CS389 INNOVATION LAB

Network Intrusion Detection System

Mukund Sharma
2101AI19

Mentored By -
Dr. Mayank Agrawal

PROBLEM STATEMENT

Network Intrusion Detection based on various machine learning & deep learning algorithms using UNSW-NB15 Dataset and applying feature selection techniques.

Performing Binary and Multi-class Classification & testing scoring metrics

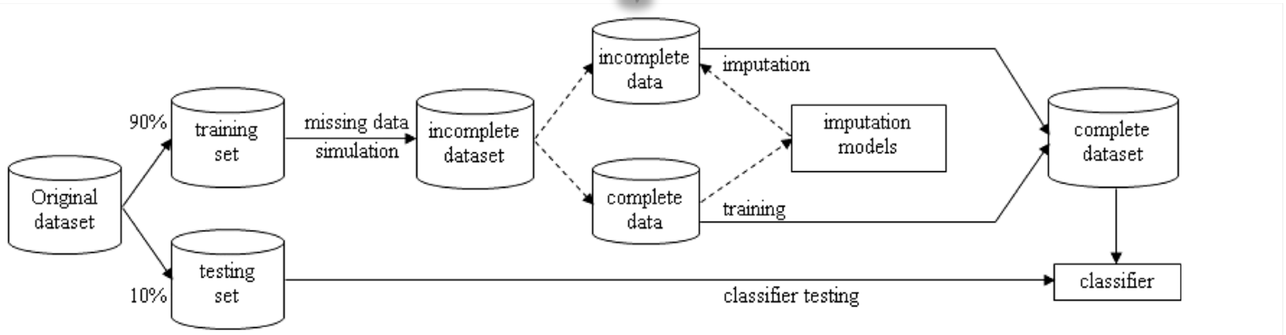
MOTIVATION

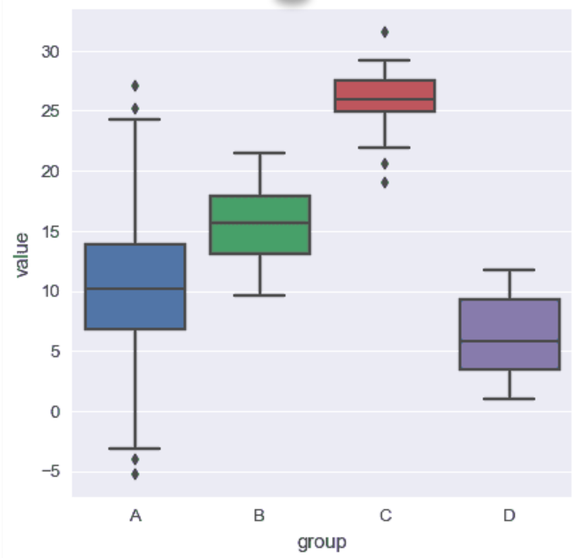
- Interest in Machine Learning Domain
- What Innovation will I contribute to ? (Target)
  - Better preprocessing of the dataset than already available methods. Performing best feature selection to favour model training. Trying large number of algorithms, tune their parameters & hyper parameters, and test their performance based on many statistical methods in order to find the best algorithm to perform classification of a new example.
  - Insufficient and imprecise details about network intruder activities, hinder security forces' ability to make crucial decisions. This lack of accurate information poses a significant risk, as minor errors in their responses could result in tragic consequences.

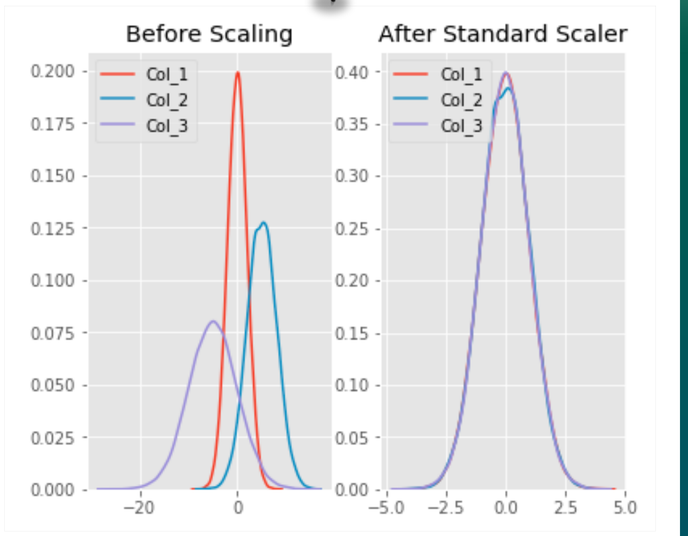
PROGRESS TILL NOW

- Preprocessing Techniques
  - NA & Duplicates Analysis, Encoding, Scaling, Train Test Split
- Exploratory Data Analysis (EDA)
  - Outlier Detection, Balancing Dataset, Univariate & BiVariate Analysis
- Feature Selection (More in next slides)
- Classification Algorithms (More in next slides)
- Hands on Experience of Libraries that will be used
- Hyper parameter tuning techniques
  - GridSearchCV Usage to find optimal hyperparam's
- Ablation Study and Analysis

Pre-Processing of the Dataset







State (Nominal Scale)	State (Label Encoding)
Maharashtra	3
Tamil Nadu	4
Delhi	0
Karnataka	2
Gujarat	1
Uttar Pradesh	5

Feature Selection - 1

Exhaustive Feature Selection

```python
clf = DecisionTreeClassifier()
efs = ExhaustiveFeatureSelector(clf,min_features = 5,max_features = 10,n_jobs = -1)
efs.fit(x_train_bin,y_train_bin)
efs.best_feature_names_

col_drop = [col for col in efs.best_feature_names_
            if col not in complete_bin_df.columns]
# complete_bin_df = complete_bin_df.drop(columns = col_drop)

# complete_bin_df.shape
col_drop,efs.best_feature_names_
```

```python
col_drop_multi = []
for i in range(corr_matrix_multi.shape[0]):
  if abs(corr_matrix_multi.iloc[-1,i]) <= 0.125:
    col_drop_multi.append(complete_multi_df.columns[i])
x_train_multi = x_train_multi.drop(columns = col_drop_multi)
x_test_multi = x_test_multi.drop(columns = col_drop_multi)

col_drop_bin = []
for i in range(corr_matrix_bin.shape[0]):
  if abs(corr_matrix_bin.iloc[-1,i]) <= 0.3:
    col_drop_bin.append(complete_bin_df.columns[i])
x_train_bin = x_train_bin.drop(columns = col_drop_bin)
x_test_bin = x_test_bin.drop(columns = col_drop_bin)
```

```python
x_train_bin.shape,x_test_bin.shape,x_train_multi.shape,x_test_multi.shape
```

Mutual Info

```python
top_10_bin = SelectKBest(mutual_info_classif,k=10)
top_10_bin.fit(x_train_bin,y_train_bin)

# top_10_bin.get_support()

#***HARD CODE ALERT***

L = np.array([ True, False, False, False, False, False,  True,  True,  True,
               True,  True,  True, False, False, False, False, False, False,
              False, False, False, False, False, False, False, False,  True,
               True, False, False, False,  True, False, False, False, False,
              False, False, False, False, False, False])

# label_bool = np.array([True])
# L = np.concatenate((L, label_bool))

col_drop = [col for col in x_train_bin.columns
            if col not in x_train_bin.columns[L]]

x_train_bin = x_train_bin.drop(columns = col_drop)
x_test_bin = x_test_bin.drop(columns = col_drop)

x_train_bin.shape,x_test_bin.shape
```

```
((140272, 10), (35069, 10))
```

ROC Feature Selection

```python
roc_values = []
i = 1
for feature in x_train_bin.columns:
  clf = DecisionTreeClassifier()
  clf.fit(x_train_bin[feature].to_frame(),y_train_bin)
  y_pred_bin = clf.predict_proba(x_test_bin[feature].to_frame())
  roc_values.append(roc_auc_score(y_test_bin,y_pred_bin[:,1]))

roc_values = pd.Series(roc_values)
roc_values.index = x_train_bin.columns
roc_values_sorted = roc_values.sort_values(ascending = False)

i = 1
top10_feature = []
for feature in roc_values_sorted.index:
  if i>10:
    break
  else:
    top10_feature.append(feature)
    i = i + 1

x_train_bin = x_train_bin.loc[:,top10_feature]
x_test_bin = x_test_bin.loc[:,top10_feature]
x_train_bin.head()
```

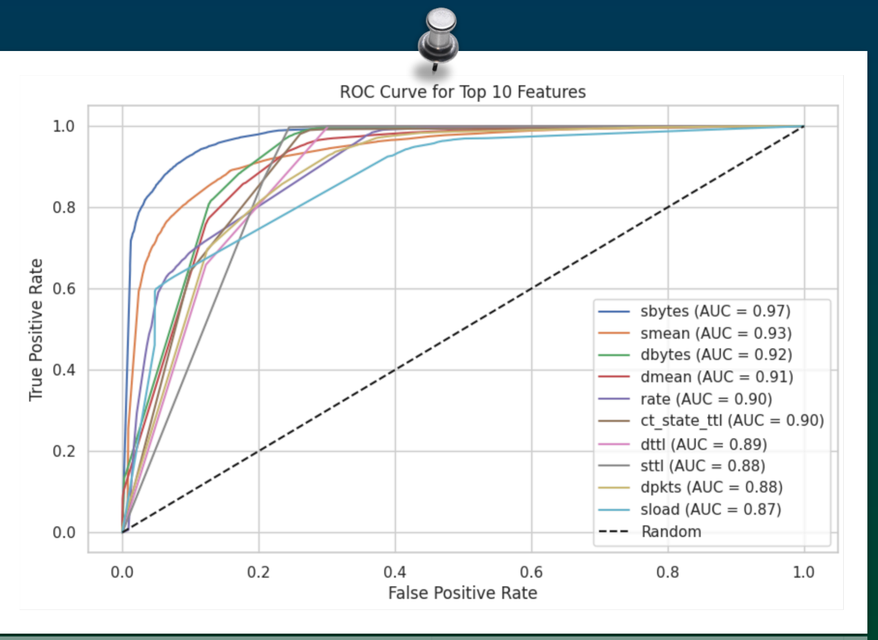
	sbytes	smean	dbytes	dmean	rate	ct_state_ttl	dttl	sttl	dpkts	sload
96203	-0.046067	-0.282161	-0.095862	0.080624	-0.576621	-0.318711	1.560002	-1.141901	-0.099490	-0.389914
58960	-0.049465	-0.179560	-0.103923	-0.480703	0.094951	0.729064	-0.720406	0.723268	-0.172047	0.081945
65069	-0.015878	0.978363	-0.099203	-0.260043	-0.576628	-0.318711	1.560002	0.723268	-0.063212	-0.389708
64133	-0.046250	-0.296818	-0.074547	1.152953	-0.576705	-0.318711	1.560002	-1.141901	-0.081351	-0.389944
111445	-0.049465	-0.179560	-0.103923	-0.480703	0.632367	0.729064	-0.720406	0.723268	-0.172047	0.459479

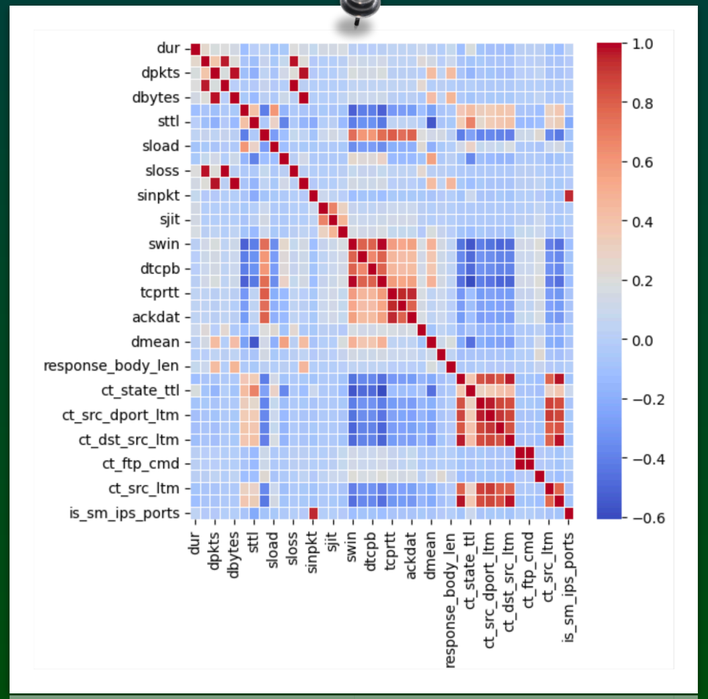
Variance Threshold (Quasi Constant)

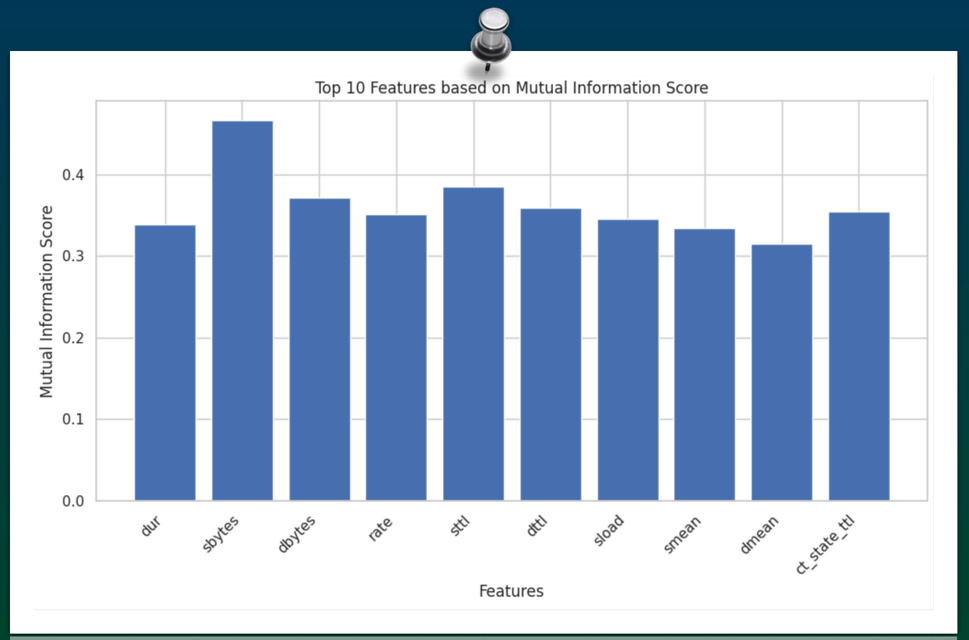
```python
VT = VarianceThreshold(threshold = 1)
VT.fit(x_train_bin)
VT.get_support()
col_drop = []
for i in range(len(x_train_bin.columns)):
  if VT.get_support()[i] == False:
    col_drop.append(x_train_bin.columns[i])
x_train_bin = x_train_bin.drop(columns = col_drop)
x_test_bin = x_test_bin.drop(columns = col_drop)

VT = VarianceThreshold(threshold = 1)
VT.fit(x_train_multi)
VT.get_support()
col_drop = []
for i in range(len(x_train_multi.columns)):
  if VT.get_support()[i] == False:
    col_drop.append(x_train_multi.columns[i])
x_train_multi = x_train_multi.drop(columns = col_drop)
x_test_multi = x_test_multi.drop(columns = col_drop)
```

```python
x_train_bin.shape,x_test_bin.shape,x_train_multi.shape,x_test_multi.shape
```

Feature Selection - 2







Classification Algo - 1

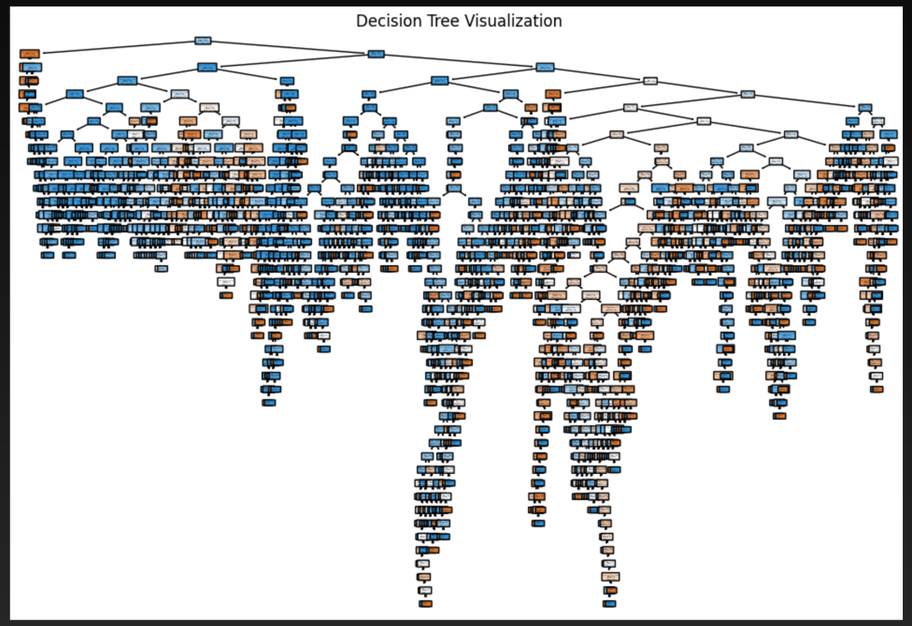
Some algorithms used to perform classification -

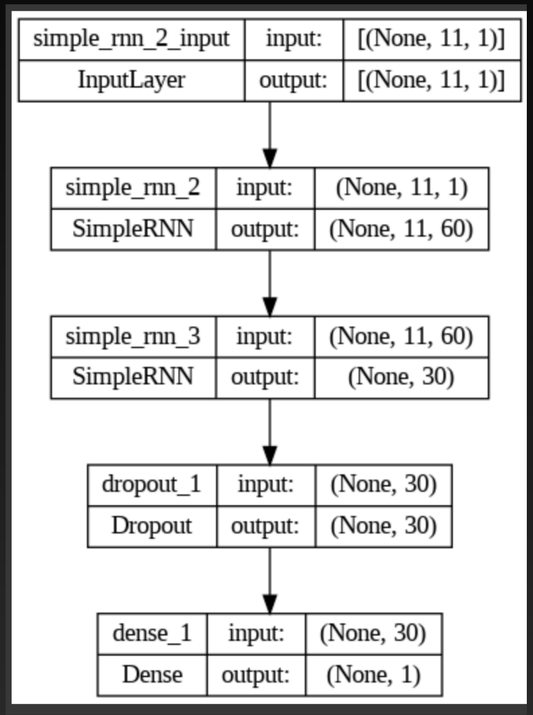
- KNN, Decision Trees, SVC, Naive Bayes (Gaussian and Multinomial), Ridge Classifier, and more
- Random Forest, AdaBoost, Gradient Boosting, and more
- CNN, SimpleRNN, MLP

HyperParams Implemented -

- Feature Sampling = {0.5,1}
- Examples Sampling = {0.5,1}
- Weights in KNN = {Uniform,Distance}
- Internal Base Algorithms = {KD Tree, Ball Tree, Brute}

Classification Algo - 2



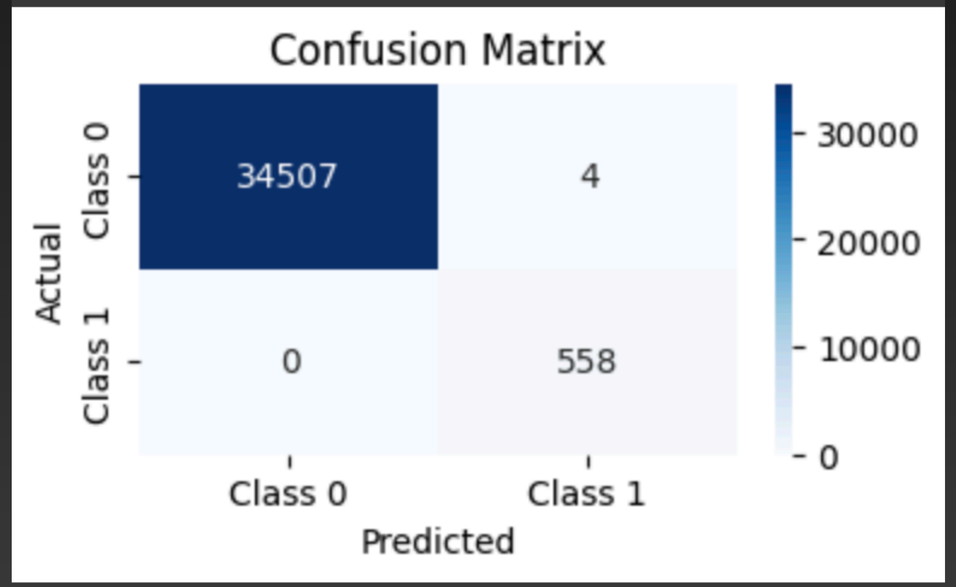


Results

		80:20 Split		
Classifier Used	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1. Adaboost	0.9327611281	0.9264729173	0.9790376569	0.952030271
2. Gaussian Naive Bayes	0.83857538	0.8302371899	0.9592887029	0.8901096768
3. KNN Classifier	0.918817189	0.9256398852	0.9578242678	0.9414570953
4. Decision Tree	0.937979412	0.9508570954	0.9585355649	0.954680891
5. Multiclass One Vs Rest (DecisionTree)	0.6093415837	0.586669038	0.6093415837	0.5676033742
6. Multiclass One vs One (DecisionTree)	0.6094271294	0.5862682383	0.6094271294	0.5676393601
7. Ridge Classifier	0.8737346374	0.9014183228	0.9147698745	0.9080450222
8. Support vector Machine	0.8311329094	0.8141688802	0.9746861925	0.887225777
9. Multinomial Naive Bayes	0.7253699849	0.9140983226	0.6589539749	0.7658343261
10. Passive Aggressive	0.7488950355	0.8596549752	0.7547698745	0.8038053649
11. Gradient Boosting	0.9365536514	0.9274304871	0.9838912134	0.9548269211
12. Random Forest	0.9433687872	0.9516488046	0.9659832636	0.9587624585
13. Extra Tree	0.9427129374	0.9519385606	0.9646443515	0.9582493402
14. Multilayer Perceptron	0.8791240127	0.9042312595	0.920083682	0.9120885958
15. CNN	0.9317630956	0.9162376621	0.99041841	0.9518849904
16. RNN	0.9354130429	0.9369471261	0.9705439331	0.9534496578

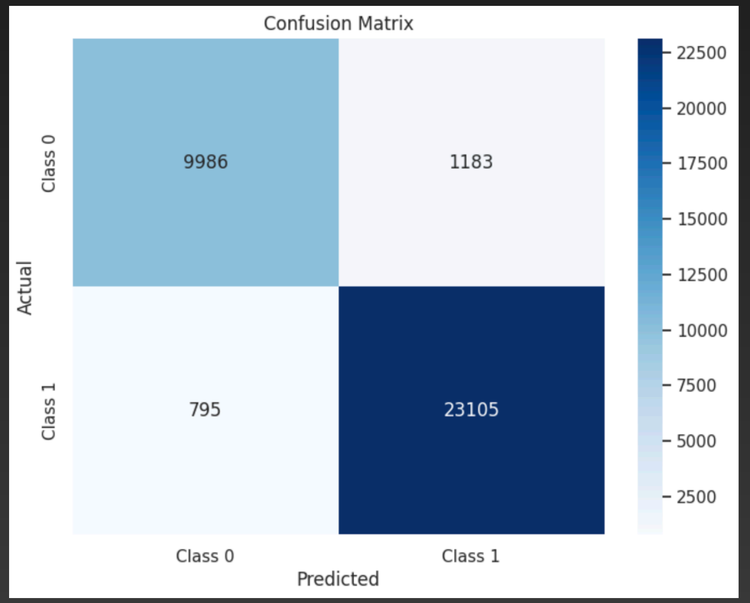
		70:30 Split		
Classifier Used	**Accuracy**	**Precision**	**Recall**	**F1 Score**
1. Adaboost	0.9335399122	0.9280173669	0.978314867	0.9525025814
2. Gaussian Naive Bayes	0.8393247533	0.8312162783	0.95880662	0.8904642181
3. KNN Classifier	0.919035036	0.9268332252	0.9566576428	0.9415093044
4. Decision Tree	0.9386726993	0.9520811958	0.9581926265	0.9551271351
5. Multiclass One Vs Rest (DecisionTree)	0.6086725092	0.5855919491	0.6086725092	0.5670253656
6. Multiclass One vs One (DecisionTree)	0.6085964679	0.5857343164	0.6085964679	0.5670405623
7. Ridge Classifier	0.8742086953	0.9022415597	0.9144037286	0.9082819322
8. Support vector Machine	0.8321198411	0.814758685	0.9752728085	0.8878190064
9. Multinomial Naive Bayes	0.7234948577	0.9132241031	0.6564427451	0.7638300291
10. Passive Aggressive	0.6874702964	0.6935113666	0.9697468672	0.8086903903
11. Gradient Boosting	0.9367336464	0.9268510493	0.9848455248	0.9549686079
12. Random Forest	0.9436153832	0.9512315677	0.9667885351	0.9589469605
13. Extra Tree	0.9425698154	0.9518039108	0.9645279227	0.9581236745
14. Multilayer Perceptron	0.8549132179	0.9241854448	0.8573302448	0.8895034023
15. CNN	0.8724086953	0.9044215597	0.9344037286	0.9602819322
16. RNN	0.9331026748	0.9311716039	0.9737657336	0.9519924694

Pearson Correlation Corr > 0.3



Ablation Study for Backdoor Attacks

Binary Classification on Complete Dataset



Future Work That Can Be Done

- Statistical Analysis Techniques
- Getting Better Domain Knowledge
  - To reduce the error beyond the point an ML model can
- Work on fine tuning the model
- Model Deployment

THANK YOU !