ABSTRACT:

A real world dataset usually have lots of features and it is difficult to visualize each and every entity in the dataset and also a very challenging task as well as computational based and as in person. Having a huge number of features is beneficial and also is a curse, beneficial as in we have lots of information with us and challenging when dealing with them. So, to reduce the reduce the computational efficiency and as well as to improve the accuracy we have many dimension reduction techniques which help us a lot by selecting important features and eliminating the features with most missing percentage, low variance, high correlation, etc. PCA is applied for dimension reduction and the feature importance model from RandomForestRegressor is used for feature selection and the results are compared before and after applying these techniques**.**

INTRODUCTION:

If you're dealing with a dataset with huge number of features or attributes or variables you might face some problems in computation efficiency or overfitting of data or also sometimes you will not get good accuracy. This is where dimension reduction comes in, that is if you having a dataset which is 3D, converting this dataset to 2D for easy visualization and for other useful purposes is dimension reduction. Having a huge number of features is beneficial and also is a curse, beneficial as in we have lots of information with us and challenging when dealing with them. So, to reduce the reduce the computational efficiency and as well as to improve the accuracy we have many dimension reduction techniques which help us a lot by selecting important features and eliminating the features with most missing percentage, low variance, high correlation, etc. PCA is applied for dimension reduction and the feature importance model from RandomForestRegressor is used for feature selection and the results are compared before and after applying these techniques.

LITERATURE SURVEY:

[1] Kernel Principal Component Analysis (KPCA) is a technique widely used to understand and visualize non-linear variation patterns by inverse mapping the projected data from a high-dimensional feature space back to the original input space. Variation patterns often occur in a small number of relevant features out of the overall set of features that are recorded in the data. It is, therefore, crucial to discern this set of relevant features that define the pattern. Here we propose a feature selection procedure that augments KPCA to obtain importance estimates of the features given the noisy training data. Our feature selection strategy involves projecting the data points onto sparse random vectors for calculating the kernel matrix. We then match pairs of such projections, and determine the preimages of the data with and without a feature, thereby trying to identify the importance of that feature. Thus, preimages' differences within pairs are used to identify the relevant features. An advantage of our method is it can be used with any suitable KPCA algorithm. Moreover, the computations can be parallelized easily leading to significant speedup. We demonstrate our method on several simulated and real data sets, and compare the results to alternative approaches in the literature.

[2]Ross Goroshin, Jonathan Tompson, Debidatta Dwibedi Fully convolutional deep correlation networks are integral components of state-of the-art approaches to single object visual tracking. It is commonly assumed that these networks perform tracking by detection by matching features of the object instance with features of the entire frame. Strong architectural priors and conditioning on the object representation is thought to encourage this tracking strategy. Despite these strong priors, we show that deep trackers often default to "tracking by saliency" detection– without relying on the object instance representation. Our analysis shows that despite being a useful prior, salience detection can prevent the emergence of more robust tracking strategies in deep networks. This leads us to introduce an auxiliary detection task that encourages more discriminative object representations that improve tracking performance.

[3]This paper by Lindsay I Smith, is for understanding the design of Principle component analysis(PCA). PCA is found as a useful technique for applications like image classification and for reducing dimensions in high dimension dataset. The author has also given some basic introduction on the maths skills required to learn the process of Principle Component Analysis. Properties that are important from matrix algebra the important fundamental to PCA like eigenvector and eigenvalues are discussed.

[4] Gender classification is one of the most challenging problems in computer vision. Facial gender detection of neonates and children is also known as a highly demanding issue for human observers. This study proposes a novel gender classification method using frontal facial images of people. The proposed approach employs principal

component analysis (PCA) and fuzzy clustering technique, respectively, for feature extraction and classification steps. In other words, PCA is applied to extract the most appropriate features from images as well as reducing the dimensionality of data. The extracted features are then used to assign the new images to appropriate classes - male or female - based on fuzzy clustering. The computational time and accuracy of the proposed method are examined together and the prominence of the proposed approach compared to most of the other well-known competing methods is proved, especially for younger faces. Experimental results indicate the considerable classification accuracies which have been acquired for FG-Net, Stanford and FERET databases. Meanwhile, since the proposed algorithm is relatively straightforward, its computational time is reasonable and often less than the other state-of-the-art gender classification methods.

[5] Principal component analysis (PCA) has been widely applied in the area of computer science. It is wellknown that PCA is a popular transform method and the transform result is not directly related to a sole feature component of the original sample. However, in this paper, we try to apply principal components analysis (PCA) to feature selection. The proposed method well addresses the feature selection issue, from a viewpoint of numerical analysis. The analysis clearly shows that PCA has the potential to perform feature selection and is able to select a number of important individuals from all the feature components. Our method assumes that different feature components of original samples have different effects on feature extraction result and exploits the eigenvectors of the covariance matrix of PCA to evaluate the significance of each feature component of the original sample. When evaluating the significance of the feature components, the proposed method takes a number of eigenvectors into account. Then it uses a reasonable scheme to perform feature selection. The devised algorithm is not only subject to the nature of PCA but also computationally efficient. The experimental results on face recognition show that when the proposed method is able to greatly reduce the dimensionality of the original samples, it also does not bring the decrease in the recognition accuracy.

[6]Annie George Anomaly detection has emerged as an important technique in many application areas mainly for network security. Anomaly detection based on machine learning algorithms considered as the classification problem on the network data has been presented here. Dimensionality reduction and classification algorithms are explored and evaluated using KDD99 dataset for network IDS. Principal Component Analysis for dimensionality reduction and Support Vector Machine for classification have been considered for the application on network data and the results are analysed. The result shows the decrease in execution time for the classification as we reduce the dimension of the input data and also the precision and recall parameter values of the classification algorithm shows that the SVM with PCA method is more accurate as the number of misclassification decreases.

[7]Shweta Saini ,Sugandha Sharma,Rupinder Singh Software metrics play an important role in Software Development Life Cycle (SDLC). In this paper we have tried to find the correlation between different software metrics. The research is done using PROMISE data repository of NASA. The positive correlation is found to exist between various software metrics. Large number of metrics available in the software industry raises the need of finding the most significant metrics for better use and control of metrics. For this, the Principal Component Analysis (PCA) based feature selection has been applied on the correlated metrics data set. Key Words: Software metrics, correlation, feature selection.

[8]The effect of E-beam irradiation on cooked and dry cured Iberian ham, minced meat, smoked salmon and soft cheese, which have different chemical compositions with respect to protein, fat, moisture, free amino acids, amino acid decomposition products and preservatives intentionally added (nitrate and nitrite), was evaluated. A decrease up to 50% in fat content was observed with the irradiation increase for cooked ham and smoked salmon. Protein was modified only in cooked ham samples, while free amino acid amounts were significantly affected in all cases. Nitrate and nitrite content were affected in cooked and Iberian ham, with losses up to 100%, and in smoked salmon. Principal component analysis (PCA) was used to model the data. The results obtained made it possible to establish that radiation doses of 6 and 8 kGy produce chemical composition changes in practically every foodstuff. At the radiation of 2 kGy, which is the dose required to reach the food safety objective (FSO), dry cured Iberian ham was the least affected food, whereas minced meat and cooked ham were the most affected. Stepwise multiple-linear regression (MLR) explained changes in nitrate and nitrite content with the radiation dose applied and the food chemical composition (especially with moisture, fat and protein).

[9]Young-Jun Yoo's. Fault detection and diagnosis of the induction motor is important to prevent the system downtime of industrial fields. Most of the fault detection and diagnosis is conducted in the frequency domain using fast Fourier transform (FFT). Although several studies have been done using FFT, this method has difficulties in finding the fault characteristic frequency component. To overcome these difficulties, this paper provides the algorithm using principal component analysis (PCA) to easily find the feature of the FFT signal and utilize the Hotelling's T2 as an index for fault detection. After selecting the peak of top five frequencies and corresponding amplitude of the FFT as a feature and reducing the dimension through PCA, it is possible to detect a motor abnormality through Hotelling's T2 value. The proposed method is verified for detecting abnormal states of three-phase squirrel-cage induction motor. It

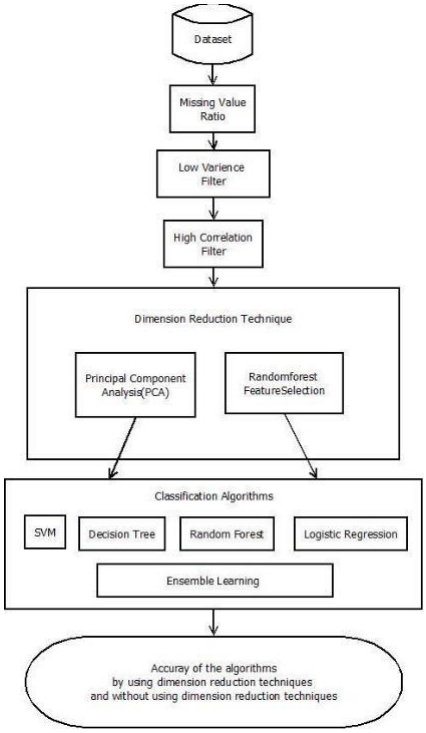
has been confirmed that using the fault characteristic frequency component of both frequency and corresponding amplitude is more accurate in determining the motor abnormality than the characteristic of only frequency.

[10]Liton Chandra Paul, Abdulla Al Sumam, This paper mainly addresses the building of face recognition system by using Principal Component Analysis (PCA). PCA is a statistical approach used for reducing the number of variables in face recognition. In PCA, every image in the training set is represented as a linear combination of weighted eigenvectors called eigenfaces. These eigenvectors are obtained from covariance matrix of a training image set. The weights are found out after selecting a set of most relevant Eigenfaces. Recognition is performed by projecting a test image onto the subspace spanned by the eigenfaces and then classification is done by measuring minimum Euclidean distance. A number of experiments were done to evaluate the performance of the face recognition system. In this thesis, we used a training database of students of Electronics and Telecommunication Engineering department, Batch-2007,
Rajshahi University of Engineering and Technology, Bangladesh. Index Terms—PCA, Eigenvalue, Eigenvector, Covariance, Euclidean distance, Eigenface.

[11]In this paper by Ronny Luss · Alexandre d'Aspremont, PCA is studied to grouping and highlight determination issues. SparsePCA looks for inadequate factors, or straight blends of the information factors, clarifying a most extreme sum of difference in the information while having no of nonzero coefficient that are foreordained. PCA is frequently utilized as a basic grouping strategy and inadequate variables permit us here to decipher the bunches as far as a decreased arrangement of factors. We start with a brief presentation and inspiration on inadequate PCA. They at that point apply these results to some exemplary bunching and highlight choice issues emerging in science.

[12]Manisha Satone, Gajanan Kharate, Many events, such as terrorist attacks, exposed serious weaknesses in most sophisticated security systems. So it is necessary to improve security data systems based on the body or behavioral characteristics, called biometrics. With the growing interest in the development of human and computer interface and biometric identification, human face recognition has become an active research area. Face recognition offers several advantages over other biometric methods. Nowadays Principal Component Analysis (PCA) has been widely adopted for the face recognition algorithm. Yet still, PCA has limitations such as poor discriminatory power and large computational load. This paper proposed a novel algorithm for face recognition in which a low frequency component of the wavelet is used for PCA representation. Best features of PCA are selected using the genetic algorithm (GA). Support vector machine (SVM) and nearest neighbor classifier (ND) are used for classification. Classification accuracy is examined by changing number of training images, number of features and kernel function. Results are evaluated on ORL, FERET, Yale and YaleB databases. Experiments showed that proposed method gives a better recognition rate than other popular methods.

ARCHITECTURE DESIGN:



MODULE DESCRIPTION:

Missing Value Ratio:

The missing value percentage of every feature in our dataset is calculated. Then we set a threshold percentage value, the missing value percentage of each feature is compared with the threshold percentage value. If the missing percentage value of a feature is greater than the threshold percentage value then the feature is eliminated.

Low Variance Filter:

The first step here is to fill up the missing values in every feature after there should be no null values in our dataset. As we know what variance is i.e. the difference between the values and the average of those values. Here we eliminate the feature with very low variance.

High Correlation Filter:

If the correlation between one feature with other feature is high that means they can have similarity in their information. Logistic Regression's performance can be affected by this. We can calculate the correlation between the numerical features and set a threshold value, if the correlation value is higher than the threshold value, we can eliminate one if the feature and selecting one and eliminating one feature is done by what domain you're using.

Principal Component Analysis:

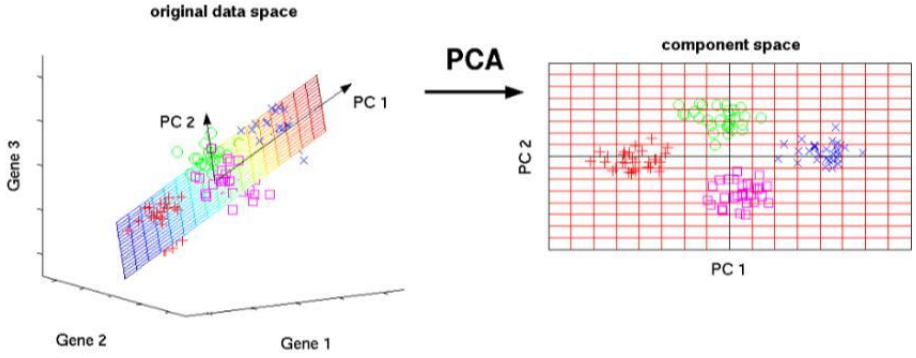
A principal component is a linear combination of the original variables

- Principal components are extracted in such a way that the first principal component explains maximum variance in the dataset
- Second principal component tries to explain the remaining variance in the dataset and is uncorrelated to the first principal component
- Third principal component tries to explain the variance which is not explained by the first two principal components and so on

The main purpose of a principal component analysis is the analysis of data to identify and find patterns to reduce the dimensions of the dataset with a minimal loss of information.

- 'N' dimension of data.
- Obtain the Covariance matrix.
- Calculate 'N' Eigen values and Eigen vectors.
- Order the Eigen vectors based on the Eigen values.
- The Eigen vector with the highest Eigen value is the principal components of the given data

set. Thus, the given 'N' dimension of data has been reduced into _p' principal components finally.



RandomForest:

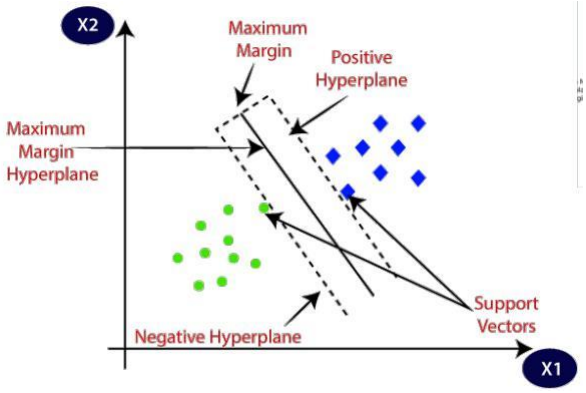
Random Forest is one of the algorithms most commonly used in function discovery. It comes bundled with the value of the in-built function and you don't need to configure it separately. This assists us in choosing a smaller subset of variables.

In RandomForestRegressor there is a module called feature_importances_ which helps us by selecting important features in our dataset and we can also visualise the Feature Importance by plotting a graph.

Support Vector Machine (SVM):

SVM are supervised learning models which are widely used by many and a popular algorithm for regression and classification analysis. The objective of the SVM algorithm is to create a hyperplane in a n-D space(n — the number of features) that unmistakably characterizes the data points. If you have x dimension data then the hyperplane will be of n-1 dimension.

There are several alternative hyperplanes which could be used to distinguish the two types of data points. Our target is to find a plane with the greatest range, that is to say the maximum distance between the data points in both groups. Maximizing the gap from the margins offers some stabilization so that potential data points can be more easily categorized.



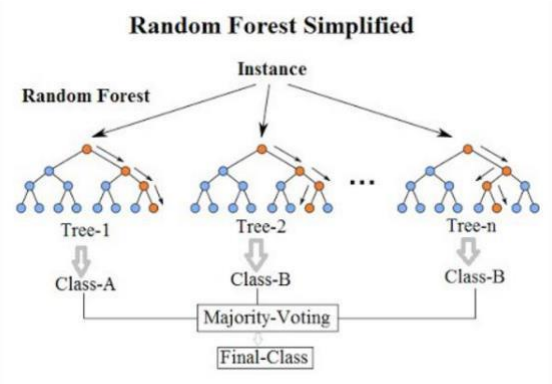
If you have n dimension data then the hyperplane will be of n-1 dimension.

Decision Tree:

It is a model that presents classification or regression as a tree. Hence, the tree is created and at the last level, the outcome is revealed. The leaves are the class labels and the conditions or the conjunctions are the branches. Hence, DT isn't delicate to noise.

RandomForest:

RandomForest first creates small subsets of different features of the original dataset and the creates the tree for every subset for classification it gets all the results of the sub trees and the final result is the max number of votes for a particular class. In case of regression final result is the average of all the results given by the sub trees.

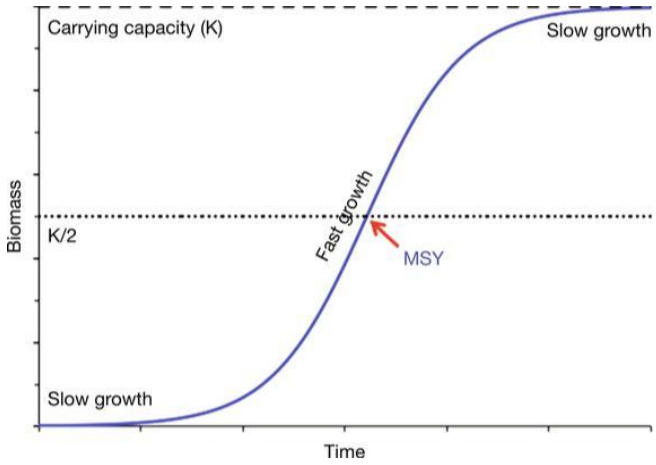


Logistic regression:

Logistic regression is a statistical technique used in the field of machine learning. It is used for binary classification. The output should be categorical in this method. Logistic regression uses Logistic function to classify data, hence the name Logistic Regression. The logistic function, also known as the sigmoid function is equated as follows:

1 / (1 + e^-P)

Where e represents Euler's number and P represents Real Value. It is an S-shaped curve, that takes any real value and maps it between 0 and 1.



Logistic regression is very similar to linear regression in terms of the equation. It takes one or more input parameters, parametrizes with weight coefficients, and computes output. The logistic regression equation is represented as

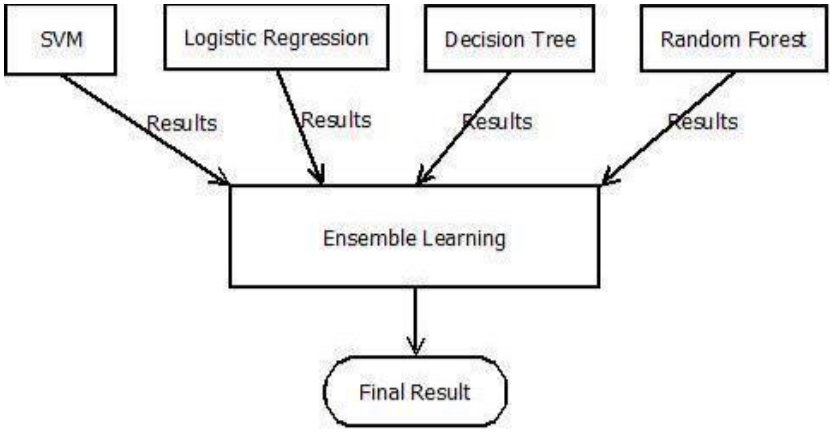
y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x)).

Here, e represents Euler's Number, b0 and b1 are the weights that control the model. x is the input vector and y is the predicted output value

Logistic regression is preferred over linear regression because linear regression needs a threshold to predict values and the output would be any continuous value. In other words, linear regression is unbounded. When the output is categorical (Example: yes or no), logistic regression is preferred.

Ensemble Learning:

In here we are using SVM, Logistic Regression, Decision tree, Random Forest algorithms for classification and what ensemble learning does is gets the results of all the algorithms and calculates the votes for each class and results the class with the highest vote.



WORKING:

Dataset:

The we used here is from Microsoft, Microsoft malware classification that it has detection or not. It has 82 attributes and 1 target attribute with more than 10,50,000 records.

Missing Ratio Value:

Here, we set the threshold percentage as 80%, that is if the missing percentage of a particular attribute is greater than 80% the particular attribute is eliminated.

```
[9] a=x.isnull().sum()/len(x)*100
    print(a.sort_values(ascending=False))
```

```
PuaMode                                          99.975600
Census_ProcessorClass                            99.579999
DefaultBrowsersIdentifier                        95.139590
Census_IsFlightingInternal                       83.030966
Census_InternalBatteryType                       71.028342
Census_ThresholdOptIn                            63.502727
Census_IsWIMBootEnabled                          63.414727
SmartScreen                                      35.659071
OrganizationIdentifier                           30.871662
SMode                                             6.011612
CityIdentifier                                    3.641807
Wdft_IsGamer                                      3.418207
Census_InternalBatteryNumberOfCharges             3.025006
Census_FirmwareManufacturerIdentifier             2.052004
Census_FirmwareVersionIdentifier                  1.791204
Census_IsFlightsDisabled                          1.779404
Census_OEMModelIdentifier                         1.131602
Census_OEMNameIdentifier                          1.054002
Firewall                                          1.035802
Census_TotalPhysicalRAM                           0.905802
Census_IsAlwaysOnAlwaysConnectedCapable           0.794802
IeVerIdentifier                                   0.675001
Census_OSInstallLanguageIdentifier                0.663601
Census_PrimaryDiskTotalCapacity                   0.593201
Census_SystemVolumeTotalCapacity                  0.593201
Census_InternalPrimaryDiagonalDisplaySizeInInches 0.548001
Census_InternalPrimaryDisplayResolutionHorizontal 0.546801
Census_InternalPrimaryDisplayResolutionVertical   0.546801
Census_ProcessorModelIdentifier                   0.470001
Census_ProcessorManufacturerIdentifier            0.469401
Census_ProcessorCoreCount                         0.469401
```

As we can see in the above image the four attributes have missing percentage value more than the threshold percentage value, we will eliminate them.

Data Cleaning:

The dataset has some numerical attributes as well as some categorical attributes. So, to convert those categorical attributes to numerical we use label encoder model from scikit-learn pre-processing class which converts the categorical values to numerical values.

As in above we removed the attributes which is having missing percentage value greater than 80%, now what about the remaining attributes which has missing percentage value less than 80%. We are going to fill those missing values by calculating the mean of the particular column.

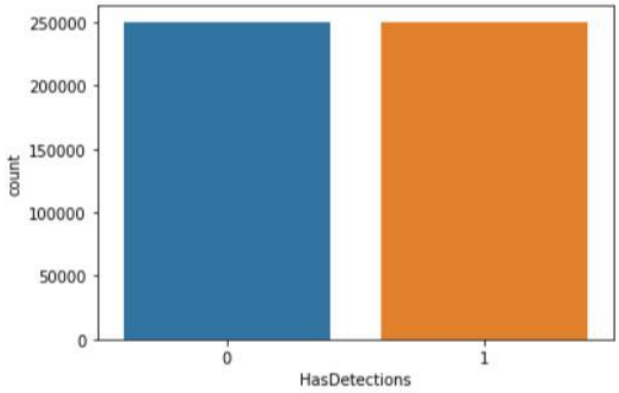
```
[17] print(x1.isnull().values.any())
```

```
False
```

Data Representation:

```
[21] sns.countplot(data['HasDetections'],label='count')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ab9932400>
```



As we can see from the above image that both classes of our dataset are equal.

Low Variance:

Eliminating the lowest variance attributes.

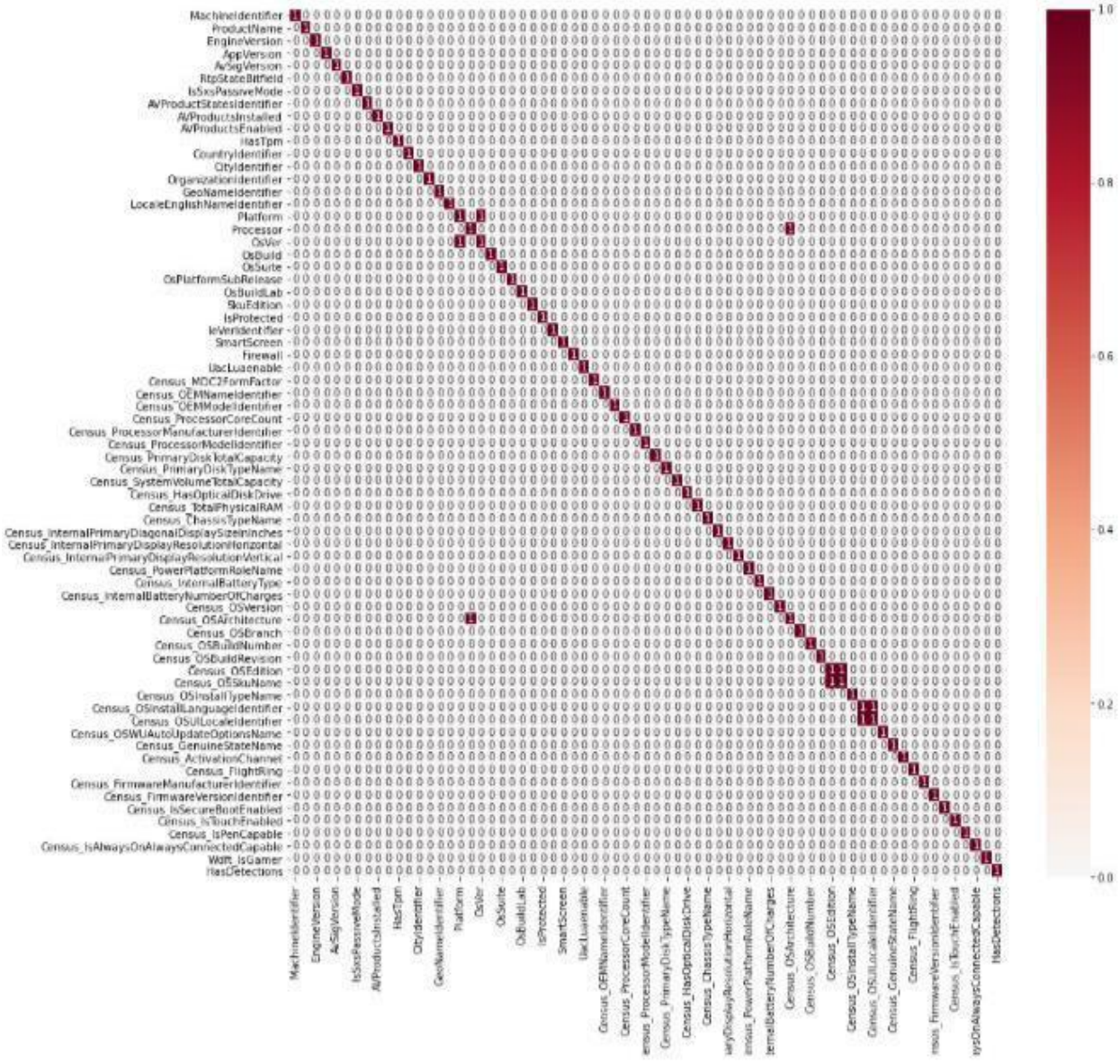
```
[32] x1.var().sort_values()
```

```
Census_IsWIMBootEnabled                          0.000000e+00
Census_IsFlightsDisabled                         7.999967e-06
IsBeta                                           9.999940e-06
AutoSampleOptIn                                  2.799933e-05
Census_ThresholdOptIn                            1.059696e-04
SMode                                            4.298050e-04
Census_IsPortableOperatingSystem                 6.355980e-04
Census_DeviceFamily                              1.613395e-03
Census_IsVirtualDevice                           6.901637e-03
HasTpm                                           1.202974e-02
IsSxsPassiveMode                                 1.698726e-02
Firewall                                         2.083016e-02
AVProductsEnabled                                2.768017e-02
Census_IsPenCapable                              3.653397e-02
ProductName                                      4.243250e-02
IsProtected                                      5.148762e-02
Census_IsAlwaysOnAlwaysConnectedCapable          5.331605e-02
UacLuaenable                                     5.450635e-02
Census_HasOpticalDiskDrive                       7.088172e-02
Processor                                        8.271878e-02
Census_IsTouchEnabled                            1.099568e-01
Census_GenuineStateName                          1.178957e-01
Census_FlightRing                                1.743435e-01
Wdft_IsGamer                                     1.958174e-01
Platform                                         2.331781e-01
Census_IsSecureBootEnabled                       2.498254e-01
AVProductsInstalled                              2.716462e-01
Census_OSArchitecture                            3.306329e-01
Census_PrimaryDiskTypeName                       5.322300e-01
SkuEdition                                       1.025310e+00
RtpStateBitfield                                 1.037777e+00
Census_ActivationChannel                         1.292530e+00
Census_ProcessorManufacturerIdentifier           1.656279e+00
Census_PowerPlatformRoleName                     1.736633e+00
```

In the above we can see that the first nine attributes have very low variance. So, we will eliminate them.

High Correlation Filter:

Highest correlated attributes are eliminated.



From the above we can visualise the highest correlated attribute and we can eliminate the one which has less correlation with the target attribute. We removed four attributes after this step.

(i) PCA:

After applying the above techniques, from 82 features we have only 64 features.

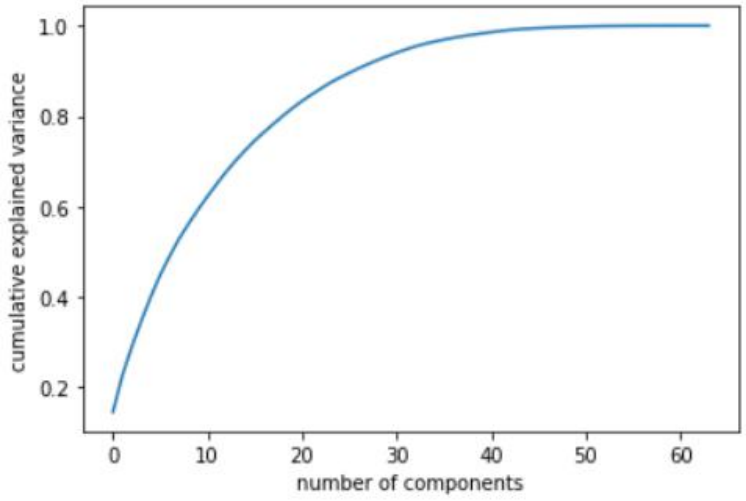
Normalization: The data needs to be normalized before applying PCA. To normalize the data we used min_max normalization that is scaling the records between the range 0 and 1.

$$v' = \frac{v - min_i}{max_i - min_i}(new\_max_i - new\_min_i) + new\_min_i$$

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x1)
x1_trans = scaler.transform (x1)
print(x1_trans)
```

```
[[0.00000000e+00 7.50000000e-01 9.44444444e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [2.00000800e-06 7.50000000e-01 7.59259259e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [4.00001600e-06 7.50000000e-01 9.44444444e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 ...
 [9.99994000e-01 7.50000000e-01 9.44444444e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [9.99996000e-01 7.50000000e-01 9.62962963e-01 ... 0.00000000e+00
  0.00000000e+00 0.00000000e+00]
 [9.99998000e-01 7.50000000e-01 9.44444444e-01 ... 0.00000000e+00
  0.00000000e+00 1.00000000e+00]]
```

PCA is applied now to reduce the dimension of our data. Cumulative explained variance graph is plotted to select the number of principle components to select is decided after visualizing the graph with the principle components.

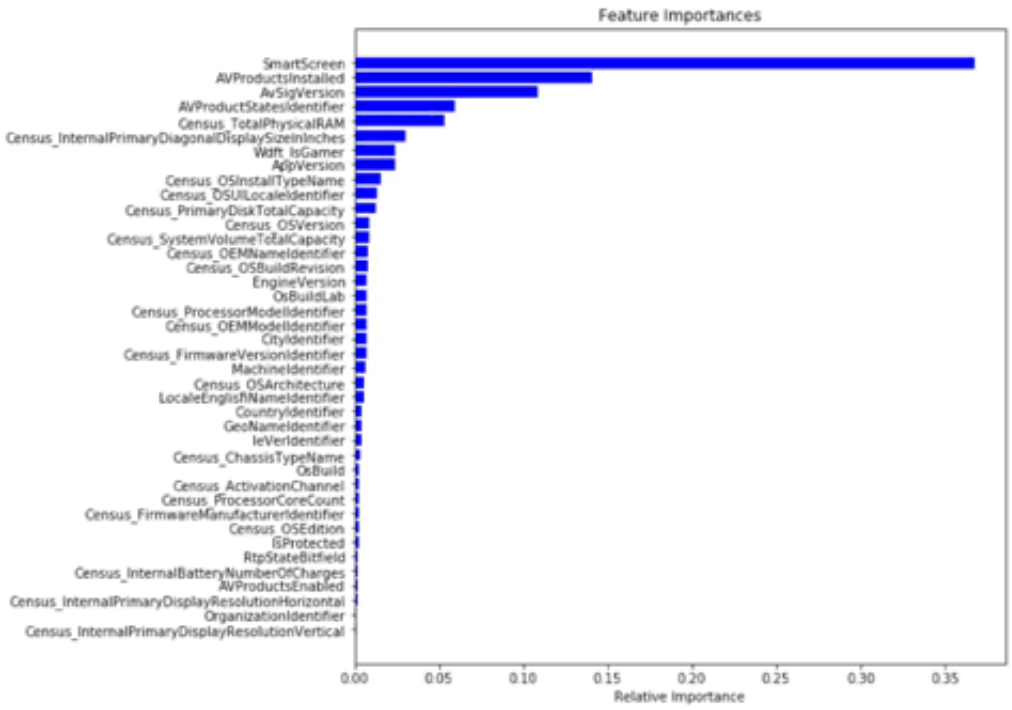


From the above image we can see that the variance after 50^{th} component is in a straight forward direction that means there is no rise in variance after the 50^{th} principal component. So, the principle components selected for training and testing are the first 50 components.

Now, the selected components and the target variable is passed to classification algorithms to calculate accuracy.

(ii) Random Forest Feature Selection:

The Random Forest Regressor class has a feature selection model which selects features with according to there weights.



From the above graph we can pick important features. Here we picked first 30 features for training and testing.

Now, the selected features and the target variable is passed to classification algorithms to calculate accuracy.

Comparison of accuracy between dimension reduction and feature selection for different algorithms:

Algorithm	**Without dimension reduction**	**Using PCA (40% dimension reduced i.e. 82 to 50)**	**Using feature selection random forest (first 30 important features selected out of 82)**	**Using feature selection random forest (first 50 important features selected out of 82)**
Logistic Regression	52.52%	61.09%	55.1%	50.59%
Decision Tree	55.66%	58.72%	61.18%	61.16%
Random Forest	59.9%	61.44%	62.73%	62.77%
SVM	49.87%	61.09%	50.57%	50.57%
Ensemble Learning	55.28%	61.92%	58.03%	50.59%

CONCLUSION:

The objective of this paper was to reduce the features as much as possible by using various techniques and you can see the results in Table that accuracy increased by using PCA for dimension reduction and also the computation was also fast. Using feature selection of random forest, we realised that computation was faster compared to PCA but accuracy was not that good but was up to the mark. It depends on the dataset you're using which technique is used. We used the Microsoft's malware detection dataset, it was a kind of a challenging task because of huge records and pre-processing was hard. For using PCA you need to normalize the data first so it takes a little more time than feature selection of random forest class.

IMPLEMENTATION:

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from sklearn.metrics import accuracy_score
import seaborn as sns
import math
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv("drive/My Drive/ms_malware.csv")
data.head()
x=data.iloc[:,:81]
y=data.iloc[:,[82]]
#print(y)
a=x.isnull().sum()/len(x)*100
print(a.sort_values(ascending=False))
```

```
PuaMode                                          99.975600
Census_ProcessorClass                            99.579999
DefaultBrowsersIdentifier                        95.139590
Census_IsFlightingInternal                       83.030966
Census_InternalBatteryType                       71.028342
Census_ThresholdOptIn                            63.502727
Census_IsWIMBootEnabled                          63.414727
SmartScreen                                      35.659071
OrganizationIdentifier                           30.871662
SMode                                             6.011612
CityIdentifier                                    3.641807
Wdft_IsGamer                                      3.418207
Census_InternalBatteryNumberOfCharges             3.025006
Census_FirmwareManufacturerIdentifier             2.052004
Census_FirmwareVersionIdentifier                  1.791204
Census_IsFlightsDisabled                          1.779404
Census_OEMModelIdentifier                         1.131602
Census_OEMNameIdentifier                          1.054002
Firewall                                          1.035802
Census_TotalPhysicalRAM                           0.905802
Census_IsAlwaysOnAlwaysConnectedCapable           0.794802
IeVerIdentifier                                   0.675001
Census_OSInstallLanguageIdentifier                0.663601
Census_PrimaryDiskTotalCapacity                   0.593201
Census_SystemVolumeTotalCapacity                  0.593201
Census_InternalPrimaryDiagonalDisplaySizeInInches 0.548001
Census_InternalPrimaryDisplayResolutionHorizontal 0.546801
Census_InternalPrimaryDisplayResolutionVertical   0.546801
Census_ProcessorModelIdentifier                   0.470001
Census_ProcessorManufacturerIdentifier            0.469401
Census_ProcessorCoreCount                         0.469401
AVProductStatesIdentifier                         0.406201
AVProductsEnabled                                 0.406201
AVProductsInstalled                               0.406201
IsProtected                                       0.404001
RtpStateBitfield                                  0.374801
Census_IsVirtualDevice                            0.182000
Census_PrimaryDiskTypeName                        0.149000
UacLuaenable                                      0.122600
Census_ChassisTypeName                            0.005400
GeoNameIdentifier                                 0.000600
Census_PowerPlatformRoleName                      0.000400
OsBuildLab                                        0.000200
Census_IsPenCapable                               0.000000
OsBuild                                           0.000000
IsSxsPassiveMode                                  0.000000
IsBeta                                            0.000000
HasTpm                                            0.000000
AvSigVersion                                      0.000000
```

```
variables = x.columns
```

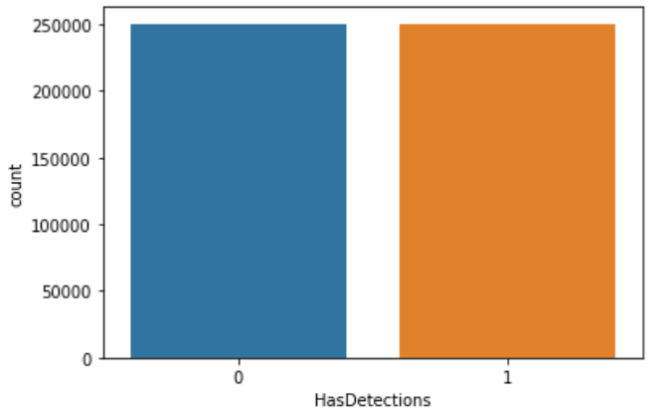
```
variable = [ ]
for i in range(0,len(a)):
    if a[i]<=80:#setting the threshold as 80%
        variable.append(variables[i])
print(len(variable))
```

Output:77

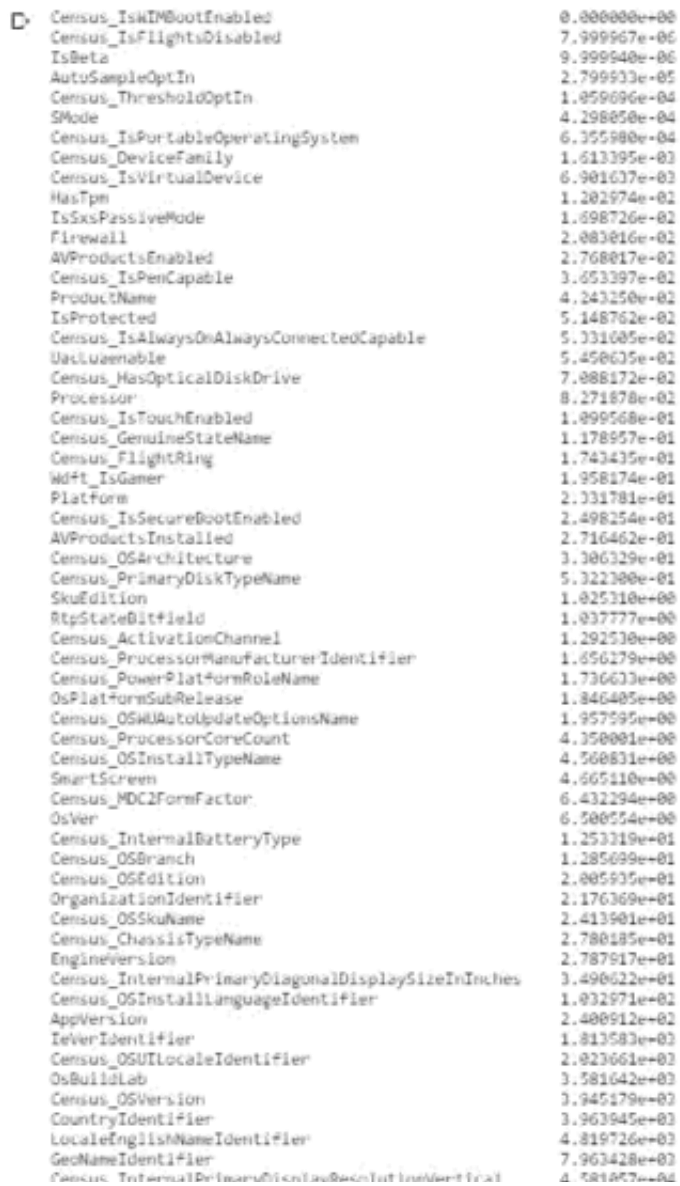
```
x1=x[variable]

col=x1.columns
num=LabelEncoder()
for i in col:
    if x1[i].dtype.name=='object':
        x1[i]=num.fit_transform(x1[i].astype('str'))

 for i in variable:
    if a[i]>0:
        x1[i].fillna(x1[i].mean(), inplace=True)

print(x1.isnull().values.any())
```

Output:False

```
sns.countplot(data['HasDetections'],label='count')
```

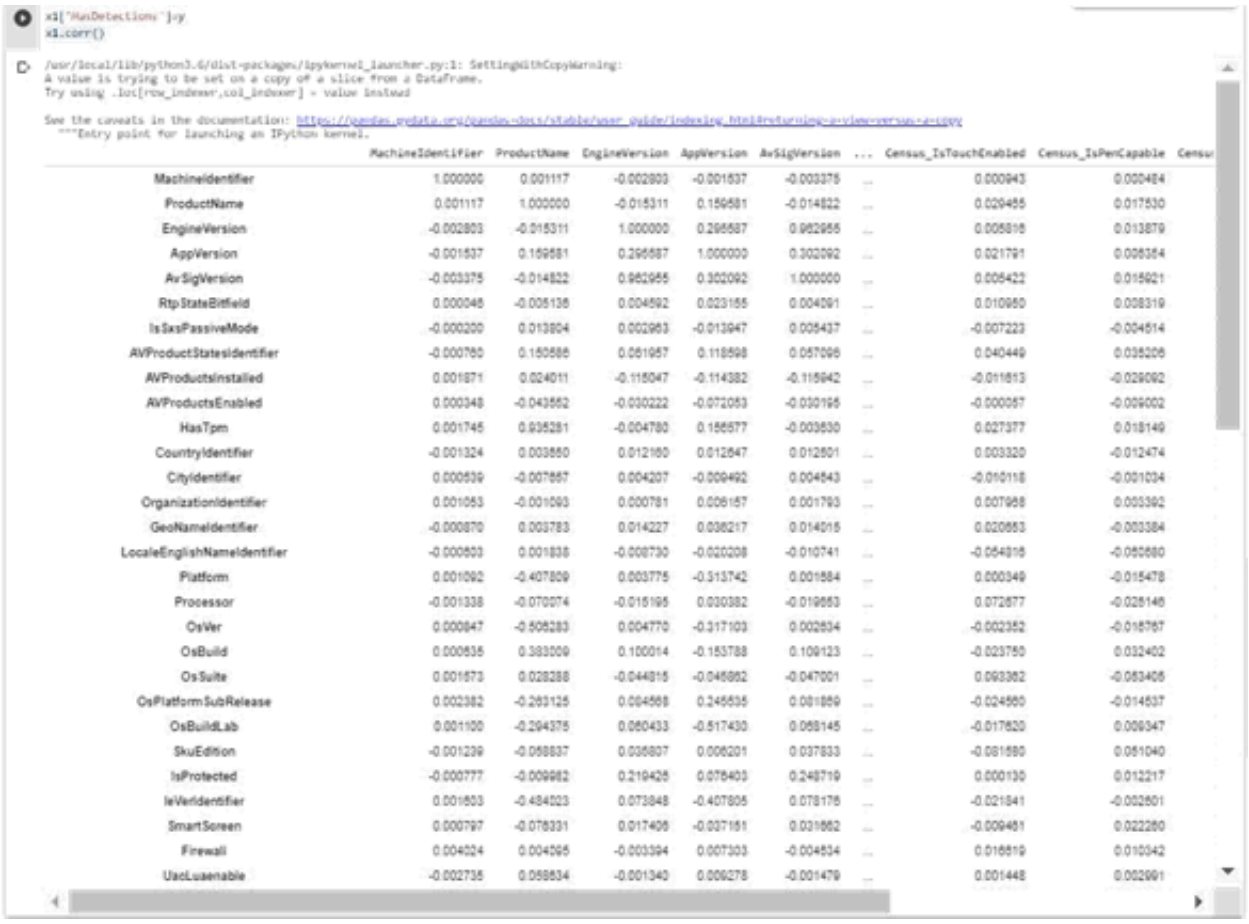



```
x1.var().sort_values()
```



```
Census_IsWIMBootEnabled                          0.000000e+00
Census_IsFlightsDisabled                         7.999967e-06
IsBeta                                           9.999940e-06
AutoSampleOptIn                                  2.799933e-05
Census_ThresholdOptIn                            1.059696e-04
SMode                                            4.298850e-04
Census_IsPortableOperatingSystem                 6.355980e-04
Census_DeviceFamily                              1.613395e-03
Census_IsVirtualDevice                           6.901637e-03
HasTpm                                           1.202974e-02
IsSxsPassiveMode                                 1.698726e-02
Firewall                                         2.083016e-02
AVProductsEnabled                                2.768017e-02
Census_IsPenCapable                              3.653397e-02
ProductName                                      4.243250e-02
IsProtected                                      5.148762e-02
Census_IsAlwaysOnAlwaysConnectedCapable          5.331605e-02
Wdft_IsGamer                                     5.450635e-02
Census_HasOpticalDiskDrive                       7.088172e-02
Processor                                        8.271878e-02
Census_IsTouchEnabled                            1.099568e-01
Census_GenuineStateName                          1.178957e-01
Census_FlightRing                                1.743435e-01
Wdft_IsGamer                                     1.958174e-01
Platform                                         2.331781e-01
Census_IsSecureBootEnabled                       2.498254e-01
AVProductsInstalled                              2.716462e-01
Census_OSArchitecture                            3.306329e-01
Census_PrimaryDiskTypeName                       5.322300e-01
SkuEdition                                       1.025310e+00
RtpStateBitfield                                 1.037777e+00
Census_ActivationChannel                         1.292530e+00
Census_ProcessorManufacturerIdentifier           1.656279e+00
Census_PowerPlatformRoleName                     1.736633e+00
OsPlatformSubRelease                             1.846405e+00
Census_OSWUAutoUpdateOptionsName                 1.957595e+00
Census_ProcessorCoreCount                        4.350001e+00
Census_OSInstallTypeName                         4.560831e+00
SmartScreen                                      4.665110e+00
Census_MDC2FormFactor                            6.432294e+00
OsVer                                            6.500554e+00
Census_InternalBatteryType                       1.253319e+01
Census_OSBranch                                  1.285099e+01
Census_OSEdition                                 2.005935e+01
OrganizationIdentifier                           2.176369e+01
Census_OSSkuName                                 2.413901e+01
Census_ChassisTypeName                           2.780185e+01
EngineVersion                                    2.787917e+01
Census_InternalPrimaryDiagonalDisplaySizeInInches 3.490622e+01
Census_OSInstallLanguageIdentifier               1.032971e+02
AppVersion                                       2.400912e+02
IeVerIdentifier                                  1.813583e+03
Census_OSUILocaleIdentifier                      2.023661e+03
OsBuildLab                                       3.581042e+03
Census_OSVersion                                 3.945179e+03
CountryIdentifier                                3.963945e+03
LocaleEnglishNameIdentifier                      4.819726e+03
GeoNameIdentifier                                7.963428e+03
Census_InternalPrimaryDisplayResolutionVertical  4.581857e+04
```

```
x1.drop('Census_IsFlightsDisabled',axis=1,inplace=True)
x1.drop('IsBeta',axis=1,inplace=True)
x1.drop('AutoSampleOptIn',axis=1,inplace=True)
x1.drop('SMode',axis=1,inplace=True)
x1.drop('Census_IsPortableOperatingSystem',axis=1,inplace=True)
x1.drop('Census_IsWIMBootEnabled',axis=1,inplace=True)
x1.drop('Census_ThresholdOptIn',axis=1,inplace=True)
x1.drop('Census_DeviceFamily',axis=1,inplace=True)
x1.drop('Census_IsVirtualDevice',axis=1,inplace=True)


x1.corr()
```

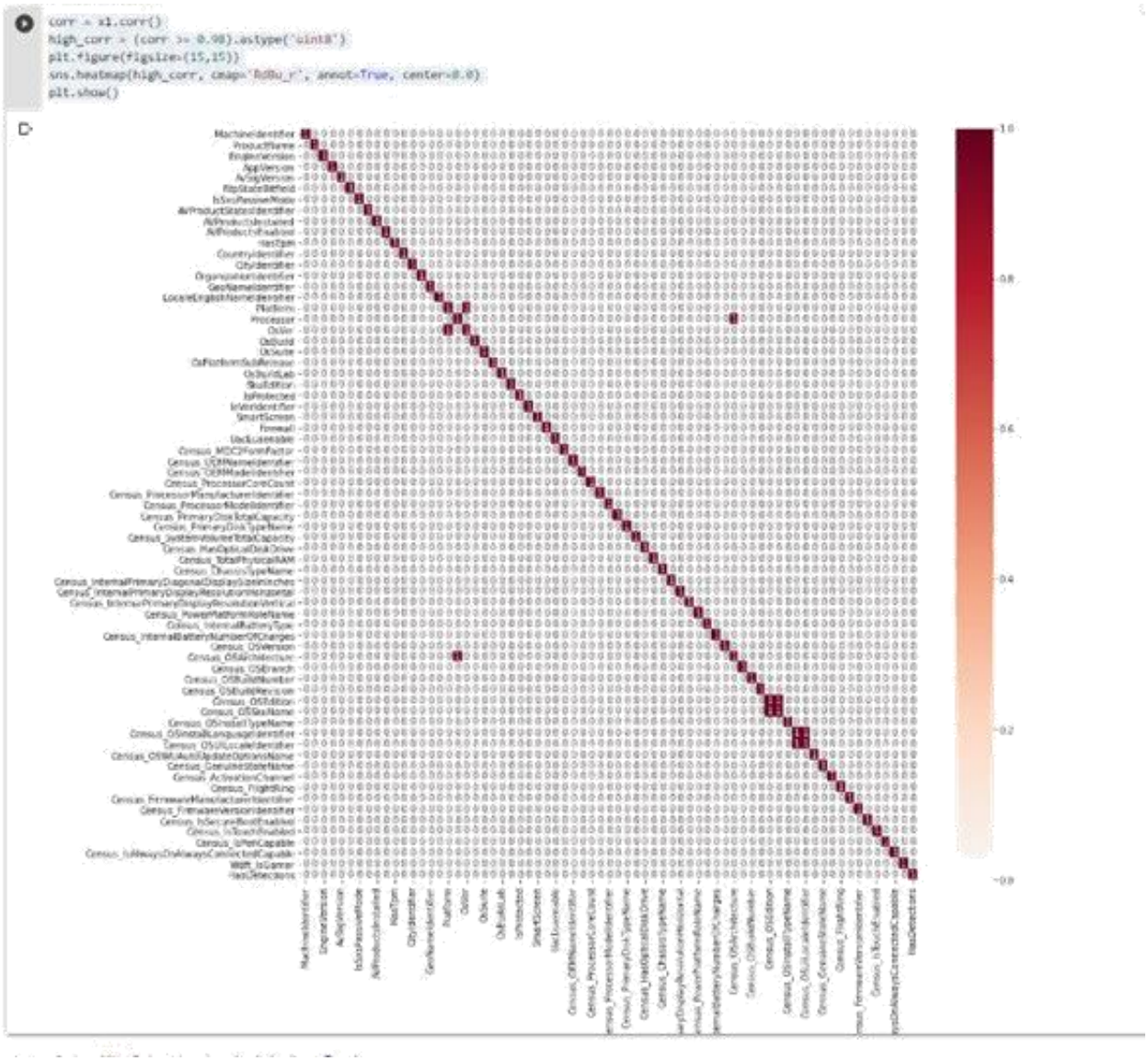


```
x1['HasDetections'].sy
x1.corr()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion	...	Census_IsTouchEnabled	Census_IsPenCapable	Censu
MachineIdentifier	1.000000	0.001117	-0.002803	-0.001537	-0.003375	...	0.000943	0.000484	
ProductName	0.001117	1.000000	-0.015311	0.159681	-0.014822	...	0.029455	0.017530	
EngineVersion	-0.002803	-0.015311	1.000000	0.296587	0.962955	...	0.005816	0.013879	
AppVersion	-0.001537	0.159681	0.296587	1.000000	0.302092	...	0.021791	0.006364	
AvSigVersion	-0.003375	-0.014822	0.962955	0.302092	1.000000	...	0.005422	0.015921	
RtpStateBitfield	0.000046	-0.005136	0.004592	0.023155	0.004091	...	0.010950	0.008319	
IsSxsPassiveMode	-0.000200	0.013804	0.002963	-0.013947	0.005437	...	-0.007223	-0.004514	
AVProductStatesIdentifier	-0.000760	0.150586	0.061957	0.118598	0.057066	...	0.040449	0.035206	
AVProductsInstalled	0.001871	0.024011	-0.115047	-0.114382	-0.115942	...	-0.011613	-0.029092	
AVProductsEnabled	0.000348	-0.043552	-0.030222	-0.072053	-0.030195	...	-0.000057	-0.009002	
HasTpm	0.001745	0.935281	-0.004780	0.156577	-0.003830	...	0.027377	0.018149	
CountryIdentifier	-0.001324	0.003550	0.012160	0.012647	0.012501	...	0.003320	-0.012474	
CityIdentifier	0.000539	-0.007557	0.004207	-0.009492	0.004543	...	-0.010118	-0.001034	
OrganizationIdentifier	0.001053	-0.001093	0.000781	0.006157	0.001793	...	0.007958	0.003392	
GeoNameIdentifier	-0.000870	0.003783	0.014227	0.036217	0.014015	...	0.020653	-0.003384	
LocaleEnglishNameIdentifier	-0.000503	0.001838	-0.000730	-0.020208	-0.010741	...	-0.054016	-0.050680	
Platform	0.001092	-0.407809	0.003775	-0.313742	0.001884	...	0.000349	-0.015478	
Processor	-0.001338	-0.070074	-0.015195	0.030382	-0.019653	...	0.072677	-0.028146	
OsVer	0.000847	-0.506283	0.004770	-0.317103	0.002834	...	-0.002352	-0.016767	
OsBuild	0.000635	0.383009	0.100014	-0.153788	0.109123	...	-0.023750	0.032402	
OsSuite	0.001573	0.028288	-0.044015	-0.045862	-0.047001	...	0.093362	-0.063406	
OsPlatformSubRelease	0.002382	-0.263125	0.004568	0.245535	0.001859	...	-0.024550	-0.014537	
OsBuildLab	0.001100	-0.294375	0.060433	-0.517430	0.059145	...	-0.017620	0.009347	
SkuEdition	-0.001239	-0.058837	0.036807	0.006201	0.037833	...	-0.081580	0.051040	
IsProtected	-0.000777	-0.009982	0.219425	0.078403	0.248719	...	0.000130	0.012217	
IeVerIdentifier	0.001603	-0.484023	0.073848	-0.407805	0.078176	...	-0.021841	-0.002601	
SmartScreen	0.000797	-0.078331	0.017406	-0.037151	0.031662	...	-0.009451	0.022260	
Firewall	0.004024	0.004095	-0.003394	0.007303	-0.004534	...	0.016519	0.010342	
UacLuaenable	-0.002735	0.059834	-0.001340	0.009278	-0.001479	...	0.001448	0.002991	

```
corr = x1.corr()
high_corr = (corr >= 0.98).astype('uint8')
plt.figure(figsize=(15,15))
sns.heatmap(high_corr, cmap='RdBu_r', annot=True, center=0.0)
plt.show()
```

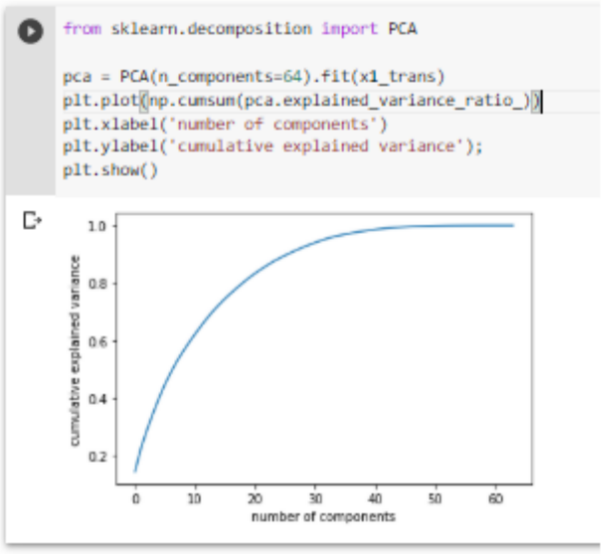



```
x1.drop('HasDetections',axis=1,inplace=True)


x1.drop('Platform',axis=1,inplace=True)
x1.drop('Census_OSSkuName',axis=1,inplace=True)
x1.drop('Census_OSInstallLanguageIdentifier',axis=1,inplace=True)
x1.drop('Processor',axis=1,inplace=True)


c=x1.columns
print(len(c))
```

Output: 64

PCA:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x1)
x1_trans = scaler.transform (x1)
print(x1_trans)
from sklearn.decomposition import PCA

pca = PCA(n_components=64).fit(x1_trans)
plt.plot(np.cumsum(pca.explained_variance_ratio_)
) plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
plt.show()
```




```
pca = PCA (n_components = 50)
principle_components = pca.fit_transform (x1_trans)
principleDf = pd.DataFrame(data = principle_components)


y=np.array(y)

X_train, X_test, y_train, y_test = train_test_split (principleDf,
y,train_ size=0.7,test_size=0.3,random_state=50)


from sklearn.linear_model import LogisticRegression
LRModel = LogisticRegression().fit(X_train, y_train)
LRPredict = LRModel.predict(X_test)
accuracy1 = accuracy_score (LRPredict, y_test)
print(accuracy1*100)
```

OUTPUT: 61.09

```
from sklearn import tree
treeModel = tree.DecisionTreeClassifier(max_depth=5).fit (X_train,
y_train )
treePredict = treeModel.predict (X_test)
accuracy2 = accuracy_score (treePredict, y_test)
print (accuracy2*100)
```
OUTPUT: 58.728

```
from sklearn.ensemble import RandomForestClassifier
RFModel = RandomForestClassifier().fit (X_train,
y_train) RFPredict = RFModel.predict(X_test)
accuracy3 = accuracy_score (RFPredict, y_test)
print(accuracy3*100)
```
OUTPUT:61.08

```
from sklearn.svm import SVC
model=SVC(kernel='rbf',C=10000,gamma=0.001)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
accuracy = accuracy_score (y_pred, y_test)
print(accuracy1*100)
```
OUTPUT: 61.09

ENSEMBLE LEARNING:

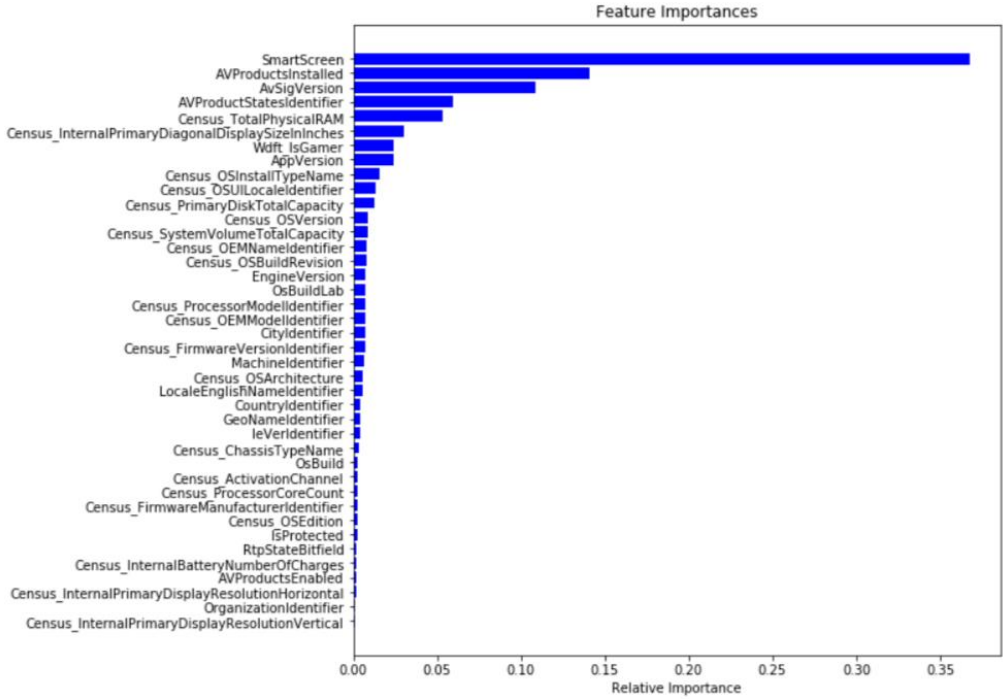
```
l=[]

for i in range(len(LRPredict)):
    k=[LRPredict[i],treePredict[i],RFPredict[i],y_pred[i]]
    l.append(k)

#print(l)

li=[]

for i in range(len(l)):
    c=0
    c1=0
    for j in range(len(l[i])):
        if l[i][j]==0:
            c=c+1
        else:
            c1=c1+1
    if c>c1:
        li.append(0)
    else:
        li.append(1)
```

```
lis=np.array(li)
accuracy4=accuracy_score (lis, y_test)
print(accuracy4*100)
```

OUTPUT: 61.90

Random Forest Feature Seclection:

```
from sklearn.ensemble import RandomForestRegressor
rfsmodel = RandomForestRegressor(random_state=1, max_depth=10)
rfsmodel.fit(x1,y)
features = x1.columns
importances = rfsmodel.feature_importances_
indices = np.argsort(importances)[-50:] # top 50 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b',
align='cent er')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.rcParams['figure.figsize']=(9,9)
plt.show()
```




```
x1_sel=pd.DataFrame()
for i in indices:
    s=features[i]
    x1_sel[s]=x1[s]
X_train1, X_test1, y_train1, y_test1 = train_test_split (x1_sel,
y,train_s ize=0.7,test_size=0.3,random_state=21)
```

```
LRModel1 = LogisticRegression().fit(X_train1, y_train1)
LRPredict1 = LRModel1.predict(X_test1)
accuracy11 = accuracy_score(LRPredict1, y_test1)
print(accuracy11*100)
```
OUTPUT: 50.59

```
treeModel1 = tree.DecisionTreeClassifier(max_depth=5).fit (X_train1,
y_tra in1)
treePredict1 = treeModel1.predict (X_test1)
accuracy21 = accuracy_score(treePredict1, y_test1)
print (accuracy21*100)
```
OUTPUT: 61.16

```
RFModel1 = RandomForestClassifier().fit (X_train1, y_train1)
RFPredict1 = RFModel1.predict(X_test1)
accuracy31 = accuracy_score (RFPredict1, y_test1)
print(accuracy31*100)
```
OUTPUT: 62.89

```
from sklearn.svm import SVC
model1=SVC(kernel='rbf',C=10000,gamma=0.001)
model1.fit(X_train1,y_train1)
y_pred1=model1.predict(X_test1)
accuracyS1 = accuracy_score (y_pred1, y_test1)
print(accuracyS1*100)
```
OUTPUT: 50.57

```
l1=[]

for i in range(len(LRPredict1)):
    k=[LRPredict1[i],treePredict1[i],RFPredict1[i],y_pred1[i]]
    l1.append(k)
li1=[]

for i in range(len(l1)):
    c=0
    c1=0
    for j in range(len(l1[i])):
        if l1[i][j]==0:
            c=c+1
        else:
            c1=c1+1
    if c>c1:
        li1.append(0)
    else:
        li1.append(1)
lis1=np.array(li1)
```

```
accuracy41=accuracy_score (lis1, y_test1)
print(accuracy41*100)
```

OUTPUT: 50.59

REFERENCE:

[1] Anshuman Sahu , Daniel W. Apley , George C. Runger "Feature selection for noisy variation patterns using kernel principal component analysis"."2014 Elsevier B.V."

[2] Ross Goroshin, Jonathan Tompson, Debidatta Dwibedi " AN ANALYSIS OF OBJECT REPRESENTATIONS IN DEEP VISUAL TRACKERS "."Cornell University, Jan 2020"J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] Lindsay I Smith "A tutorial on Principle Components Analysis", Department of Computer Science, University of Otago, Feb 2012.

[4] Hamid Hassanpour, Amin Zehtabian, Avishan Nazari, Hossein Dehgha"Gender classification based on fuzzy clustering and principal component analysis"."IET Computer Visi.

[5] Fengxi Song, Zhongwei Guo, Dayong Mei "Feature selection using principal component analysis"."2010 International Conference on System Science, Engineering Design and Manufacturing Informatization".

[6] Annie George "Anomaly detection based on machine learning Dimensionality reduction using Principal Component Analysis,Classification using Support Vector Machine"."International Journal of Computer Applications (0975– 8887) Volume47– No.21, June 2012 "

[7] Shweta Saini ,Sugandha Sharma,Rupinder Singh "Better utilization of correlation between metrics using Principal Component Analysis (PCA) "."IEEE INDICON 2015 1570173355 ".

[8] Principal component analysis (PCA) and multiple linear regression (MLR) statistical tools to evaluate the effect of E-beam irradiation on ready-to-eat food Author links open overlay panelVanesa Guillén-CaslaNoeliaRosales-ConradoMaría EugeniaLeón-GonzálezLuis VicentePérez-ArribasLuis MaríaPolo-Díez.

[9] Young-Jun Yoo "Fault Detection of Induction Motor Using Fast Fourier Transform with Feature Selection via Principal Component Analysis", "International Journal of Precision Engineering and Manufacturing' June 2019.

[10] Liton Chandra Paul, Abdulla Al Sumam "Face Recognition Using Principal Component Analysis Method"."International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 9, November 2012 "

[11] Ronny Luss · Alexandre d' Aspremont " Clustering and feature selection using sparse principal component analysis"."Received: 30 March 2007 / Accepted: 13 October 2008 / Published online: 6 November 2008 in Springer Science+Business Media, LLC 2008"

[12] Manisha Satone, Gajanan Kharate "Feature Selection Using Genetic Algorithm for Face Recognition Based on PCA, Wavelet and SVM "."International Journal on Electrical Engineering and Informatics - Volume 6, Number 1, March 2014".