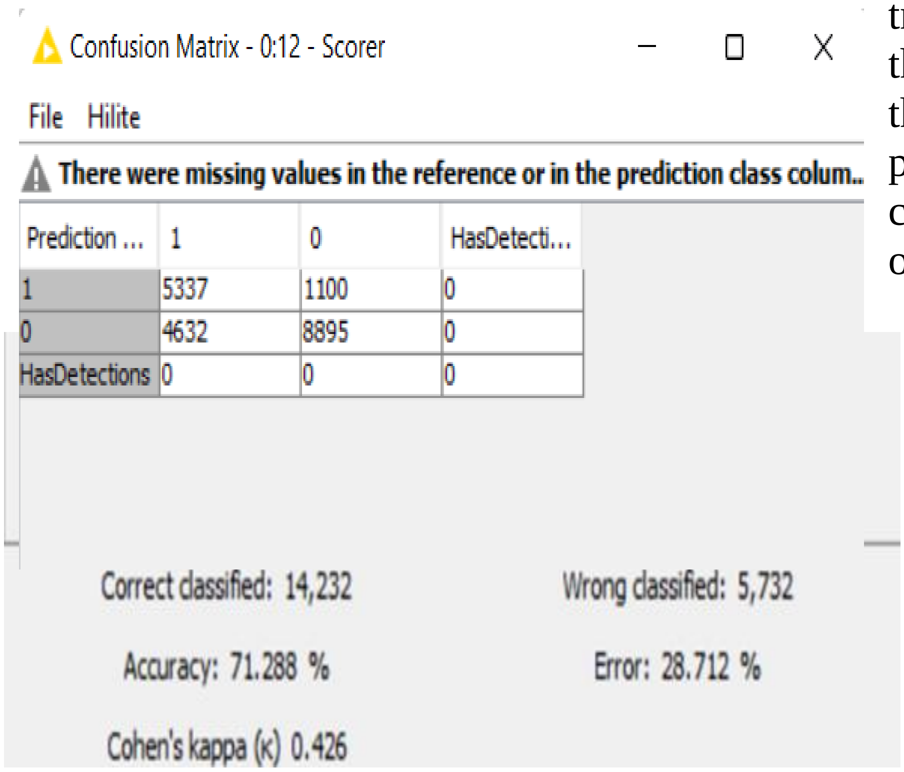
secondly some column shows a large **variance,** and some have very few therefore we must have removed low variance column to get the perfect data for modelling .And to apply **low variance filter** we must normalize the data first and then applied the bound of 0.01 variance

WORKFLOW DIAGRAM



4. DATA MODELLING

Previously we have prepared the data according to our problem with the help of data exploration now firstly we have to portion our data into training and testing therefore we can train the data for this we have put the default configuration that is ratio of 30% to 70% for testing and training respectively. Moreover, as we are not modeling with full training or testing data therefore, we cannot test our model on Kaggle therefore we have to rely on


Confusion Matrix - 0:12 - Scorer

File Hilite

There were missing values in the reference or in the prediction class colum...

Prediction ...	1	0	HasDetecti...
1	5337	1100	0
0	4632	8895	0
HasDetections	0	0	0

Correct classified: 14,232

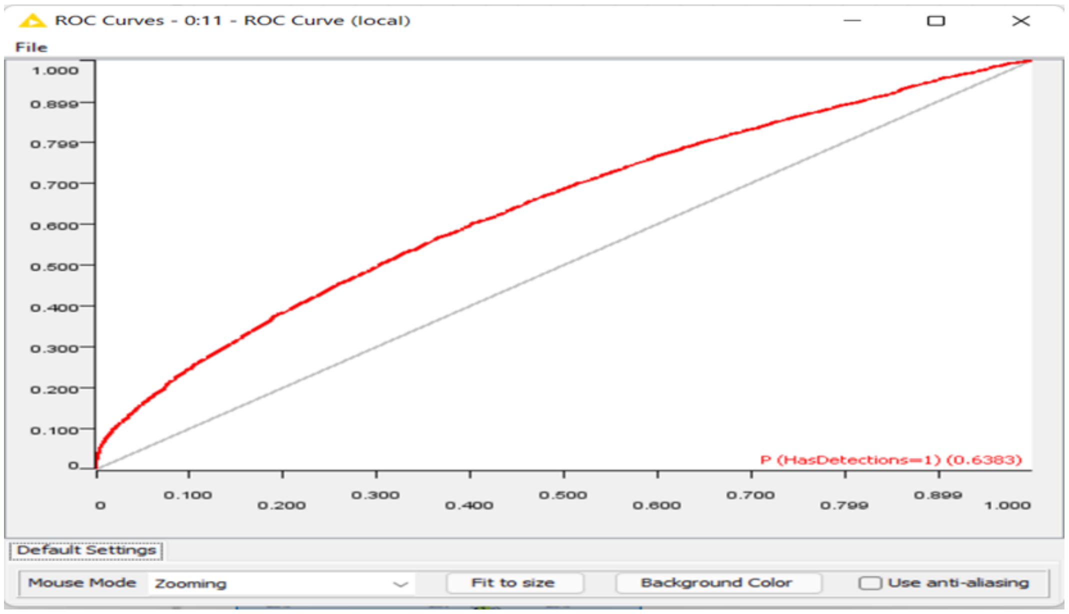
Wrong classified: 5,732

Accuracy: 71.288 %

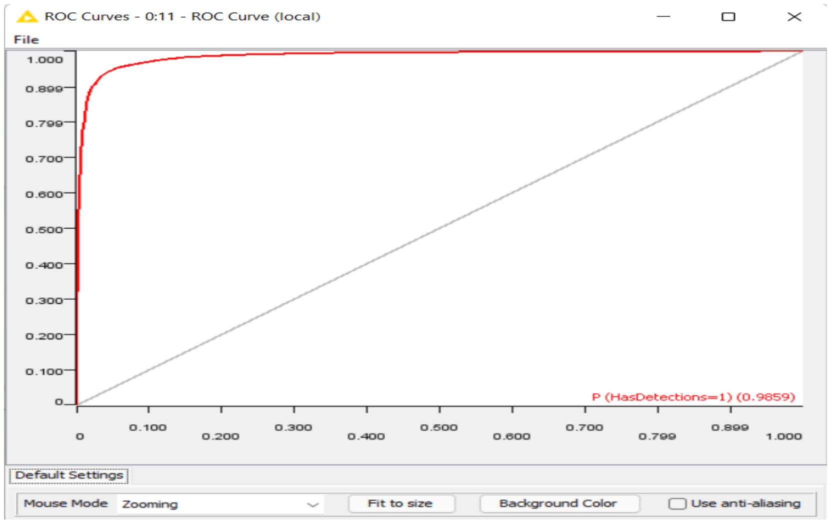
Error: 28.712 %

Cohen's kappa (κ) 0.426

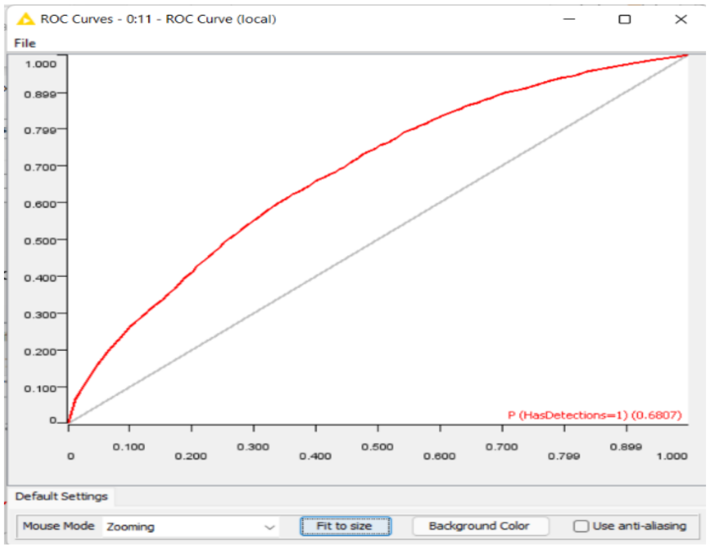
accuracy of training that is of roc curve. firstly, modeled our data on **decision tree** and predict the accuracy this shows the **accuracy of 71.2%** with Gini index however we believe that as the data is not always binary and have multiclass therefore information gain ratio will work well but the result shows different case the accuracy of decision tree with Gini index is higher therefore for further modelling, we use Gini-index only.

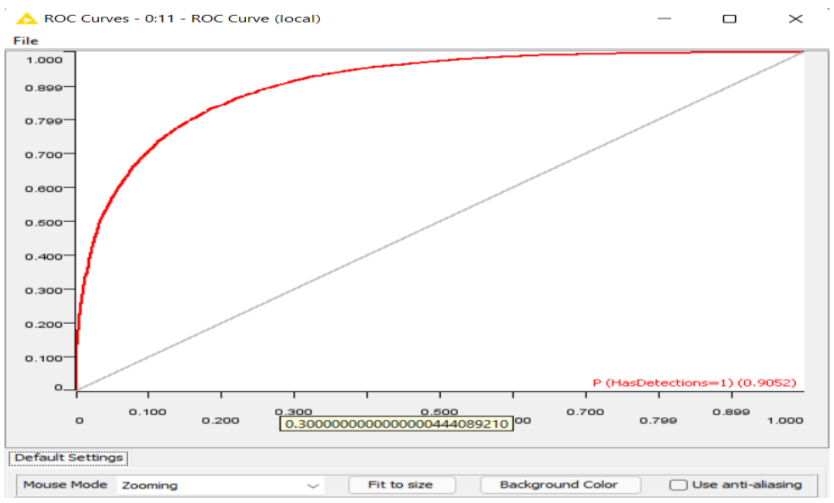


After decision tree we move towards **naïve bayes** so that we can decide that we must move forward with tree like algorithms or other with naïve bayes default configuration the accuracy tends to decrease and got the prediction with **accuracy of 63%** as the size of data is huge with larger number of multiclass columns the naïve bayes unable to predict with good accuracy this shows that further we must move with only tree algorithms.



Moreover we then use **random forest** with 10 models first and glad to see rise in **the accuracy from 71 of decision tree to 89%** with random forest we than rerun the model with the increase of models to 100 and predict the **accuracy of 98%(due to same test and training dataset)** this is not common and still we cannot say anything because this accuracy is still with training data set but this seems to be overfitting therefore we decide to change the configuration hence we limit the depth of the tree to 7 and got the best accuracy of random forest that is of **68%.**





After serval tries, we decide to move towards gradient boosting as previously working with this type of data **gradient boosting** performs good, after running this model we get to know that gradient boosting performs good in this case also Boosting works well because it builds models intelligently by giving more and more weight to observations that are hard to classify as compared to random forest however the risk of overfitting also exist. In gradient boosting maximum accuracy occur with 0.2 learning rate of 6 tree depth with 1000 models this cause the accuracy to reach 90.5%.

5.FINDINGS

- Gradient Boosting proved to be the best algorithm for prediction of this data set
- Strongly Correlated variables may cause redundancy as they do not contain extra information.
- There is always a tradeoff when we choose different machine learning algorithms to carry out our training and testing
- Pre-processing data is very important in all cases as Data algorithm would have performed vey badly if these steps were omitted.
- Algorithm with higher runtime perform better in most cases. As most time took by Gradient boosting. And random forest also took more time than naive Bayes hence higher accuracy.
- Hyper-parameter like learning rate/Tree depth effect the performance of the algorithm.

- the number of new malware samples has explosively increased so an intelligent prediction method should be developed to combat the changes and data collection should be increased
- Data collection should be done on daily bases to increase the effectiveness of the algorithm
- we would suggest that in future data collections Microsoft should focus on preventing data redundancy like in this dataset there is separate columns of city name country name geo address region-name whereas they can just replace it with geo address to get all other details
- Malware detection approaches are divided into two main categories that include behavior-based and signature-based methods.
- Signature methods are used to identify Known malware whereas behavioral predicts based on activities. This project is based on behavioral.