Network Intrusion Detection System (IDS)

***Abstract:***

**Since last decade, many researches have been done on network intrusion detection (IDS) to make to it efficient and more accurate and KDDCUP’99 is the dataset used by most of the researchers. It is important a very important research area because of the internet users are increasing rapidly.** **Many studies concluded that by decreasing redundancy selecting relevant features from a dataset set can make a model more accurate and efficient. So used Recursive Feature Elimination Unsupervised Algorithm to select most relevant 17 features. Then we used K-Nearest Neighbor Classifier to classify the attacks. And then evaluated models by K-Fold Cross Validation.**

***Keywords: Data Preprocessing, KDDCup99 Dataset, Network Intrusion Detection, Pandas, Scikit-Learn, Matplotlib, Feature Selection, Correlation Matrix, Recursive Feature Elimination, K-Nearest Neighbour.***

***Introduction:***

Day by day networks are becoming more complex. The growing need of internet made us to depend on computer network, so the network security plays a major role. That’s why, we need to find a best possible way to protect our systems. The security of the system is compromised when the intrusion takes place. [1][2][3]Intrusion is a kind of malicious assault that may enter to the gadget or network silently. Intrusion Detection System (IDS) is a software that keeps an eye on network or system for spiteful activities warns to take appropriate action in contradiction of intrusion.[10][11] An IDS must be highly accurate to detect the nasty activities on the network. The intrusion detection systems (IDS) are primarily focused on identifying possible incidents, logging information about them, and reporting attempts. We can use data mining and statistical techniques to make a better intrusion detection system. IDS check the various parameters and classifies that is it an attack or a normal situation.

***Literature Review:***

Many studies[5-9] have suggested that by removing redundancy or by selecting relevant features from a dataset can improve the accuracy and performance of the model as in a research done by [7]Vinutha H.P. and Dr. Poornima B , they selected features using different feature selection algorithms and then applied Naïve Bayes. And [5]Herve Nkiama used RFE for feature selection and then used decision tree classifiers. Some researchers [8]used hybrid feature selection algorithms for selecting features precisely. Many research papers on Network Intrusion Detection were reviewed since last decade and several methods had used to make the IDS more precise to detect intrusion as [9]Sevcan YILMAZ GÜNDÜZ used fuzzy unordered rule induction algorithm (FURIA) and support vector machine,

***Dataset-***

The right selection of data set plays a major role in performance of intrusion detection system. [4][5] KDD-Cup is the widely used dataset for training and testing of IDS. There are 41 features which are classified into Basic, Content and Traffic features. KDD-Cup is developed on the basis of DARPA’98 IDS evaluation program.

All the records in KDD dataset are categorized into :

1. As Normal intrusion

2. As a particular kind of intrusion

There are four categories of intrusion in the dataset[2][3][5][6] :

• **DOS:** Denial-of-service, DoS is one of the oldest forms of cyber extortion attack. In this attack the attacker makes some computing or memory resource too busy or too full to handle genuine requests or denies genuine users’ access to a machine.

• **R2L:** It’s a Remote to Local attack, An R2L attack which exploits a bug in a particular version of ncftp, the popular ftp client. The a user on the victim host ftp's to the attacker’s machine, and attempts to download, recursively, a directory. Contained in the directory is directory with a very long name, embedded in the name is one or more commands that are then executed (un-intentionally) by the ftp client with the permissions of that user. Example – Guessing passwords, etc.

• **U2R:** Unauthorized access to local superuser (root) privileges, It Basically get access to the root of the system and performs various attacks and unauthorized attempts. Example – various Buffer overflow attacks, Perl, rootkit, etc.

•**Probe:** This attack is an attempt to gather information about a network of computers for the apparent purpose of circumventing its security controls. For example, an empty message can be sent simply to see whether the destination exists. Ping is a common utility for sending such a probe. Example – Port scanning, SATAN, SAINT, portweep, etc.

These 4 main types are with 24 sub-types in training set and additional 14 sub-types in test set only.

There are 41 features for each record[4]:

|  |  |  |
| --- | --- | --- |
| **SO.NO.** | **Feature** | **Description** |
| 1 | duration | Duration of the connection |
| 2 | protocol type | Connection protocol (e.g. TCP, UDP, ICMP) |
| 3 | service | Destination service |
| 4 | flag | Status flag of the connection |
| 5 | source bytes | Bytes sent from source to destination |
| 6 | destination bytes | Bytes sent from destination to source |
| 7 | land | 1 if connection is from/to the same host/port; 0 otherwise |
| 8 | wrong  fragment | Number of wrong fragments |
| 9 | urgent | Number of urgent packets |
| 10 | hot | Number of “hot” indicators |
| 11 | failed logins | Number of failed logins |
| 12 | logged in | 1 if successfully logged in; 0 otherwise |
| 13 | #compromis  ed | Number of “compromised” conditions |
| 14 | root shell | 1 if root shell is obtained; 0 otherwise |
| 15 | su attempted | 1 if “su root” command attempted;  0 otherwise |
| 16 | #root | Number of “root” accesses |
| 17 | #file creations | Number of file creation operations |
| 18 | #shells | Number of shell prompts |
| 19 | #access files | Number of operations on access control files |
| 20 | #outbound cmds | Number of outbound commands in a ftp session |
| 21 | is hot login | 1 if login belongs to the “hot” list; 0 otherwise |
| 22 | is guest  login | 1 if the login is the “guest” login; 0 otherwise |
| 23 | count | Number of connections to the same host as the current connection in the  past 2 seconds |
| 24 | srv count | Number of connections to the same service as the current connection in  the past two seconds |
| 25 | serror rate | % of connections that have “SYN” errors |
| 26 | srv serror rate | % of connections that have “SYN” errors |
| 27 | rerror rate | % of connections that have REJ errors |
| 28 | srv rerror rate | % of connections that have REJ errors |
| 29 | same srv rate | % of connections to the same service |
| 30 | diff srv rate | % of connections to different services |
| 31 | srv diff host rate | % of connections to different hosts |
| 32 | dst host  count | Count of connections having the same destination host |
|  |  |  |
| 33 | dst host srv count | Count of connections having the same destination host and using the  same service |
| 34 | dst host  same srv rate | % of connections having the same destination host and using the same  service |
| 35 | dst host diff srv rate | % of different services on the current host |
| 36 | dst host  same src port rate | % of connections to the current host having the same src port |
| 37 | dst host srv  diff host rate | % of connections to the same service coming from different hosts |
| 38 | dst host  serror rate | % of connections to the current host that have an S0 error |
| 39 | dst host srv  serror rate | % of connections to the current host and specified service that have an  S0 error |
| 40 | dst host  rerror rate | % of connections to the current host that have an RST error |
| 41 | dst host srv  rerror rate | % of connections to the current host and specified service that have an RST error. |

***Libraries Used:***

***Numpy:*  Numpy**is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

***Pandas:*** Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

***Matplotlib:*** Matplotlib is an amazing visualization library in Python for making plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

***Scikit-Learn:*** Scikit-learn is a machine learning library written in python. Most of the learning algorithm implement in scikit-learn required data to be stored in a two-dimensional array or matrix.Sckit-learn provides a range of supervised and unsupervised learning algorithms via consistent interface in Python.

***Methodology And Individual Contributions :***

***Data Cleaning and Pre-processing(VAIBHAV SAWANDRE 11805312)*** Basically, in this step the dataset has to go through a cleaning process to remove duplicate records. In order to decrease the number of columns we have to find the correlated data and delete them.

Next a Pre-processing operation has to be taken in place because the dataset contains numerical and non-numerical instances. Label Encoder has to be used to encode the categorical data. The technique will transform each categorical feature with different numbers.

First open the csv file with pandas.read\_csv function and assign it to a variable named

***data***.

Now I need to make the correlation matrix and delete the correlated features, but for making a correlation matrix all the data should be in a numeric format. For this I created a duplicate variable of ***data*** named ***data1.*** In ***data1*** I dropped all the categorical features in the ***data.*** To find the correlation matrix there is a function given in pandas named *corr().*

**Pearson’s Correlation :**

The Pearson’s correlation coefficient is calculated as the covariance of the two variables divided by the product of the standard deviation of each data sample.

Pearson's correlation coefficient = covariance(*x, y*) / (stdv(*x*) \* stdv(*y*))

The result of the calculation, the correlation coefficient can be interpreted to

***About corr() function :***

Dataframe.corr() is used to find the pairwise correlation of all columns in the dataframe.

**Parameters:**

* **method:**

*pearson :* standard correlation coefficient

*kendall :* Kendall Tau correlation coefficient

*spearman:* Spearman rank correlation

* **min\_period :** Minimum number of observations required per pair of columns to have a valid result. Currently only available for pearson and spearman correlation.

Pearson’s Method is the default method of corr() function.

Let’s see how Pearson’s Correlation Method works.

It makes the correlation matrix of features. All the values in the matrix will be between -1 and 1. Values between -0.5 and 0.5 are less correlated and values greater than 0.5 or below -0.5 indicates a notable correlation. I have selected the columns to drop having correlation value greater than or equal to 0.7 and less than or equal to -0.7.

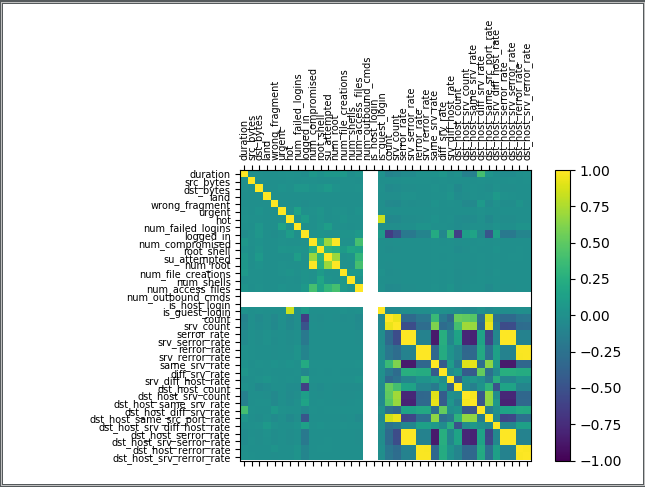
understand the relationship.

The coefficient returns a value between -1 and 1 that represents the limits of correlation from a full negative correlation to a full positive correlation. A value of zero means no correlation. The values below -0.5 or above 0.5 indicates a notable correlation, and values below those values suggests a less notable correlation.

This can be done by calculating a matrix of the relationships between each pair of variables in the dataset. The result is a symmetric matrix called a correlation matrix with a value of 1.0 along the diagonal as each column always perfectly correlates with itself.

I am giving the plot of correlation matrix below:

Fig1. Plot of Correlation Matrix



Below is the set of 14 column names selected to drop from the original dataset.

{'dst\_host\_srv\_rerror\_rate', 'num\_root', 'same\_srv\_rate', 'dst\_host\_srv\_serror\_rate', 'serror\_rate', 'srv\_rerror\_rate', 'dst\_host\_same\_srv\_rate', 'is\_guest\_login', 'dst\_host\_same\_src\_port\_rate', 'dst\_host\_srv\_count', 'dst\_host\_serror\_rate', 'num\_compromised', 'count', 'dst\_host\_rerror\_rate'}

Now we must drop these columns from original ***data*** to remove the collinearity between the features. Number of columns before dropping the correlated features was 42 and after deleting, we are left with 28 features.

For training the model, dataset must not have categorical variables. The columns {‘protocol\_type’, ‘service’, ‘flag’, ‘label’} in our dataset are having categorical variables so we

must convert these variables to numeric. A class named *LabelEncoder* is very useful for this purpose. It encodes the variables from categorical to numeric.

Now, convert the dataframe in the into a numpy matrix of rows and columns. Separate the input features and the label features from the matrix. I assigned ***dataX*** to the input features and ***dataY*** to the label feature.

In input features ***dataX*** column indexes 1, 2, 3 are categorical so I encoded them in numerical from using label encoder and the label ( ***dataY*** ) also encoded to numerical form.

***Feature Selection (UJJWAL DUBEY 11805234)***

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of your model. The data features that you use to train your machine learning models have a huge influence on the performance you can achieve. Irrelevant or partially relevant features can negatively impact model performance. Feature selection and Data cleaning should be the first and most important step of your model designing.

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

***What are Benefits of performing feature selection before modeling your data?***

**· Reduces Overfitting:** Less redundant data means less opportunity to make decisions based on noise.

**· Improves Accuracy:** Less misleading data means modeling accuracy improves.

**· Reduces Training Time:** fewer data points reduce algorithm complexity and algorithms train faster.

Correlations Matrix is also a feature selection technique, but I have done this previously.

I have used 4 feature selection techniques:

* Correlation Matrix
* Univariate Feature Selection:
* Recursive Feature Elimination (RFE)
* Principle Component Analysis (PCA)
* Linear Discriminant Analysis (LDA)

**Univariate Feature Selection:**

Statistical tests can be used to select those features that have the strongest relationship with the output variable.The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features. I have used chi-squared(chi2) statistical test for non-negative features to select the best features from the dataset.

**Recursive Feature Elimination (RFE):**

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model’s or feature\_importances\_ attributes, and by recursively eliminating a small number of features per loop, RFE attempts to eliminate dependencies and collinearity that may exist in the model. RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. RFE builds a model using the remaining attributes and calculates model accuracy. RFE is able to work out the combination of attributes that contribute to the prediction on the target variable (or class). Scikit Learn does most of the heavy lifting just import RFE from sklearn.feature\_selection and pass any classifier model to the RFE() method with the number of features to select. Using familiar Scikit Learn syntax, the .fit() method must then be called. I have used LogisticRegression classifier for RFE.

**Principle Component Analysis (PCA):**

Principal Component Analysis (PCA) is an unsupervised linear transformation technique that is widely used across different fields, most prominently for feature extraction and dimensionality reduction. Other popular applications of PCA include exploratory data analyses and de-noising of signals in stock market trading, and the analysis of genome data and gene expression levels in the field of bioinformatics.

PCA finds a new set of dimensions such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means more important principle axis occurs first. (more important = more variance/more spread out data).

**How does PCA work -**

1. Calculate the covariance matrix X of data points.
2. Calculate eigen vectors and corresponding eigen values.
3. Sort the eigen vectors according to their eigen values in decreasing order.
4. Choose first k eigen vectors and that will be the new k dimensions.
5. Transform the original n dimensional data points into k dimensions.

[k is the number of features that are given to select.]

**Linear Discriminant Analysis (LDA):**

Linear Discriminant Analysis is a dimensionality reduction technique used as a preprocessing step in Machine Learning and pattern classification applications. The main goal of dimensionality reduction techniques is to reduce the dimensions by removing the redundant and dependent features by transforming the features from higher dimensional space to a space with lower dimensions. Linear Discriminant Analysis is a supervised classification technique which takes labels into consideration. This category of dimensionality reduction is used in biometrics, bioinformatics and chemistry.

The goal of Linear Discriminant Analysis is to project the features in higher dimension space onto a lower dimensional space.

This can be achieved in three steps:

The first step is to calculate the separability between different classes (i.e. the distance between the mean of different classes) also called as between-class variance.

Second Step is to calculate the distance between the mean and sample of each class, which is called the within class variance.

The third step is to construct the lower dimensional space which maximizes the between class variance and minimizes the within class variance.

***Model (ASHISH 11805251)***

Here, a function *acuu()* was made to find the accuracies by implementing the different algorithms on the dataset. This function takes two arguments *x, y.* ***x*** is input data and ***y*** is the label. Inside the function data was divided into training and testing set using train\_test\_split. 33% of data was separated as testing data from the original data given as arguments to the function.

Next step inside the function was feature scaling. Features scaling is a common requirement of machine learning methods, to avoid that features with large values may weight too much on the final results. For feature scaling Normalization was used. Normalization rescales the value into a range of [0,1] but the pattern between the values doesn’t change.

***Normalization:***

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

Inside this function 5 different classifiers were used for experiment. Below is the list of all the algorithms that were used in the function:

1. Support Vector Machine
2. Perceptron
3. K-Nearest Neighbors
4. Stochastic Gradient Descent
5. XGBoost

Then, for model evaluation 2 methods were used-

1. Accuracy Score
2. K-fold Cross Validation With each classifier *accuracy\_score()* function was used for finding the accuracy of that classifier and then *cross\_val\_score()* was also used for model evaluation with 10 folds.

***K-Fold Cross Validation:***

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
4. Take the group as a hold out or test data set
5. Take the remaining groups as a training data set

1. Fit a model on the training set and evaluate it on the test set
2. Retain the evaluation score and discard the model.

4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

***Experiments and Results:***

In this project 5 feature selection algorithms were used and the function *acuu()* was called 11 times.

First time, the dataset after deleting the correlation was given (which was having 28 features) to the function *acuu().* Function *acuu()* performed all the algorithms on the dataset and gave the accuracies and the graphs of accuracies given by the functions accuracy\_score() and cross\_val\_score() .

Then, on a random basis 17 features were selected from the dataset (which was having 28 features) by each of the four feature scaling algorithms and passed into the function one by one.

After that, 11 features were selected from the dataset (which was having 28 features) by each of the four feature scaling algorithms and passed into the function one by one.

Then, all the four feature scaling algorithms were used as a stack i.e., first 23 features were selected from the dataset (which was having 28 features) by Univariate Feature Selection and then from those 23 features, 20 best features were selected by Recursive Feature Elimination and then from those 20 features, 16 features were selected by Principal Component Analysis and then from those 16 features, 11 features were selected by Linear Discriminant Analysis and then those 11 features were passed as argument to the function *acuu().*

At last, 2 algorithms Recursive Feature Elimination and Linear Discriminant Analysis where chosen to perform feature scaling as a stack. First 17 columns from the dataset (which was having 28 features) were selected by Recursive Feature Elimination and then from those 17 features, 11 best features were selected by Linear Discriminant Analysis and those 11 features were passed as an argument to the function *acuu().*

**[ Note: Every feature selection technique except correlation matrix is applied on dropped correlations dataset (28 features). ]**

Below, all the results from the function *acuu()* are given:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Classifier Used* Line arrow Straight**  ***Method Used*Line arrow Straight** | | **SVM** | **Perceptron** | **KNN** | **SGD** | **XGBoost** |
| After Dropping Correlations | ***No. of Features*** | 78.53% | 78.28% | 99.88% | 92.31% | 10.23% |
| 28 |
| By Using Univariate Feature Selection | 17 | 78.85% | 80.65% | 99.86% | 92.21% | 22.29% |
| By Using RFE | 17 | 58.32% | 88.8% | 99.9% | 70.16% | 20.16% |
| By Using PCA | 17 | 22.6% | 22.17% | 99.87% | 24.97% | 8.2% |
| By Using LDA | 17 | 98.08% | 79.54% | 99.87% | 99.02% | 77.79% |
| By Using Univariate Feature Selection | 11 | 79.35% | 79.19% | 99.84% | 88.46% | 18.99% |
| By Using RFE | 11 | 27.78% | 27.79% | 99.28% | 19.74% | 76.86% |
| By Using PCA | 11 | 22.16% | 22.17% | 99.87% | 24.98% | 11.02% |
| By Using LDA | 11 | 97.73% | 79.08% | 99.87% | 98.98% | 74.8% |

**TABLE : Accuracies by using method *sklearn.metrics.*accuracy\_score():**

**9.**

**8.**

**7.**

**6.**

**5.**

**4.**

**3.**

**2.**

**1.**

**Sr. No.**

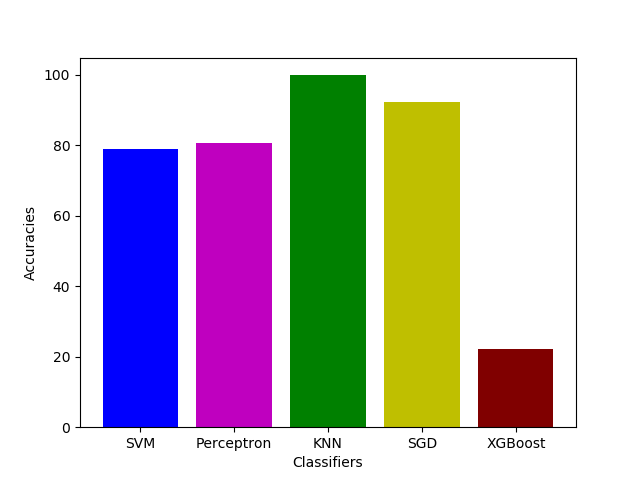
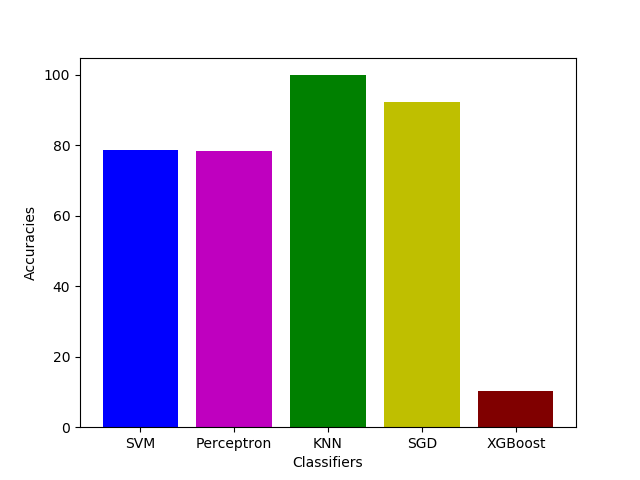
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| By using all the selection algorithms as a stack | 11 | 80.26% | 74.01% | 99.91% | 98.5% | 59.44% |
| By using RFE and LDA as a stack | 11 | 92.43% | 79.26% | 99.88% | 98.95% | 75.49% |

**10**

**11**

**[ Note: Every feature selection technique except correlation matrix is applied on dropped correlations dataset (28 features). ]**

**Graphs of accuracies returned by accuracy\_score() function:**

****

**Fig.2** By Using UFS (17 features) plt.bar(x,y,color=['b','m','g','y', 'maroon','cyan'])

plt.xticks(x)

plt.xlabel("Classifiers")

plt.ylabel("Accuracies")

plt.show()

**Fig.1** After Dropping Correlations (28 features) plt.bar(x,y,color=['b','m','g','y', 'maroon','cyan'])

plt.xticks(x)

plt.xlabel("Classifiers")

plt.ylabel("Accuracies")

plt.show()

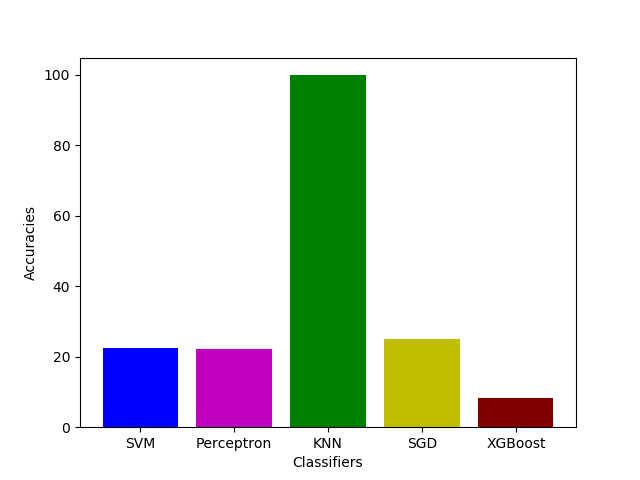
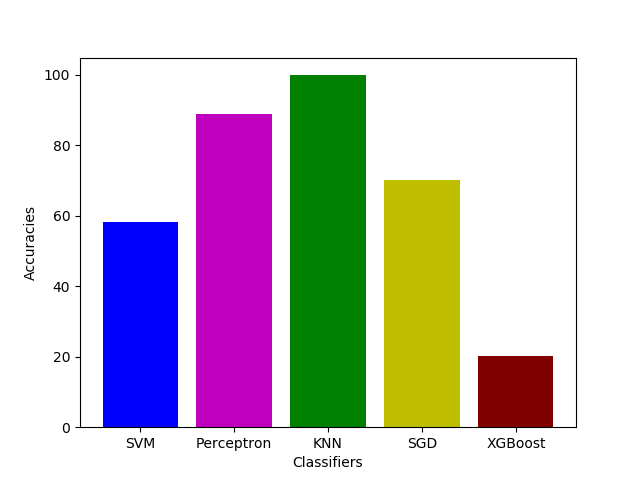


Fig.4 By Using PCA (17 features)

Fig.3 By Using RFE (17 features)

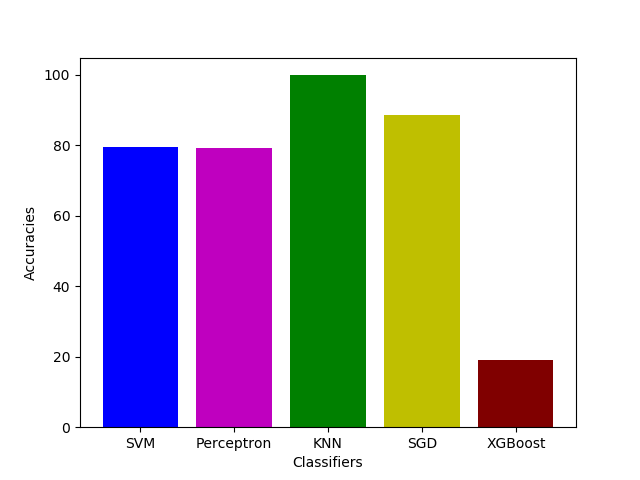
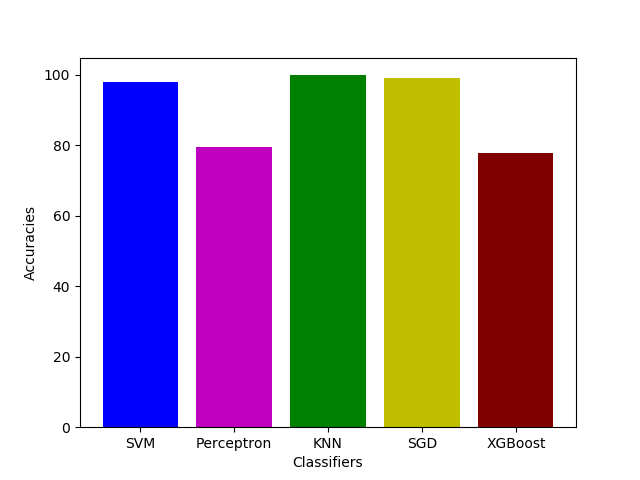


Fig.6 By Using UFS (11 features)

Fig.5 By Using LDA (17 features)

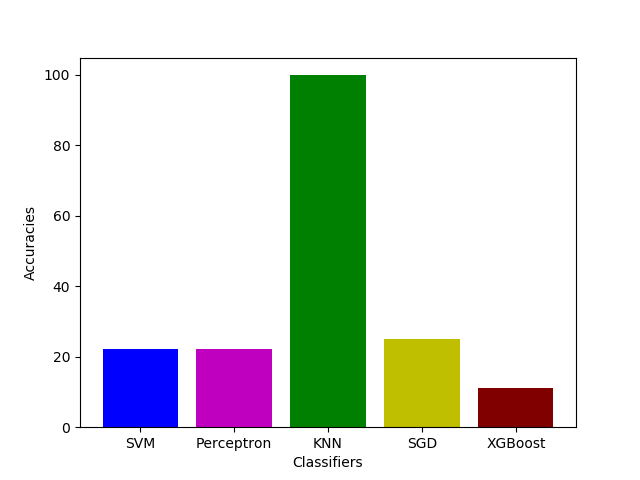
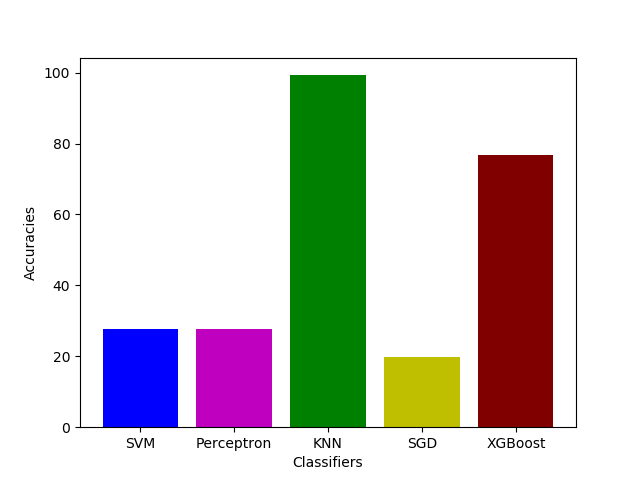


Fig.8 By Using PCA (11 features)

Fig.7 By Using RFE (11 features)

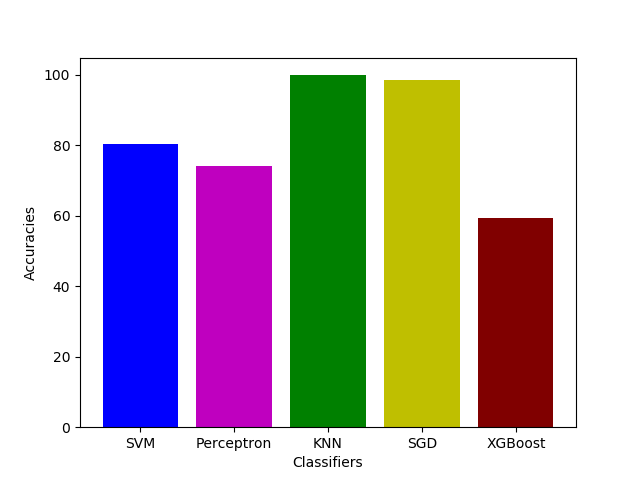
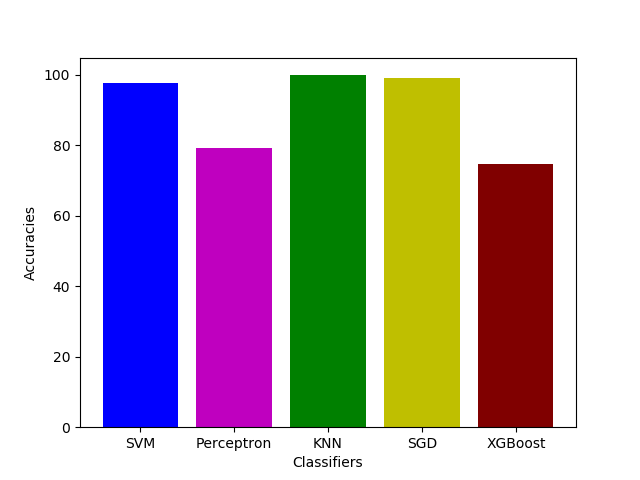


Fig.10 By Using all algorithms as a stack (11 features)

Fig.9 By Using LDA (11 features)

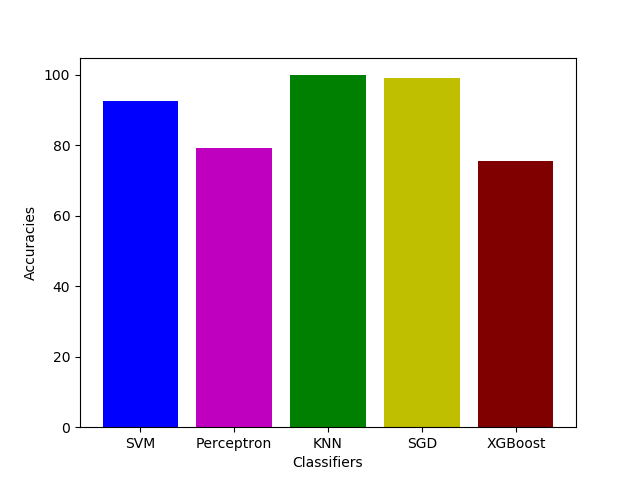
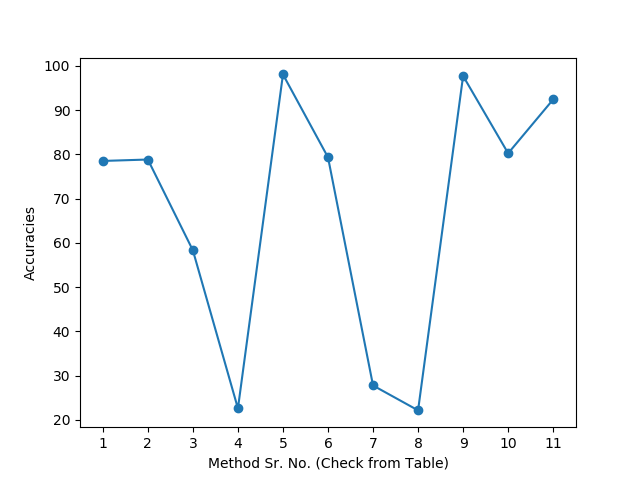


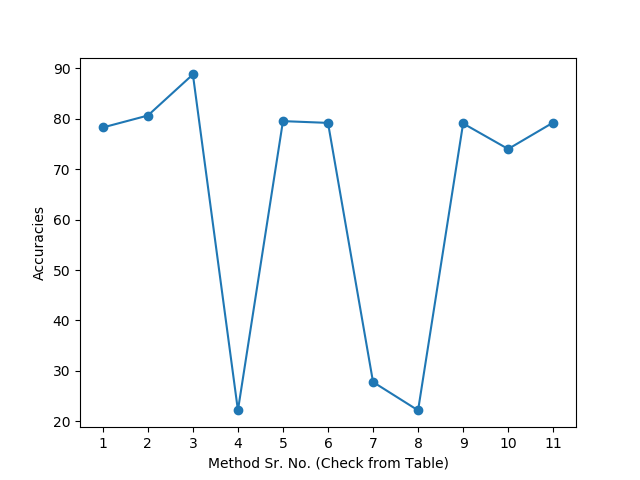
Fig.11 By Using RFE and LDA as stack (11 features)

**Visualization of increase or decrease in the accuracies of a classifier by using different feature selection techniques** **–**



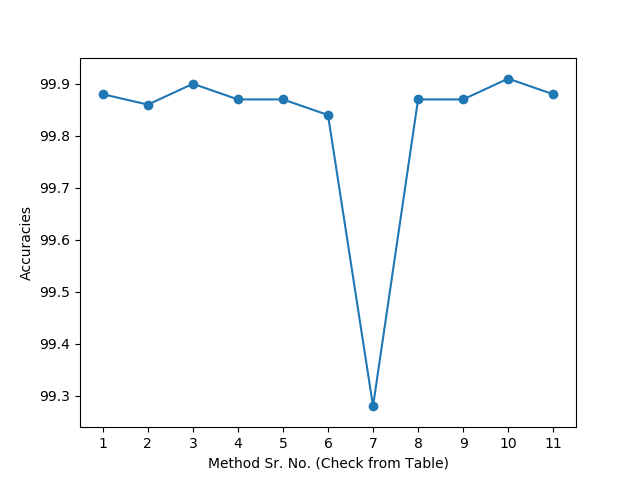
**SVM**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



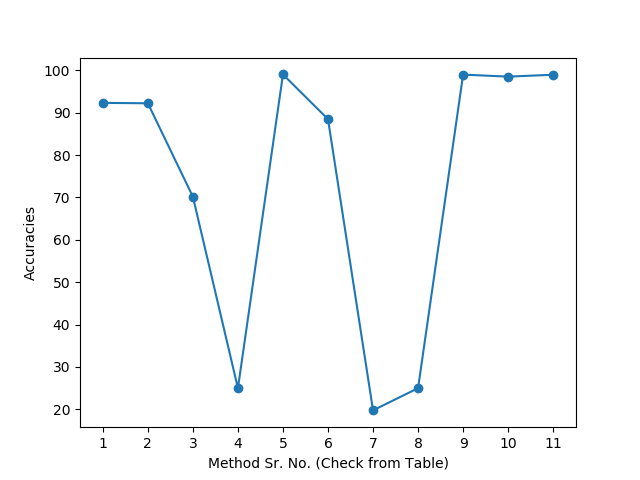
**PERCEPTRON**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



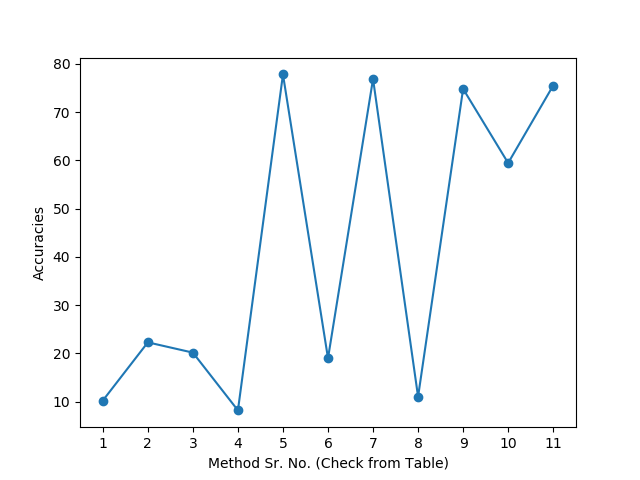
**KNN**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



**SGD**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



**XGBoost**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Classifier Used* Line arrow Straight**  ***Method Used*Line arrow Straight** | | **SVM** | **Perceptron** | **KNN** | **SGD** | **XGBoost** |
| After Dropping Correlations | ***No. of Features*** | 77.82% | 25.44% | 99.82% | 22.32% | 99.96% |
| 28 |
| By Using Univariate Feature Selection | 17 | 63.77% | 25.41% | 99.81% | 26.79% | 99.93% |
| By Using RFE | 17 | 86.25% | 28.43% | 99.87% | 29.34% | 99.94% |
| By Using PCA | 17 | 96.5% | 23.33% | 99.82% | 12.74% | 99.94% |
| By Using LDA | 17 | 99.18% | 98.31% | 99.79% | 98.87% | 99.86% |
| By Using Univariate Feature Selection | 11 | 91.35% | 27.44% | 99.79% | 23.57% | 99.91% |
| By Using RFE | 11 | 98.82% | 87.54% | 99.11% | 88.84% | 99.43% |
| By Using PCA | 11 | 95.93% | 23.33% | 99.82% | 7.34% | 99.9% |
| By Using LDA | 11 | 99.16% | 98.58% | 99.79% | 98.48% | 99.84% |

**TABLE 2 : K-Fold Cross Validation -**

MODEL EVALUATION :-

**9.**

**8.**

**7.**

**6.**

**5.**

**4.**

**3.**

**2.**

**1.**

**Sr. No.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| By using all the selection algorithms as a stack | 11 | 99.13% | 98.62% | 99.86% | 98.63% | 99.87% |
| By using RFE and LDA as a stack | 11 | 99.16% | 98.59% | 99.81% | 98.65% | 99.82% |

**10**

**11**

**Graphs of accuracies by K-Fold Cross Validation:**

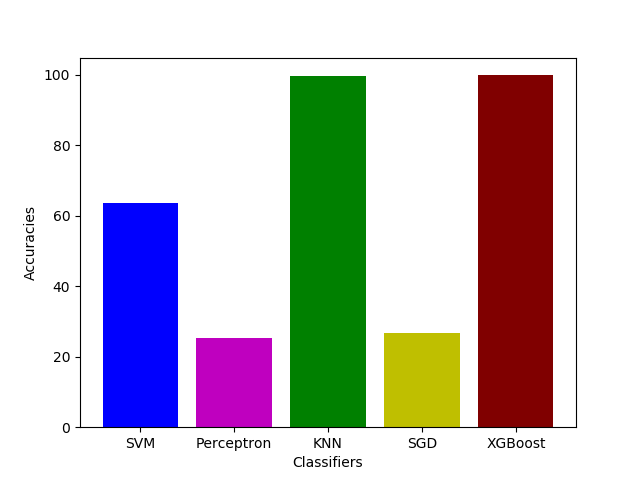
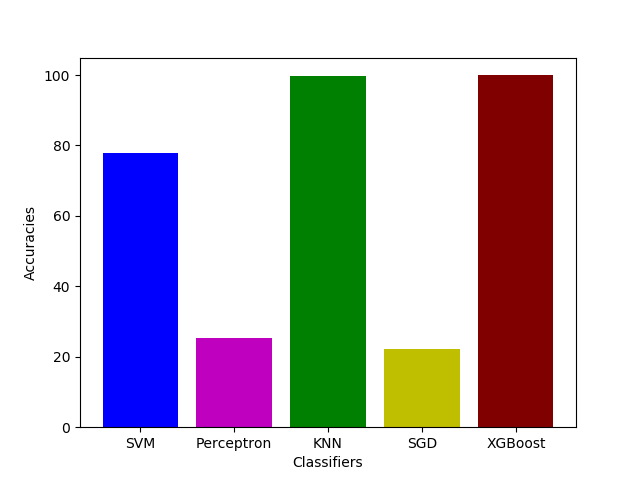


Fig.2 By using UFS (17 features)

Fig.1 After dropping Correlations(28 features)

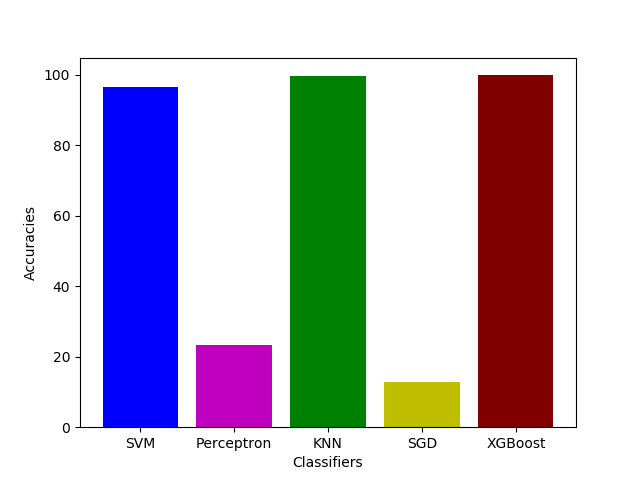
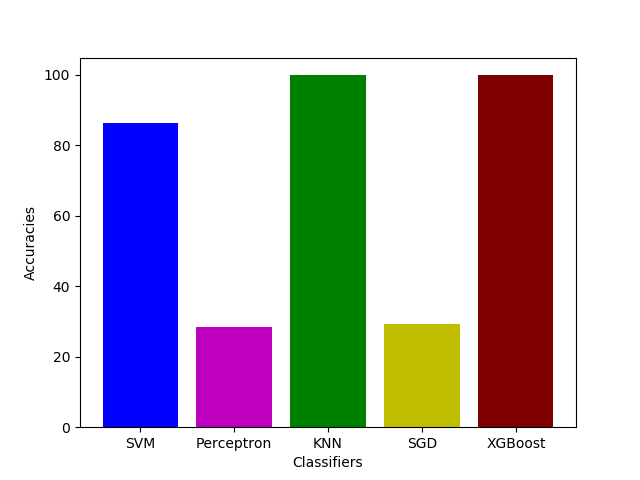


Fig.4 By using PCA(17 features)

Fig.3 By using RFE(17 features)

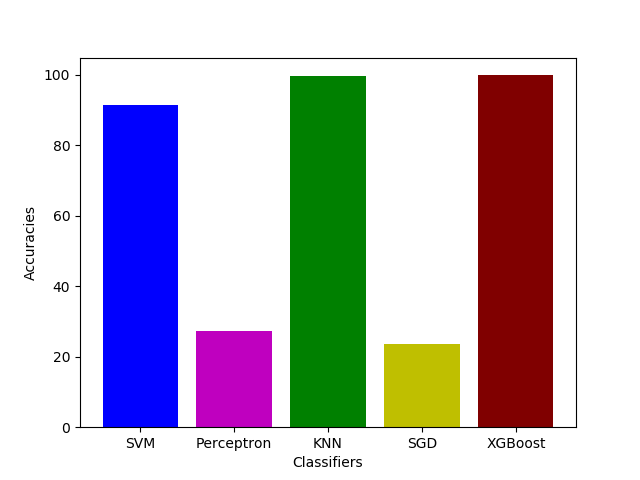
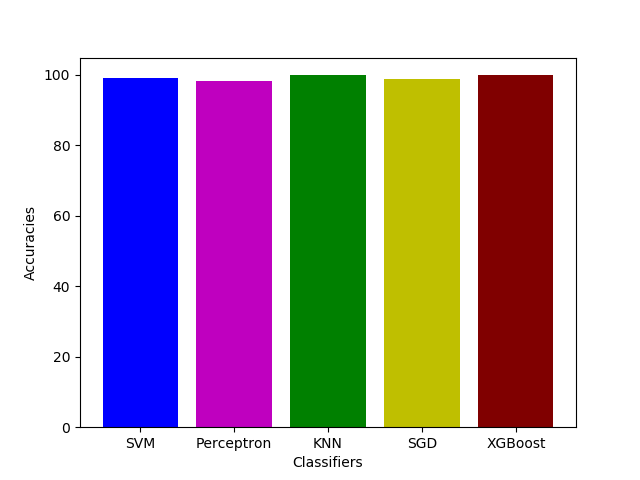


Fig.6 By using UFS(11 features)

Fig.5 By using LDA(17 features)

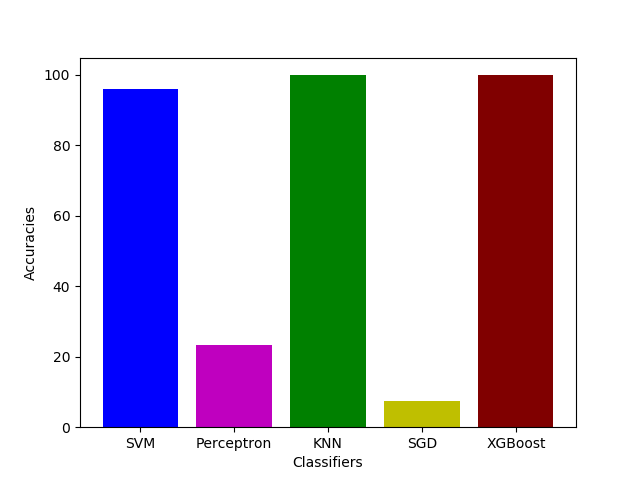
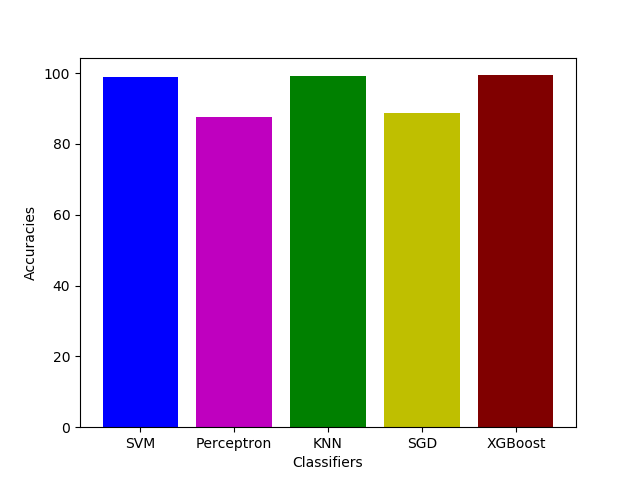


Fig.8 By using PCA(11 features)

Fig.7 By using RFE(11 features)

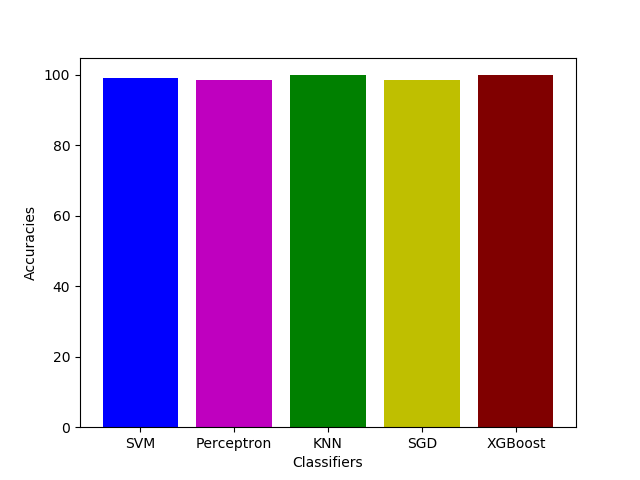
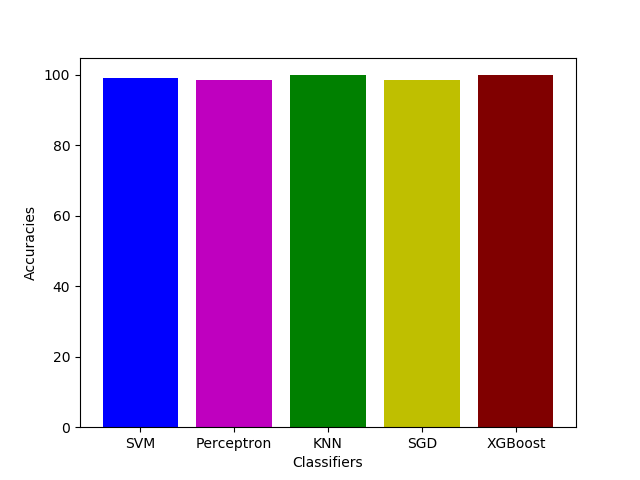


Fig.10 BY suing all the algorithms as a stack (11 features)

Fig.9 By using LDA(11 features)

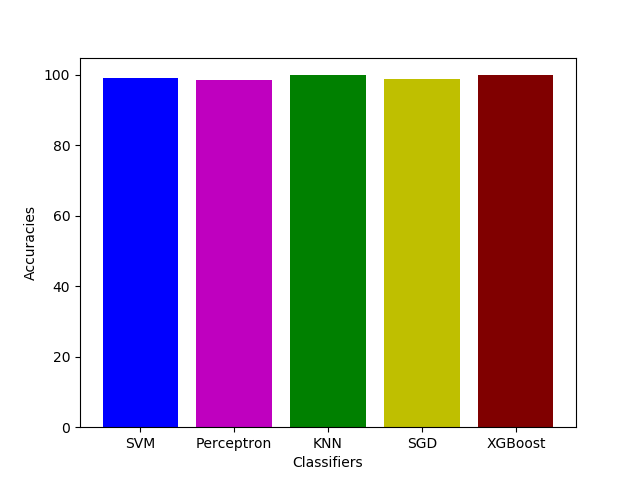
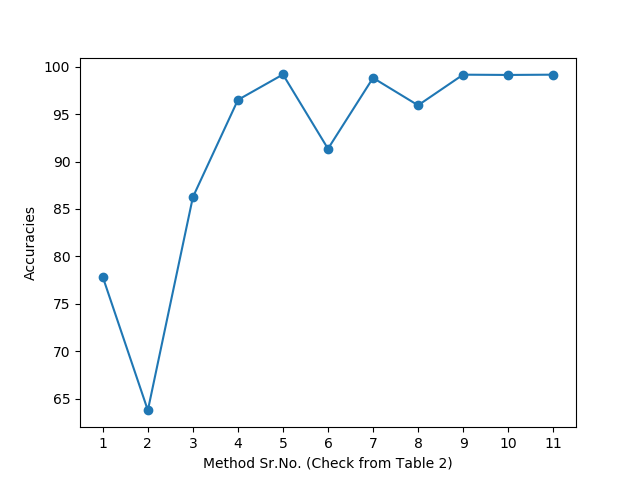


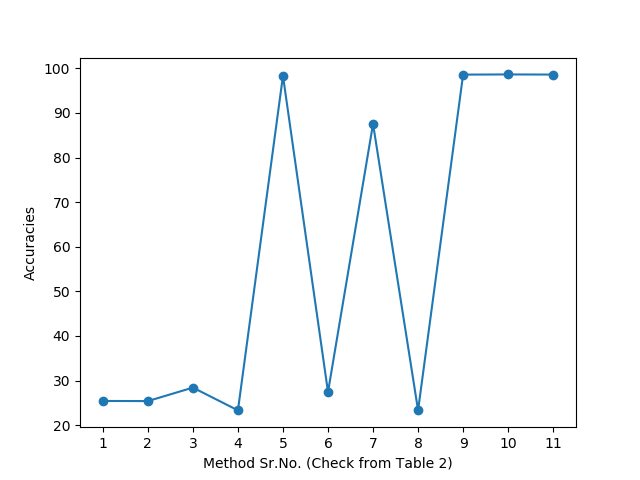
Fig.11 By RFE and LDA as a stack(11 features)

**Visualization of increase or decrease in the accuracies of a classifier by using different feature selection techniques** **(in K-fold Cross Validation) –**



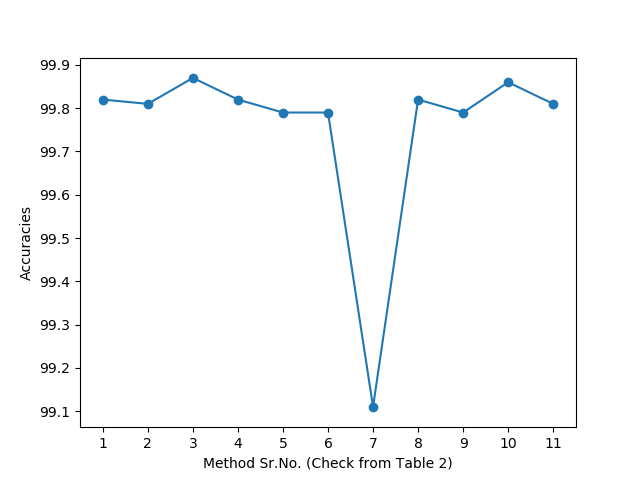
**SVM**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



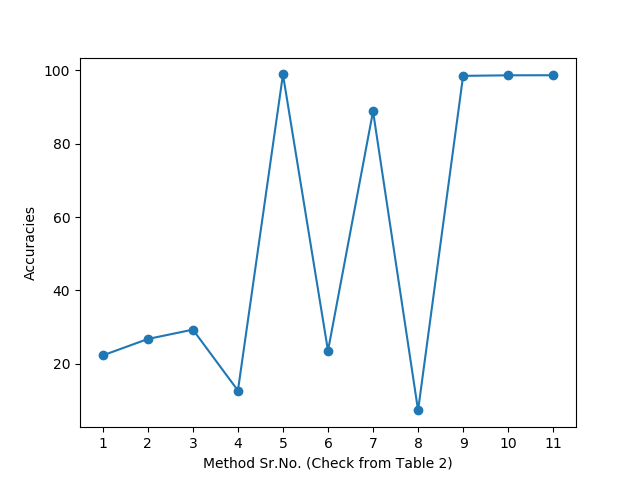
**Perceptron**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



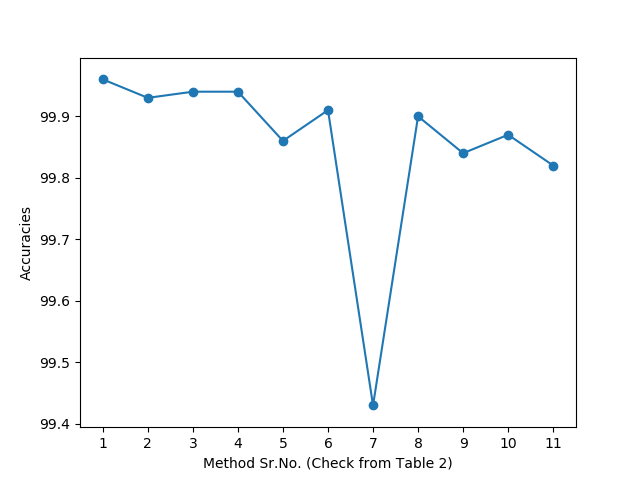
**KNN**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



**SGD**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)



**XGBoost**

1. After Deleting Correlation (28 Features)
2. UFS (17 Features)
3. RFE (17 Features)
4. PCA (17Features)
5. LDA (17Features)
6. UFS (11 Features)
7. RFE (11 Features)
8. PCA (11 Features)
9. LDA (11 Features)
10. All as stack (11 Features)
11. RFE and LDA as stack (11 Features)

References:

[1] Aakansha Patel, Santosh Sammarvar, Amar Naik Department of Information technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal(M.P.),India. Data Mining Vs Statistical Techniques for Classification of NSL-KDD Intrusion Data <https://pdfs.semanticscholar.org/f224/ce6a9b5605fffc2e6894d19d67918c0e8b92.pdf>

[2] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani . A Detailed Analysis of the KDD CUP 99 Data Set

[3] Uma Subramanian, Hang See Ong , Department of Electronics and Communication Engineering, College of Engineering, University Tenaga Nasional, Jalan IKRAM - UNITEN, Kajang, Selangor, 43000, Malaysia. Systematic Review and Comparison of Anomaly Based Network Intrusion Detection Systems Based on Efficiency

[4] Md. Al Mehedi Hasan, Mohammed Nasser, Shamim Ahmad, Khademul Islam Molla Department of Computer Science & Engineering, University of Rajshahi, Rajshahi, Bangladesh. Feature Selection for Intrusion Detection Using Random Forest

[5] Herve Nkiama, Syed Zainudeen Mohd Said, Muhammad Saidu, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 4, 2016. A Subset Feature Elimination Mechanism for Intrusion Detection System

[6] P. Ghosh, C. Debnath, and D. Metia, “An Efficient Hybrid Multilevel Intrusion Detection System in Cloud Environment,” IOSR J. Comput. Eng., vol. 16, no. 4, pp. 16–26, 2014.

[7] Vinutha H.P.1, Dr.Poornima B2 Assistant Professor, Dept. of CS&E, BIET, Davangere, Karnataka, India1 Professor& Head, Dept. of IS&E, BIET, Davangere, Karnataka, India2. International Journal of Engineering and Manufacturing Science. ISSN 2249-3115 Volume 8, Number 1 (2018) pp. 167-175 © Research India Publications. Analysis of Feature selection algorithms for Naïve Bayes classifier using NSL-KDD

[8] Zahra Karimi, Mohammad Mansour Raihi Kashani, Ali Harounabadi, International Journal of Computer Applications (0975 – 8887) Volume 78 – No.4, September 2013. Feature Ranking in Intrusion Detection Dataset using Combination of Filtering Methods.

[9] Sevcan YILMAZ GÜNDÜZ , Muhammet Nurullah ÇETER, Computer Engineering, Faculty of Engineering, Anadolu University, Eskişehir, Turkey , Anadolu Üniversitesi Bilim ve Teknoloji Dergisi A- Uygulamalı Bilimler ve Mühendislik Anadolu University Journal of Science and Technology A- Applied Sciences and Engineering . FEATURE SELECTION AND COMPARISON OF CLASSIFICATION ALGORITHMS FOR INTRUSION DETECTION

[10] Dr. Tanuja Dhope, Ashish K. Uplenchwar et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (4) , 2015, 3778-3783, Intrusion Detection System (IDS) framework for Digital Network

[11] M.Judith Lucia1 and T.Thirunavukarasu 2 , 1 PG scholar & 2 Assistant Professor Department of Information Technology, SNS College of Technology, Coimbatore, India, M.Judith Lucia et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014, 3190 – 3195, Rapid Decentralized Network Intrusion Defense System on Multiple Virtual Machines

<https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029>

<https://machinelearningmastery.com/k-fold-cross-validation/>