Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

1. Abstract:

In this project, we will be applying **Multinomial Logistic Regression**, **Naïve Bayes Classifier** and **Neural Network algorithms** for solving a multi-class classification problem, along with **Principal Component Analysis (PCA)** as a pre-processing step. We will apply the above machine learning algorithms on a simulated dataset first, and then these algorithms will be applied on a real-world dataset. The goal of this project is to implement all three algorithms correctly, and provide a comparative analysis based on the following machine learning evaluation metrics:

- Bias/Variance Tradeoff: We will use Normalized-Test-MSE to determine the Bias/Variance Tradeoff for each algorithm as in Homework-2.
- Accuracy, Precision and Recall: We will determine the accuracy metric for each algorithm to evaluate performance as in Homework-4. The performance metrics will be extended to determine Precision and Recall also.
- F-Measure (F1 Score): We will determine the F1 Score based on the Precision (Pr) and Recall (Rc) values for each algorithm as per below equation:

$$F1 = \left(\frac{2}{\left(\frac{1}{Pr}\right) + \left(\frac{1}{Rc}\right)}\right)$$

2. Introduction

With the rapid development of the smart grid and increasingly integrated communication networks, power grids are facing serious cyber-security problems. For example, data injection attacks represent a serious threat to the power system as they could affect its ability to consistently produce energy, prevent the operators of the system from obtaining its true operating conditions, and cause a power imbalance between supply and demand. Coordination between different attacks, whether they belong to the cyber or the physical domain, can definitely bring higher impacts on the power system as evidenced by the Ukrainian power system cyberattack in 2015. Detection and prevention of such attacks becomes paramount to ensure the security of power systems in order to avoid critical consequences of their failure. Several methods can be employed to perform the detection/prevention of those attacks, such as: firewalls, network security monitoring tools, encryption tools, web vulnerability scanning tools, antivirus software, intrusion prevention systems, intrusion detection systems, etc. Intrusion detection systems (IDS) can either be signature-based or anomaly-based. Anomaly-based IDS proved to be more effective as they rely on using Machine Learning (ML) techniques to train the detection system to recognize a normalized baseline; the baseline being how the system normally behaves. Then, any out-of-the-ordinary behavior flagged by the system would trigger an alert.[1]–[3]

This project focuses on formulating machine learning algorithms that could be exploited by anomaly-based IDS to perform accurate decisions when detecting attacks on power systems. The selected machine learning techniques will be discussed in the next sections along with some performance metrics to compare them.

Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

3. Problem Formulation

Objective: Formulate Machine Learning algorithms to help intrusion detection systems and anomaly detection systems in making accurate decisions when detecting attacks on Industrial Control Systems/Power Systems.

Data: For the purpose of this project, we have considered using datasets relevant to power systems. The dataset selected is an Industrial Control System Cyber Attack Dataset [4]obtained through a general search on Google's Dataset Search Engine. It was created by Uttam Adhikari, Shengyi Pan, and Tommy Morris in collaboration with Raymond Borges and Justin Beaver of Oak Ridge National Laboratories (ORNL).

This dataset includes measurements related to electric transmission system normal, disturbance, control, cyber-attack behaviors. Measurements in the dataset include synchrophasor measurements and data logs from Snort (an Intrusion Detection tool), a simulated control panel, and relays.

The dataset has a total of **128 features** (representing various measurement parameters) and **3 class labels** (No Event, Attack and Natural). The dataset has a total of **712832 datapoints**.

This dataset has been used for multiple works related to power system cyber-attack classification.

4. Proposed Method with Details:

- Synthetic Data Generation: We will generate our own training and testing data first. We will use a synthetic data generation process as in Homework-3. Dependent variables will be generated from a multivariate gaussian distribution to account for the fact that the features are correlated.

- Learning Parameters using Training Data: We will write the code and the necessary wrapper functions for Multinomial Logistic Regression, Naïve Bayes Classifier, and Neural Network algorithms. We will use the training data to train the respective models.

- Testing using Test Data: We will utilize the test data that we have generated to determine the evaluation metrics for each model. We will provide a comparative analysis for all three algorithms.

- Real Data: After we have trained and tested our models using synthetic data, we will test our models on a real dataset and evaluate the performance metrics. The dataset will be imported as a ‘csv’ file. The first step will be to read the data and get an overview of it. We will organize the dataset and remove the features that are not significant to our goal.

- Pre-processing using PCA: We will implement PCA as a dimensionality reduction technique to construct relevant features through non-linear (kernel PCA) combinations of the original features. We think that using PCA to project the data onto orthogonal principal components will improve the performance of our model.

- Training, Testing & Validation on Real Data: We will split the real data into training, testing and validation sets.

- Reporting Results and Observations: We will provide an analysis of the evaluation metrics mentioned above for all three algorithms.

5. Solution Approach

First of all, we will attempt to clean up our data using a pre-processing technique called PCA: Principal Component Analysis. This will allow us to account for the most important features while still preserving the quality of the data.

Then, we will be selecting three algorithms for the purpose of this project. Those three algorithms are well suited for our project because they specifically cater to classification problems:

- *Naïve Bayes Classifier* is a probabilistic classifier based on Bayes Theorem with strong independence assumptions between features. We can decompose the conditional property by using Bayes' theorem as following:

  $$P(C_k \mid X) = \frac{P(X \mid C_k)\, P(C_k)}{P(X)}$$

  Where X = (x_1; ….; x_n) represents a vector of n independent features and C_k represents each class. Assuming that features are not correlated with each other is not a true assumption in many problems and it can conversely affect the accuracy of the classifier. The main advantage of Naive Bayes is that it is an online algorithm, and its training can be completed in linear time.

- *Multinomial Logistic Regression* is a classification method that uses the main idea of logistic regression to classify multiclass problems. Logistic regression is a predictive analysis like other regression analyses. Logistic regression can describe data and explain the relationship between features and classes.

- *Neural Networks* are designed to recognize patterns which they use to either cluster or classify unlabeled data.

After implementing those three algorithms, we are going to evaluate how well they are performing on the data using three metrics:

- Bias/Variance Tradeoff: We will use Normalized-Test-MSE to determine the Bias/Variance Tradeoff for each algorithm as in Homework-2.

- Accuracy, Precision and Recall: We will determine the accuracy metric for each algorithm to evaluate performance as in Homework-4. The performance metrics will be extended to determine Precision and Recall also.
- Precision (Pr) or Positive Predictive value: It is the ratio of correctly classified attacks flows (TP), in front of all the classified flows (TP+FP).

- Recall (Rc) or Sensitivity: It is the ratio of correctly classified attack flows (TP), in front of all generated flows (TP+FN).

- F-Measure (F1 Score): We will determine the F1 Score based on the Precision (Pr) and Recall (Rc) values for each algorithm as per below equation:

$$F1 = \left(\frac{2}{\left(\frac{1}{Pr}\right) + \left(\frac{1}{Rc}\right)}\right)$$

6. Algorithms:

- ***Generating Simulated Dataset:***

First, we have defined a method generate_data(). This method is used to generate synthetic data for our experiments following the procedure mentioned in HW-3 and HW-6b.

For generation of Synthetic Data for training and testing of Logistic Regression, Naïve Bayes and Neural Network, we use the GDA model because that is the only one which provides a generative model.
For a classification problem,
$y^{(i)} = \{0, 1, 2\}$ ~ we have k = 3 classes.
Hence, we have phi_1, phi_2 and phi_3 = 0.3, 0.4 and 0.3 respectively.
$x^{(i)} \in R^{d+1}$ ~ Gaussian Distribution ($\mu^{y(i)}$ = mean and Σ = covariance)

n = (number of datapoints) = 100
d = (number of features) = 10
e = (error or noise)
$\Sigma_x = tmp * evals * tmp'$
where,
tmp = randn(d, d)
evals = diag([100 ∗ randn(round(d/8), 1)2; randn(d − round(d/8), 1)2])
Using Python and Numpy function, n number of independent training data has been generated.
Where $x = \mu_x + (tmp * sqrt(evals) * tmp') * randn(d, 1)$

After generating the dataset, we have normalized the data using the min_max_scaler() function. This normalized the data within [0, 1].

Then, we have used the train_test_split() function to first divide the data into training set (80%) and validation set (20%). Then, we have divided the training set into training data (70%) and testing data (30%).

- ***Calculating Metrics:***

Next, we have defined a method calculate_metrics(). This method is used to determine the performance metrics for each of the three algorithms that we have defined earlier in this report. The metrics that are calculated in this method are : accuracy, precision, recall, and F1 score. We have used the sklearn.metrics() functions to determine the metrics.

- ***Multinomial Logistic Regression:***

Next, we have defined a method logistic_regression(). This method is used to perform logistic regression on both the simulated dataset and the real dataset.
We have used the class sklearn.linear_model.LogisticRegression() to implement multi-class logistic regression using the ‘lbfgs’ solver.
Solver is the algorithm to use in the optimization problem. Default is ‘lbfgs’. We tested other solvers as well but obtained the best results using ‘lbfgs’.

After defining the logistic regression model, we then fit the model according to the given training data.
Then, we have determined the predicted class labels for the samples in X.
Finally, we have called the calculate_metrics() method using the predicted labels and test labels to determine the performance metrics.

We run this algorithm twice for each simulated data and real data – once without PCA, and then with PCA.

- ***Naïve Bayes Classifier:***

  Next, we have defined a method naïve_bayes(). This method is used to perform naïve bayes algorithm on both the simulated dataset and the real dataset.
  We have used the class sklearn.naive_bayes.MultinomialNB() to implement multi-class naïve bayes.
  After defining the naïve bayes model, we then fit the model according to the given training data.
  Then, we have determined the predicted class labels for the samples in X.
  Finally, we have called the calculate_metrics() method using the predicted labels and test labels to determine the performance metrics.

  We run this algorithm twice for each simulated data and real data – once without PCA, and then with PCA.

- ***Neural Network:***

  Next, we have defined a method neural_network(). This method is used to perform neural network algorithm on both the simulated dataset and the real dataset.
  We have used the neural network class functions from keras to implement multi-class neural network.
  First, we have defined the neural network model that we wish to implement. Our neural network model has 2 hidden layers, each with 1000 neurons. The output layer has 3 neurons, as there are three class labels.
  Now we need to define the loss function according to our task. We have used ‘mean_squared_error’ as our loss function.
  We also need to specify the optimizer to use with learning rate and other hyperparameters of the optimizer. We have used the ‘adam’ algorithm as the optimizer. We tested our neural network model with several optimizers, such as SGD, but obtained the best results with ‘adam’.
  Then, we have defined ‘accuracy’ as the metrics that we want our model to calculate.
  After defining the neural network model, we then fit the model according to the given training data. This is the training step of the neural network. Here we need to define the number of epochs for which we need to train the neural network.
  Then, we have determined the predicted class labels for the samples in X.
  Finally, we have called the calculate_metrics() method using the predicted labels and test labels to determine the performance metrics.

  We run this algorithm twice for each simulated data and real data – once without PCA, and then with PCA.

- ***PCA Pre-processing:***

  Next, we have defined a method select_rank_r() to perform PCA analysis as a pre-processing step in our experiments.

We perform PCA to convert high dimensional data to low dimensional data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of variance that they cause in the output.
To perform PCA, our algorithm determines the rank of the matrix, and the rank will be equivalent to the number of principal components.
We have performed PCA as per HW-6a and HW-6b. Then, we determine the Normalized-Test-MSE for all three algorithms – logistic regression, naïve bayes, and neural network. The value of 'r' which has the lowest error is then returned as the number of principal components.

We run this algorithm twice for each simulated data and real data – once without PCA, and then with PCA.
For the real data, we have modified the PCA code from HW-6a and HW-6b by using the sklearn pca methods. The reason for this was to speed up the program runtime when using the real data.

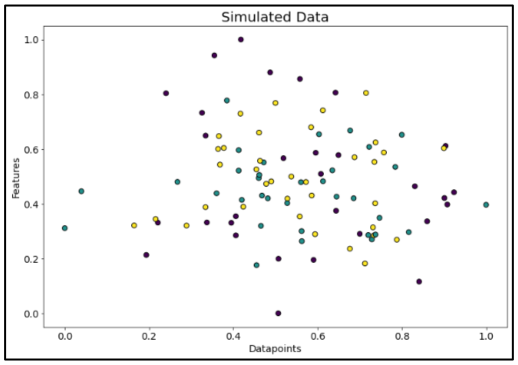
- ***Visualization of results:***

We have used matplotlib() and plotly() classes to visualize the results of all three algorithms - logistic regression, naïve bayes, and neural network.
For each algorithm, we have reported a confusion matrix that provides the performance metric values.
For PCA, we have reported plots for the Testing Data MSE and Accuracy for all three algorithms.
Finally, we have provided a radar chart for comparative analysis of all three algorithms.

7. Experimental Evaluation & Results

- ***Simulated Dataset:***

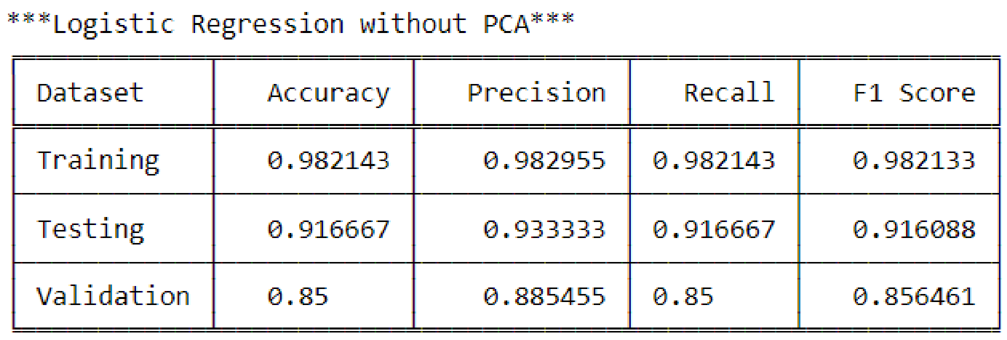
```
***Simulated Dataset***
Number of Datapoints (n) : 100
Number of Features (d) : 10
Shape of X (Input/Features) : (100, 10)
Shape of y (output/class labels) : (100, 1)

***Splitting Dataset into Training and Validation Sets***
Shape of Training Data (X_train, y_train): (80, 10), (80, 1)
Shape of Validation Data (X_val, y_val): (20, 10), (20, 1)

***Splitting Training Dataset into Training and Testing Sets***
Shape of Training Data (X_train, y_train): (56, 10), (56, 1)
Shape of Test Data (X_test, y_test): (24, 10), (24, 1)
```



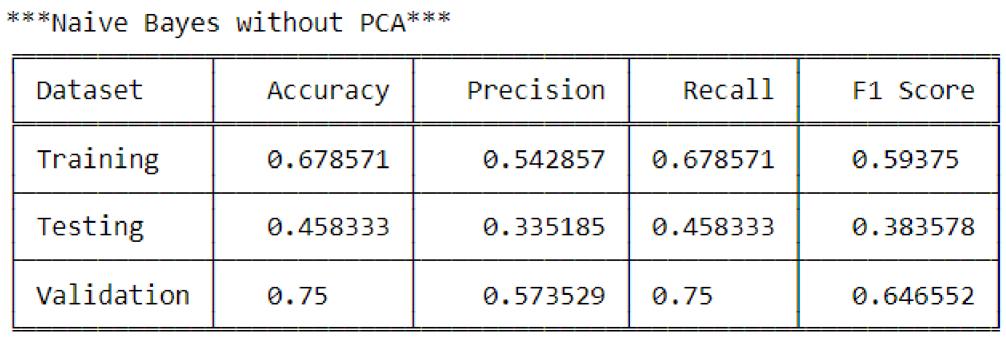
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

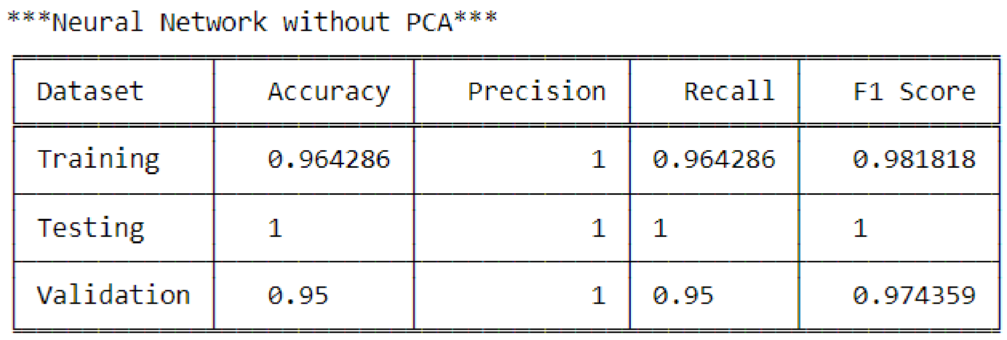
Results for Simulated Data without PCA:


Logistic Regression without PCA

Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.982143	0.982955	0.982143	0.982133
Testing	0.916667	0.933333	0.916667	0.916088
Validation	0.85	0.885455	0.85	0.856461


Naive Bayes without PCA

Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.678571	0.542857	0.678571	0.59375
Testing	0.458333	0.335185	0.458333	0.383578
Validation	0.75	0.573529	0.75	0.646552


Neural Network without PCA

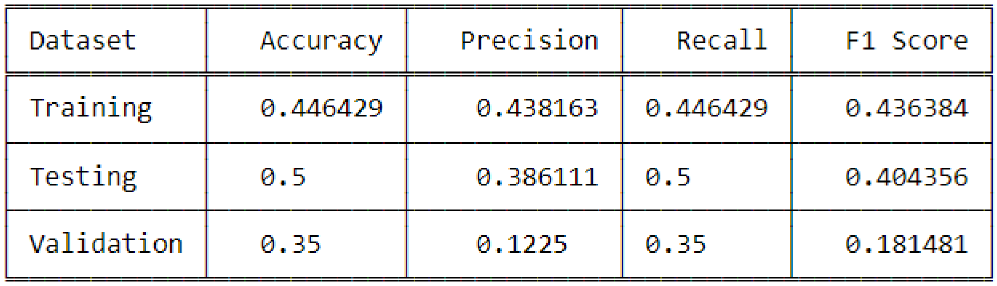
Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.964286	1	0.964286	0.981818
Testing	1	1	1	1
Validation	0.95	1	0.95	0.974359

Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

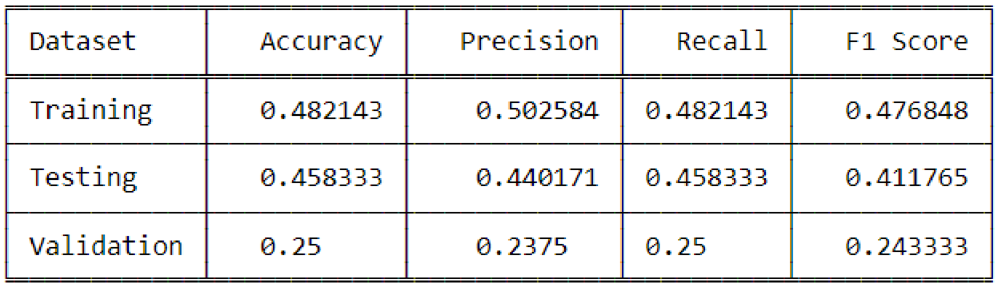
➢ Results for Simulated Data with PCA:

```
***Logistic Regression with PCA***
 Best value of r for Logistic Regression with PCA =  5
 Minimum value of error for Logistic Regression with PCA =  0.75
 Accuracy of Logistic Regression with PCA =  0.5
```



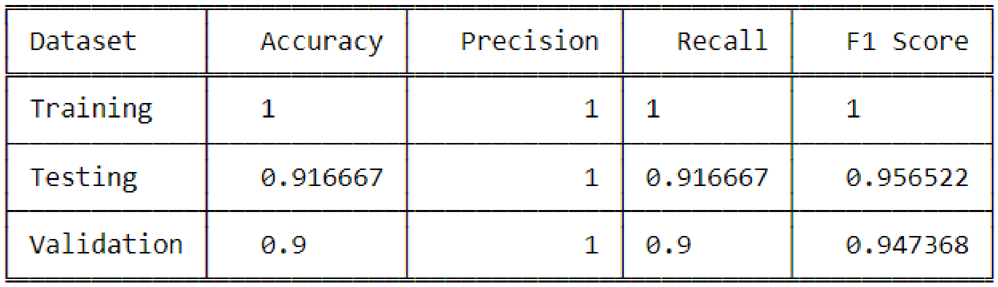
Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.446429	0.438163	0.446429	0.436384
Testing	0.5	0.386111	0.5	0.404356
Validation	0.35	0.1225	0.35	0.181481

```
***Naive Bayes with PCA***
Best value of r for Naive Bayes with PCA =  6
Minimum value of error for Naive Bayes with PCA =  1.2083
Accuracy of Naive Bayes with PCA =  0.4583333333333333
```



Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.482143	0.502584	0.482143	0.476848
Testing	0.458333	0.440171	0.458333	0.411765
Validation	0.25	0.2375	0.25	0.243333

```
***Neural Network with PCA***
Best value of r for Neural Network with PCA =  5
Minimum value of error for Neural Network with PCA =  1.166696
Accuracy of Neural Network with PCA =  0.9
```



Dataset	Accuracy	Precision	Recall	F1 Score
Training	1	1	1	1
Testing	0.916667	1	0.916667	0.956522
Validation	0.9	1	0.9	0.947368

Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

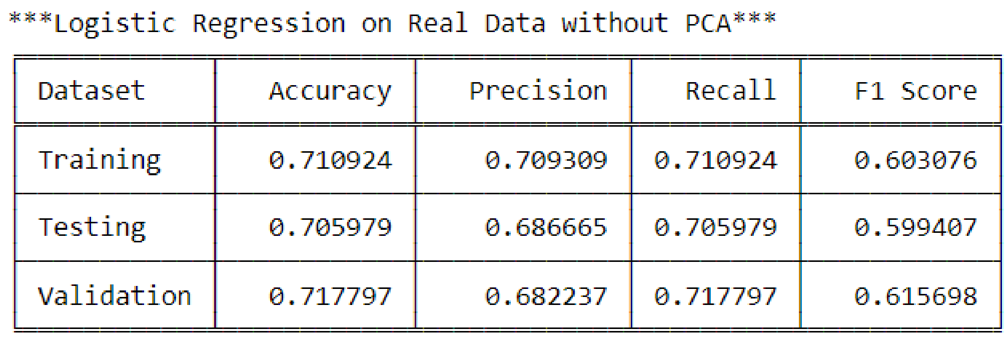
➢ ICS Cyber Attack Dataset (Real Data):

```
***ICS Cyber Attack Dataset***

Original Dataset shape :  (5569, 129)

Displaying first five rows of Data:

   R1-PA1:VH     R1-PM1:V   R1-PA2:VH     R1-PM2:V   R1-PA3:VH     R1-PM3:V  
\
0 -19.858717  130531.4436 -139.847539  130506.3704  100.147293  130606.6634
1 -18.821664  131083.0556 -138.787567  131057.9823  101.195806  131133.2021
2 -18.546644  131333.7883 -138.518277  131308.7150  101.476555  131383.9348
3 -18.254435  131584.5210 -138.231798  131559.4477  101.768764  131634.6675
4 -18.128385  131634.6675 -138.094288  131584.5210  101.883355  131684.8140

   R1-PA4:IH   R1-PM4:I   R1-PA5:IH   R1-PM5:I  ...  control_panel_log4  \
0 -19.572238  419.50501 -143.050373  438.18223  ...                   0
1 -18.867500  420.42056 -141.875809  436.16802  ...                   0
2 -18.781557  420.96989 -141.514846  435.06936  ...                   0
3 -18.735720  421.70233 -141.251285  434.70314  ...                   0
4 -19.033658  423.89965 -141.119505  434.33692  ...                   0

   relay1_log  relay2_log  relay3_log  relay4_log  snort_log1  snort_log2  \
0           0           0           0           0           0           0
1           0           0           0           0           0           0
2           0           0           0           0           0           0
3           0           0           0           0           0           0
4           0           0           0           0           0           0

   snort_log3  snort_log4    marker
0           0           0  NoEvents
1           0           0  NoEvents
2           0           0  NoEvents
3           0           0  NoEvents
4           0           0  NoEvents

[5 rows x 129 columns]

***Removing columns with infinity and NaN values***
Modified Dataset shape :  (5085, 129)
Count of each class labels :  Counter({'Attack': 3586, 'Natural': 1173, 'NoEv
ents': 326})

Shape of X (Input/Features) : (5085, 125)
Shape of y (output/class labels) : (5085,)

***Splitting Dataset into Training and Validation Sets***
Shape of Training Data (X_train, y_train): (4068, 125), (4068,)
Shape of Validation Data (X_val, y_val): (1017, 125), (1017,)

***Splitting Training Dataset into Training and Testing Sets***
Shape of Training Data (X_train, y_train): (2847, 125), (2847,)
Shape of Test Data (X_test, y_test): (1221, 125), (1221,)
```

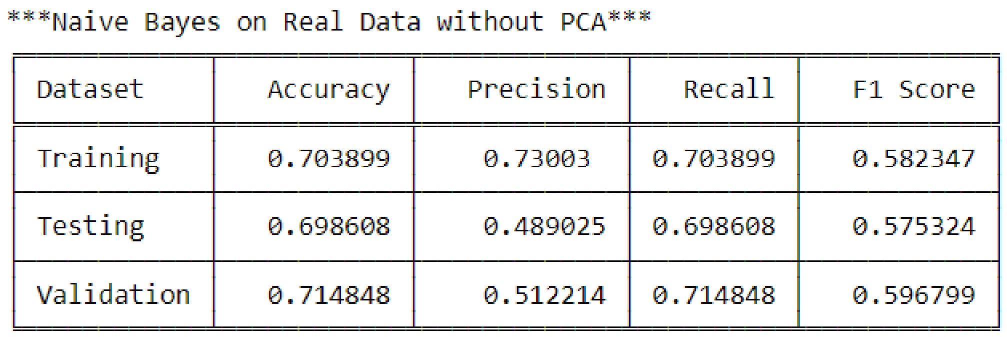
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

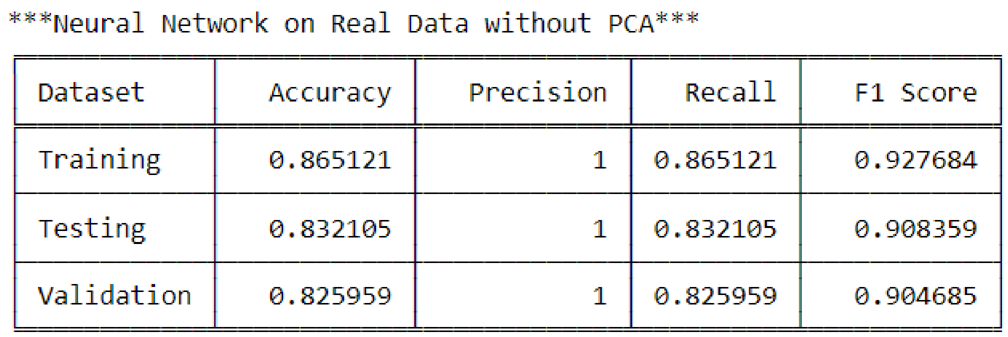
➢ **Results for Real Data without PCA:**


Logistic Regression on Real Data without PCA

Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.710924	0.709309	0.710924	0.603076
Testing	0.705979	0.686665	0.705979	0.599407
Validation	0.717797	0.682237	0.717797	0.615698


Naive Bayes on Real Data without PCA

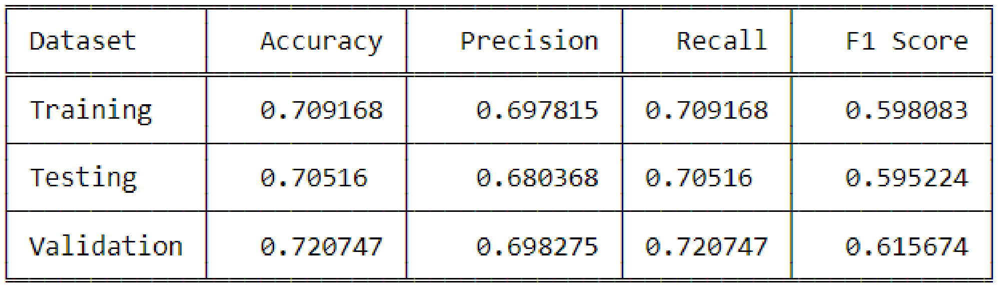
Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.703899	0.73003	0.703899	0.582347
Testing	0.698608	0.489025	0.698608	0.575324
Validation	0.714848	0.512214	0.714848	0.596799


Neural Network on Real Data without PCA

Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.865121	1	0.865121	0.927684
Testing	0.832105	1	0.832105	0.908359
Validation	0.825959	1	0.825959	0.904685

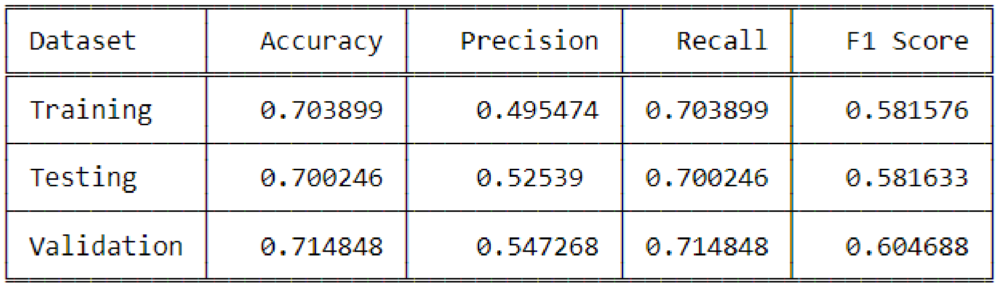
➢ Results for Real Data with PCA:

```
***Logistic Regression with PCA***
Best value of r for Logistic Regression with PCA =  23
Minimum value of error for Logistic Regression with PCA =  0.474201
Accuracy of Logistic Regression with PCA =  0.7059787059787059
```



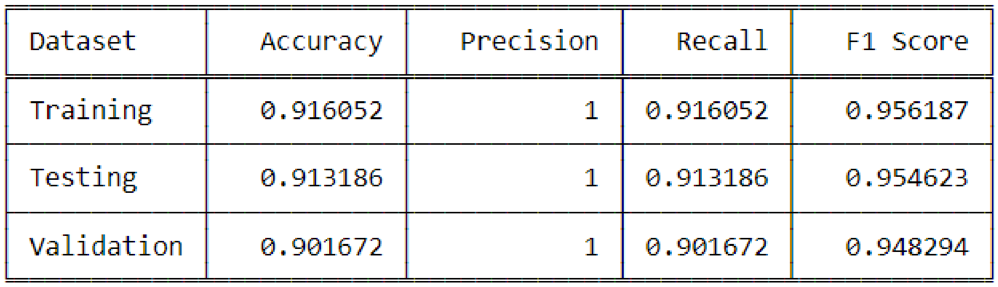
Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.709168	0.697815	0.709168	0.598083
Testing	0.70516	0.680368	0.70516	0.595224
Validation	0.720747	0.698275	0.720747	0.615674

```
***Naive Bayes with PCA***
Best value of r for Naive Bayes with PCA =  22
Minimum value of error for Naive Bayes with PCA =  0.481572
Accuracy of Naive Bayes with PCA =  0.7002457002457002
```



Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.703899	0.495474	0.703899	0.581576
Testing	0.700246	0.52539	0.700246	0.581633
Validation	0.714848	0.547268	0.714848	0.604688

```
***Neural Network with PCA***
Best value of r for Neural Network with PCA =  100
Minimum value of error for Neural Network with PCA =  2.055964
Accuracy of Neural Network with PCA =  0.9131859131859131
```



Dataset	Accuracy	Precision	Recall	F1 Score
Training	0.916052	1	0.916052	0.956187
Testing	0.913186	1	0.913186	0.954623
Validation	0.901672	1	0.901672	0.948294

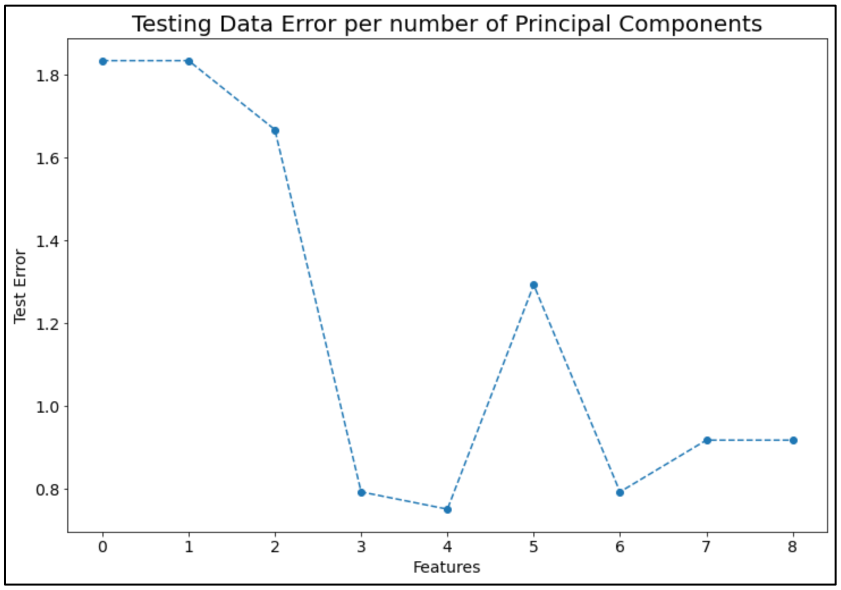
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

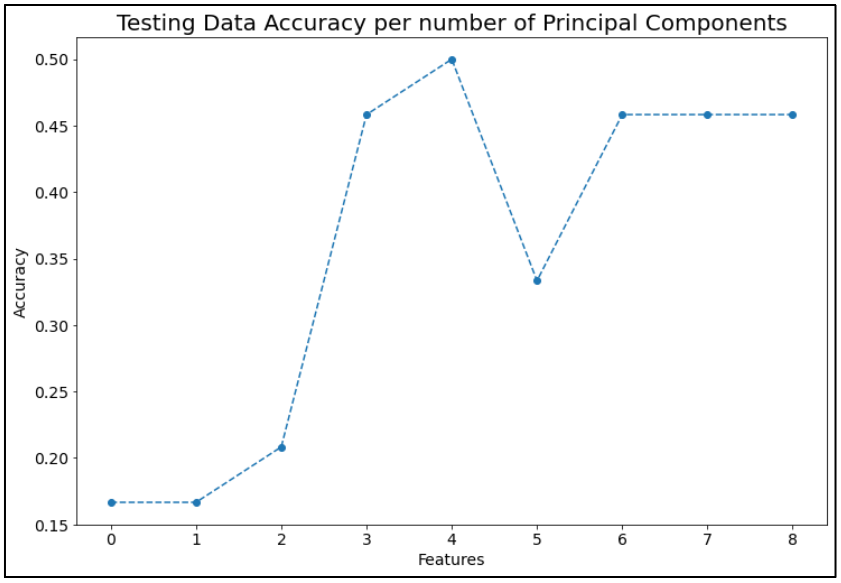
➢ Analysis with PCA:

When PCA is performed on the **simulated dataset**, we get the following results for the testing data:

For Logistic Regression:
Best value of r for Logistic Regression with PCA = 5
Minimum value of error for Logistic Regression with PCA = 0.75
Accuracy of Logistic Regression with PCA = 0.5

Below are the plots for Normalized-Test-MSE and Accuracy of Logistic Regression on simulated data after PCA:



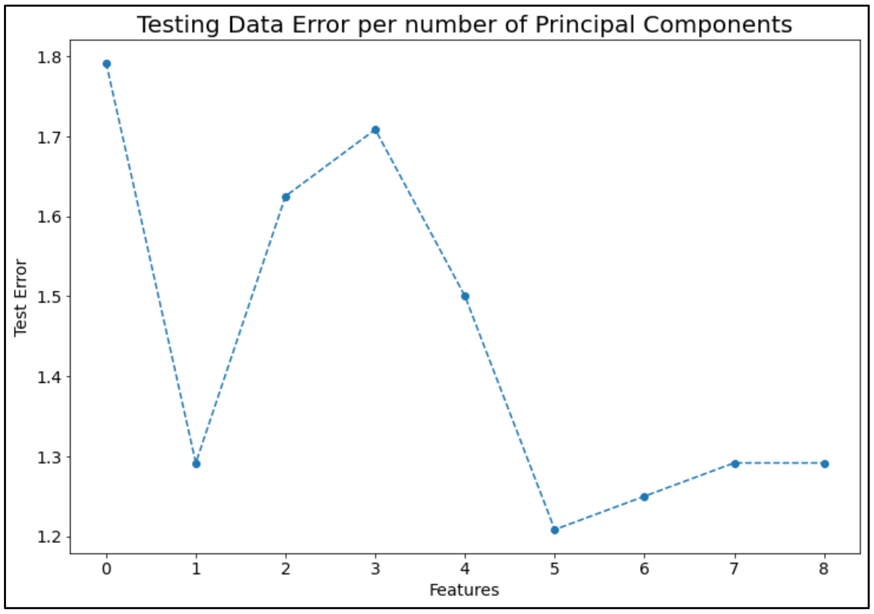


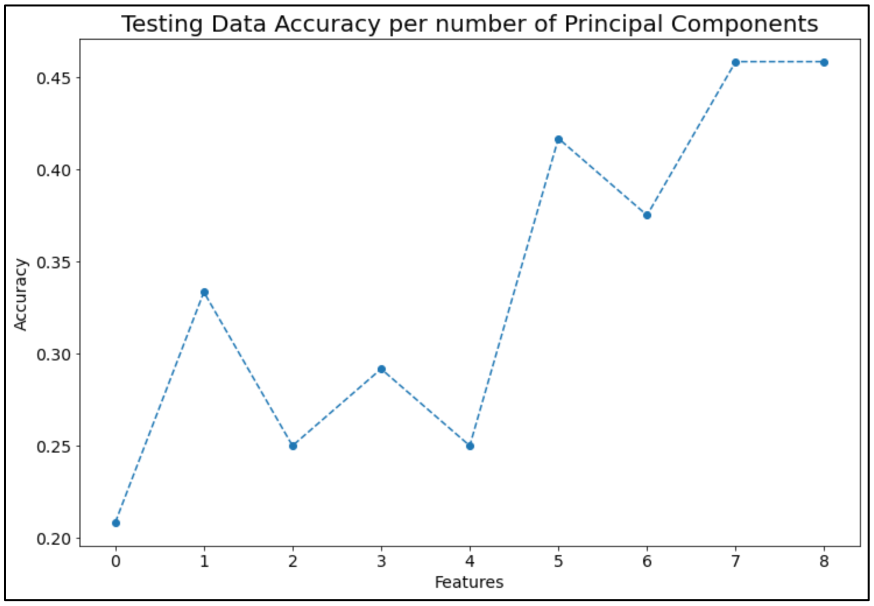
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

<u>EE-425 Project Report</u>

<u>For Naïve Bayes:</u>
Best value of r for Naive Bayes with PCA = 6
Minimum value of error for Naive Bayes with PCA = 1.20
Accuracy of Naive Bayes with PCA = 0.458

Below are the plots for Normalized-Test-MSE and Accuracy of Naïve Bayes on simulated data after PCA:





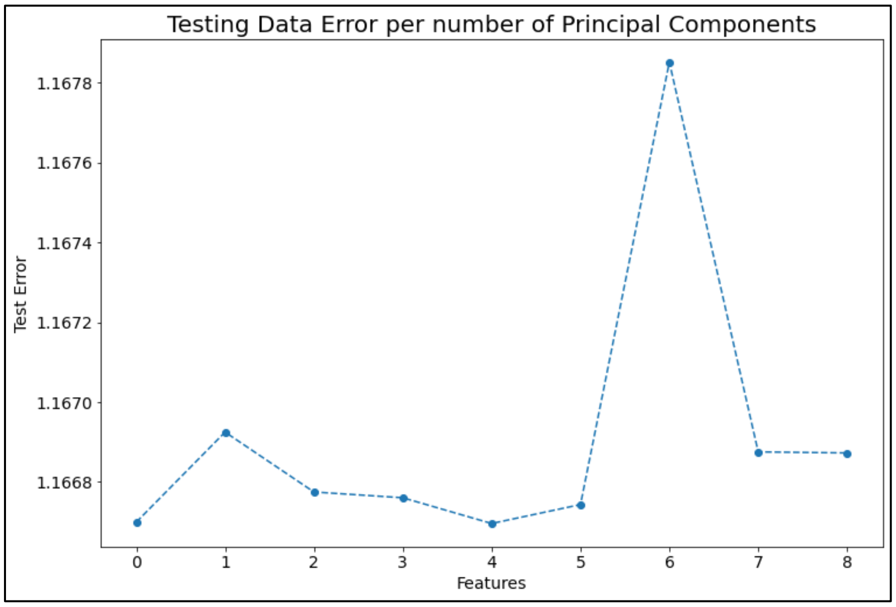
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

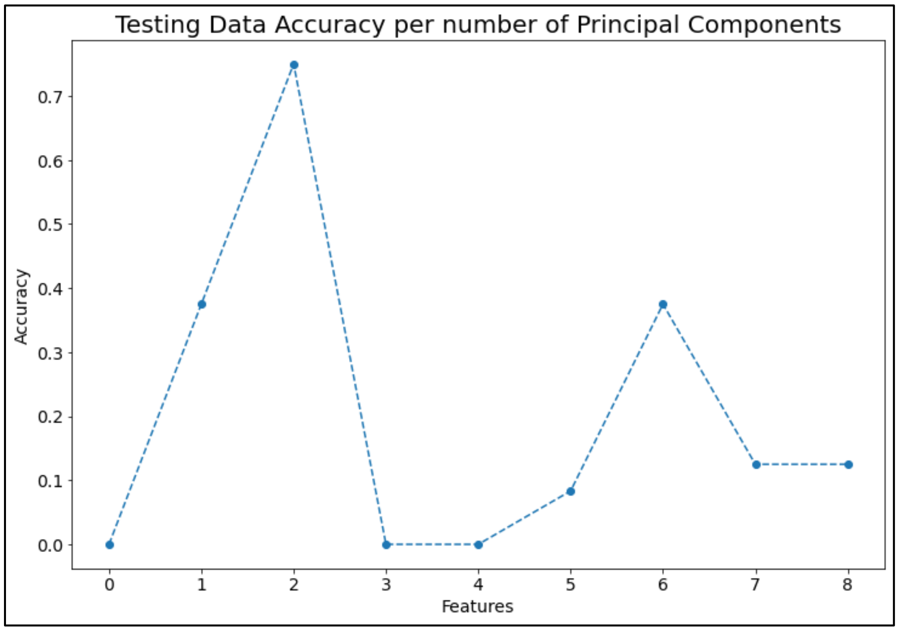
<u>EE-425 Project Report</u>

<u>For Neural Network:</u>

Best value of r for Neural Network with PCA = 5
Minimum value of error for Neural Network with PCA = 1.167
Accuracy of Neural Network with PCA = 0.9

Below are the plots for Normalized-Test-MSE and Accuracy of Neural Network on simulated after PCA:

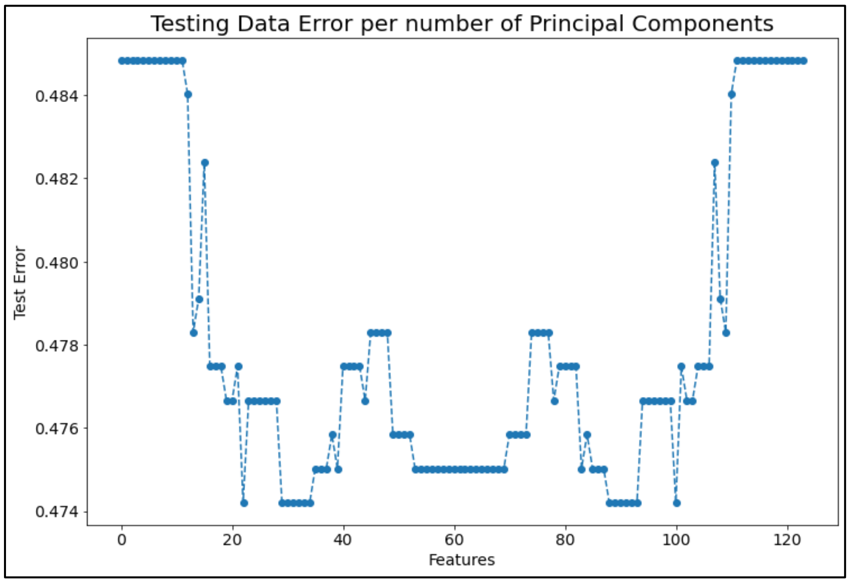


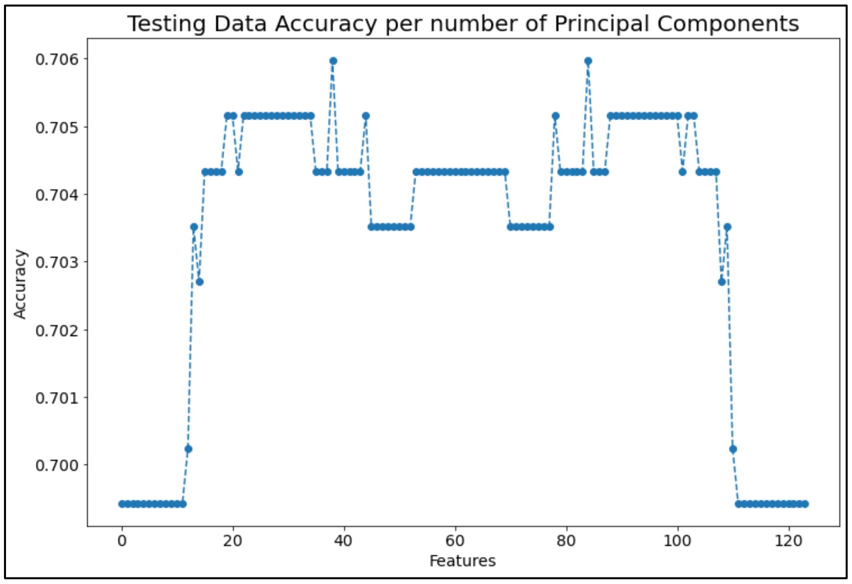


When PCA is performed on the **ICS Cyber Attack (real) dataset**, we get the following results for the testing data:

For Logistic Regression:
Best value of r for Logistic Regression with PCA = 23
Minimum value of error for Logistic Regression with PCA = 0.474
Accuracy of Logistic Regression with PCA = 0.706

Below are the plots for Normalized-Test-MSE and Accuracy of Logistic Regression on real data after PCA:





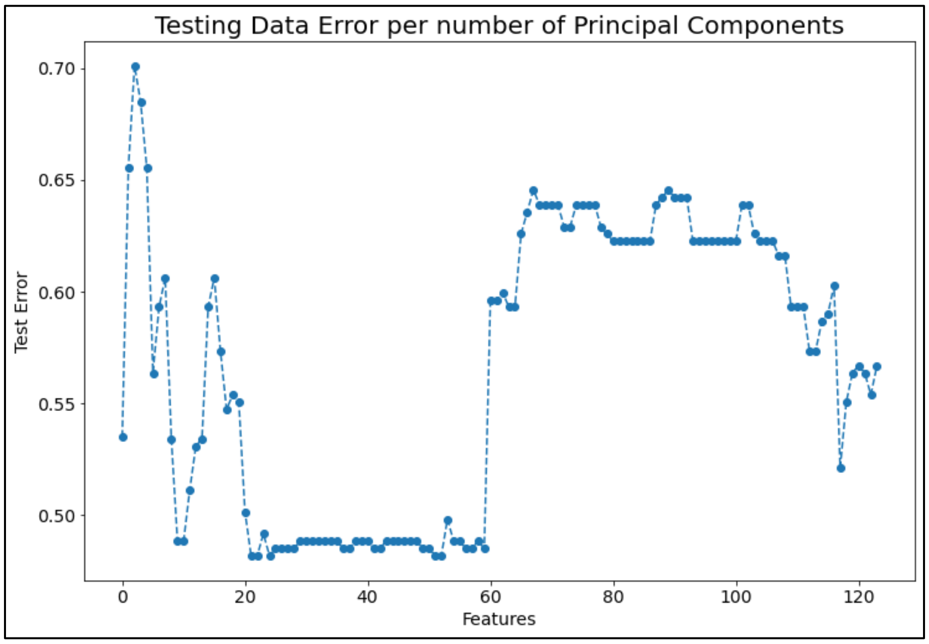
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

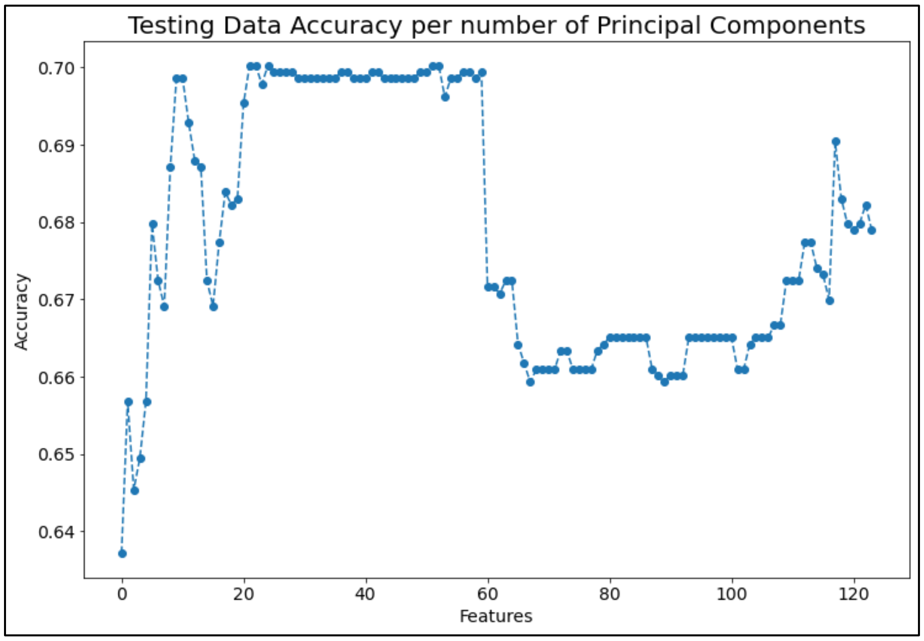
EE-425 Project Report

For Naïve Bayes:

Best value of r for Naive Bayes with PCA = 22
Minimum value of error for Naive Bayes with PCA = 0.481
Accuracy of Naive Bayes with PCA = 0.700

Below are the plots for Normalized-Test-MSE and Accuracy of Naïve Bayes on real data after PCA:



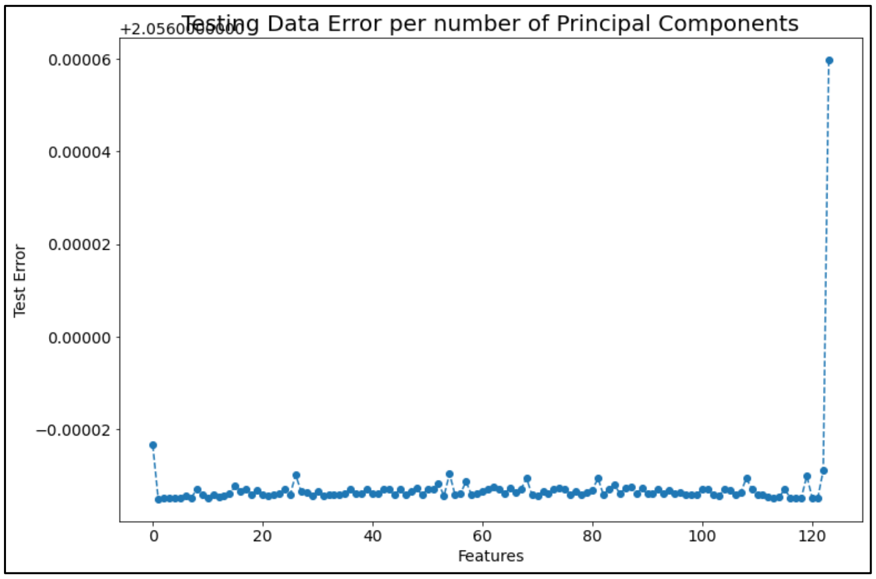


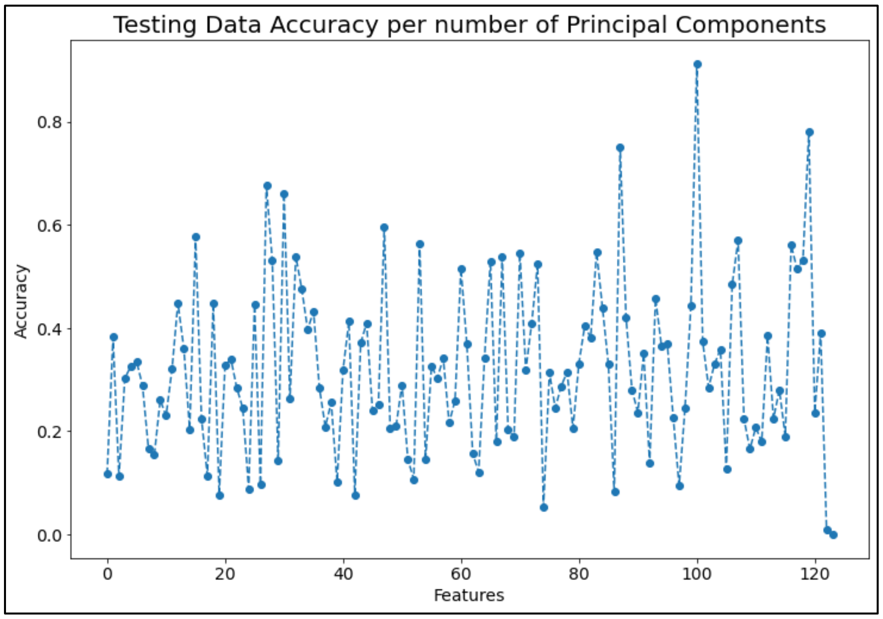
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

<u>EE-425 Project Report</u>

<u>For Neural Network:</u>
Best value of r for Neural Network with PCA = 100
Minimum value of error for Neural Network with PCA = 2.056
Accuracy of Neural Network with PCA = 0.913

Below are the plots for Normalized-Test-MSE and Accuracy of Neural Network on real after PCA:



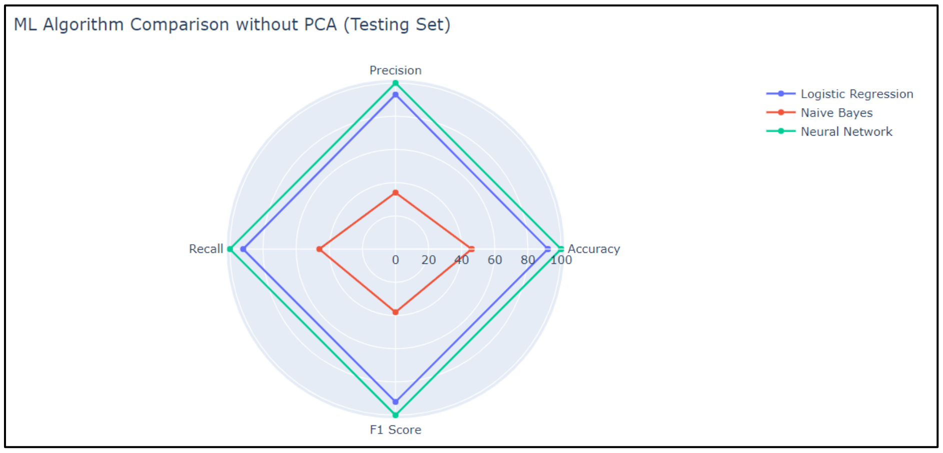


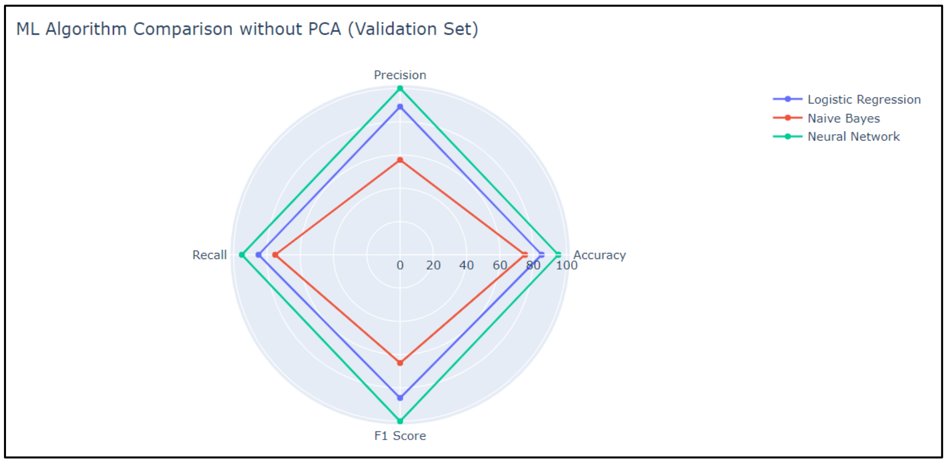
8. Inferences from Analysis & Experiments

- The tables and plots provided in Section-7 for the three algorithms - Multinomial Logistic Regression, Naïve Bayes Classifier, and Neural Network, shows the performance examination results in terms of the weighted average of our evaluation metrics for the three selected common machine learning algorithms derived from the generated dataset and the ICS Cyber Attack real world dataset.
- We divided the simulated and real datasets into three parts – training data, testing data, and validation data.
- Based on our experiments on the simulated datasets, Logistic Regression and Naïve Bayes where the fastest while training the model and classifying the testing set. Neural network took a little longer as we have defined the model with 60 epochs and 10 batches per epoch.
- Initially when we compared the three algorithms without performing PCA, according to the weighted average of the evaluation metrics (accuracy, precision, recall and F1 score), the highest accuracy belongs to Neural Network, followed by Logistic Regression and lastly Naïve Bayes.

  We obtained the following results:
  - Neural Network – Accuracy 100% for testing data, and 95% for validation set.
  - Logistic Regression – Accuracy 92% for testing data, and 85% for validation set.
  - Naïve Bayes – Accuracy 45% for testing data, and 75% for validation set.

  Refer below charts for detailed analysis:



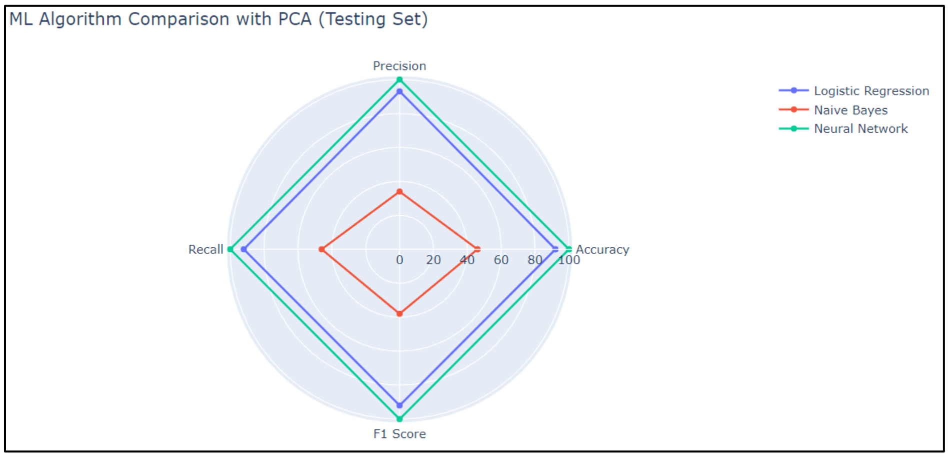


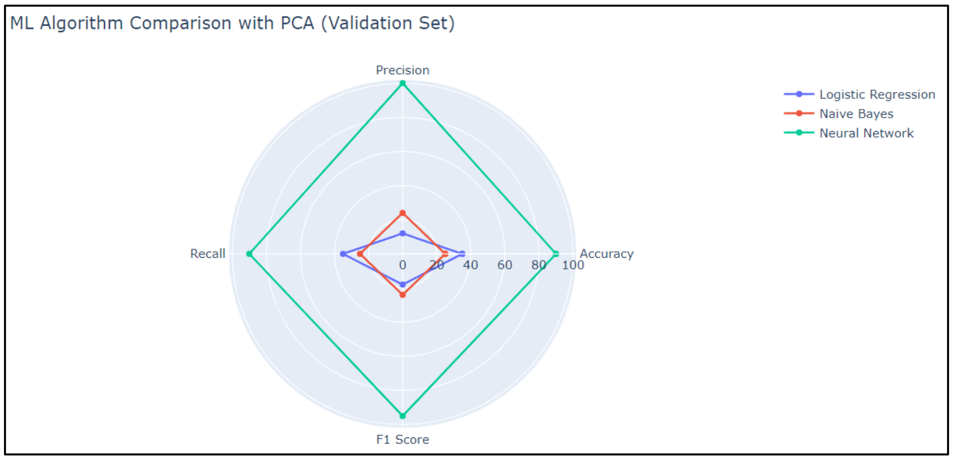
Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

EE-425 Project Report

- After performing PCA, the accuracy of the three algorithms decreased as expected. This is due to the fact that we are reducing the dimensions of the training data by performing PCA, hence some of the features are discarded.
  However, Neural Network was still able to maintain high accuracy comparatively.
  We obtained the following results:
  - Neural Network – Accuracy 91% for testing data, and 90% for validation set.
  - Logistic Regression – Accuracy 50% for testing data, and 35% for validation set.
  - Naïve Bayes – Accuracy 45% for testing data, and 25% for validation set.

  Refer below charts for detailed analysis:



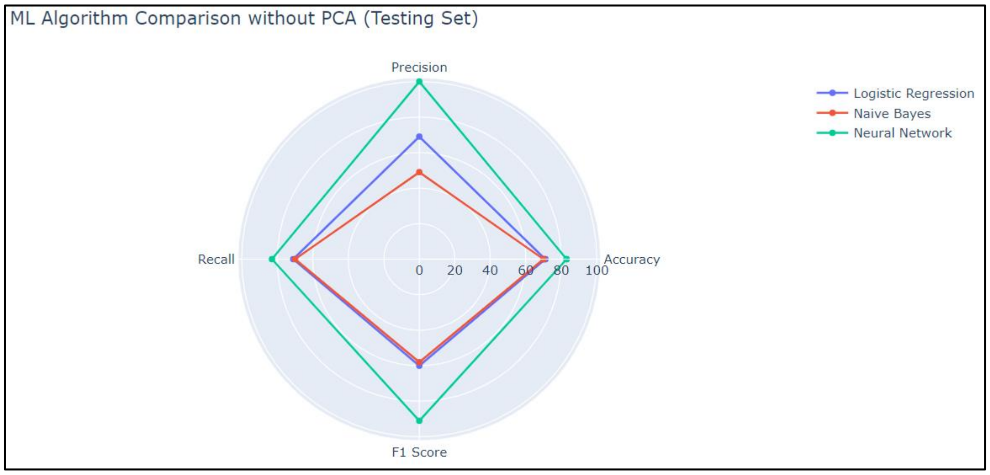


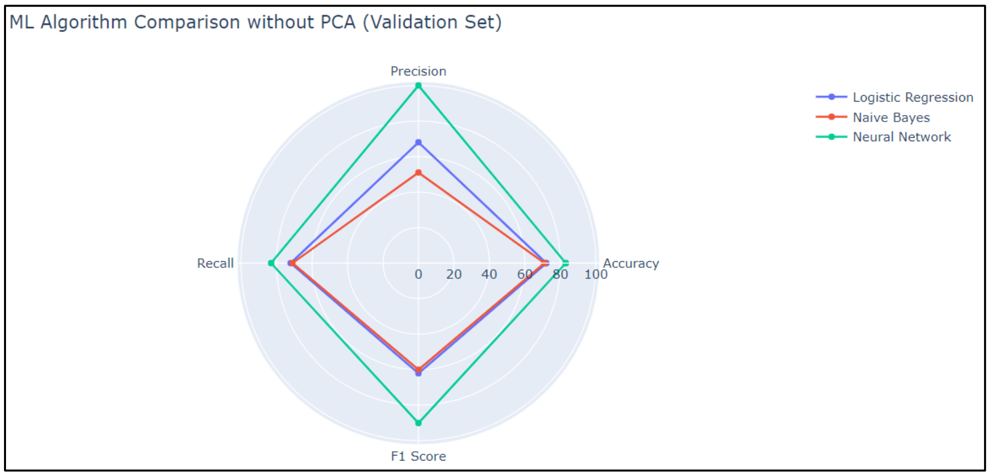
- While performing the above experiments on the ICS Cyber Attack Dataset, the Neural Network algorithm took around 30 minutes to execute. However, the Logistic Regression and Naïve Bayes were much faster, with each taking about 5 to 10 minutes to run.
- When we compared these algorithms on the real data without PCA, Neural Network still had the best performance metrics. The results of Logistic Regression and Naïve Bayes had also improved compared to the analysis with simulated data.
  We obtained the following results:
  - Neural Network – Accuracy 83% for testing data, and 82% for validation set.
  - Logistic Regression – Accuracy 70% for testing data, and 71% for validation set.
  - Naïve Bayes – Accuracy 69% for testing data, and 71% for validation set.

  Refer below charts for detailed analysis:

Chaonan Yang
Koffi Adohinzin
Souradeep Bhattacharya

<u>EE-425 Project Report</u>

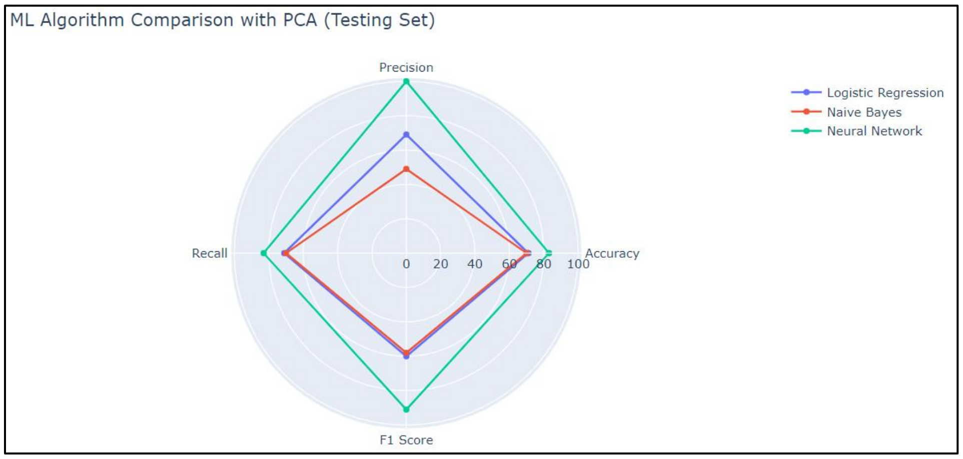


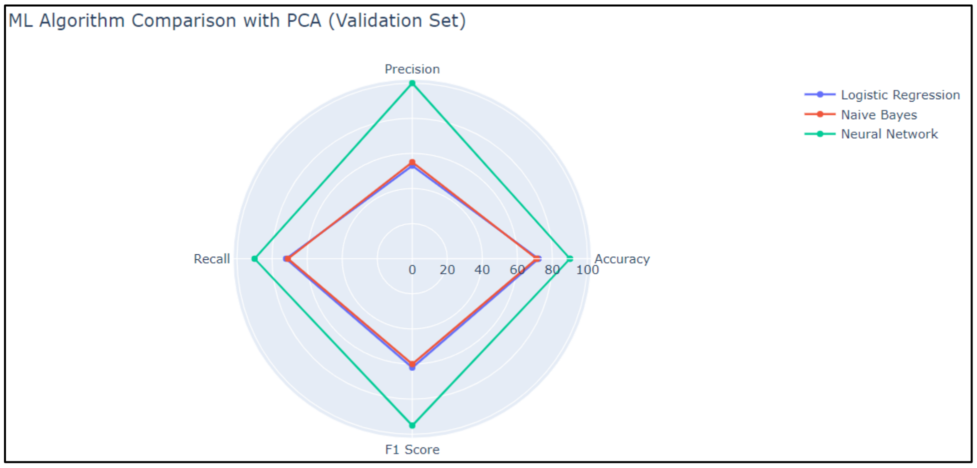


- When we compared these algorithms on the real data with PCA, the performance of all three algorithms were similar to the experiment without PCA. However, the Neural Network performance improved considerably.

  We obtained the following results:
  - Neural Network – Accuracy 91% for testing data, and 90% for validation set.
  - Logistic Regression – Accuracy 70% for testing data, and 72% for validation set.
  - Naïve Bayes – Accuracy 70% for testing data, and 71% for validation set.

  Refer below charts for detailed analysis:





9. Conclusion

The main contribution of this project was to implement three common Machine Learning Algorithms – such as Multinomial Logistic Regression, Naïve Bayes Classifier, and Neural Network, and to evaluate their performance while attempting to solve a multi-class classification problem. Considering the performance evaluation metrics, such as accuracy, precision, recall, and F1 score, we can conclude that Neural Network is the best suited algorithm for solving multi-class classification problems. Although the execution time for the Neural Network algorithm was higher compared to the other two algorithms, it produced the best results when implemented on both simulated data and real data.

Through this project, we were able to gain a much deeper understanding of machine learning fundamentals, related to the mathematics behind it and also the algorithm implementation in python. We were able to analyze three different algorithms and evaluate their pros and cons. We were also able to enhance our coding skills in python and familiarized ourselves with the vast library of machine learning classes and functions available.

10. References

[1] Y. Xu, “A review of cyber security risks of power systems: from static to dynamic false data attacks,” *Protection and Control of Modern Power Systems*, vol. 5, no. 1, Dec. 2020, doi: 10.1186/s41601-020-00164-w.

[2] Anna University and IEEE Aerospace and Electronic Systems Society, *2019 International Carnahan Conference on Security Technology (ICCST) : ICCST 2019 : IEEE 53rd International Carnahan Conference on Security Technology : October 01-03, 2019, Anna University, Chennai, India.*

[3] S. Pan, T. Morris, and U. Adhikari, “Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems,” *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 3104–3113, Nov. 2015, doi: 10.1109/TSG.2015.2409775.

[4] “ICS Cyber Attack Dataset, https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets.”