

## Performance Analysis

All the three models were executed on the same dataset after same pre-processing. Our target column is the category column which consists of 4 distinct values, DoS, DDoS, Normal and Reconnaissance.

### Accuracy Score

#### 1. Base model

XGBoost	0.6174
Logistic Regression	0.9798
SVM	0.8990

#### 2. K-Fold CrossValidation

XGBoost	0.6889
Logistic Regression	0.9867
SVM	0.9280

#### 3. Hyper Parameter Tuning

XGBoost	0.9048
Logistic Regression	0.9883
SVM	0.9973

### Computational Time

#### 1. Base Model

XGBoost	32.2 seconds
Logistic Regression	24.95 seconds
SVM	3138.75 seconds

## 2. Hyper Parameter Tuning

XGBoost	350 min (5 sets of tuning model)
Logistic Regression	33.0 mins
SVM	36.9min

### Classification Report

#### 1. XG Boost

	precision	recall	f1-score	support
0	1.00	0.98	0.99	385309
1	0.99	0.81	0.89	330112
2	1.00	1.00	1.00	107
3	0.21	1.00	0.35	18163
accuracy			0.90	733691
macro avg	0.80	0.95	0.81	733691
weighted avg	0.98	0.90	0.93	733691

*Figure-1: Classification report for XGBoost*

#### 2. Logistic Regression

	precision	recall	f1-score	support
0	0.98	0.99	0.98	385309
1	0.99	0.97	0.98	330112
2	1.00	1.00	1.00	107
3	0.83	0.96	0.89	18163
4	0.00	0.00	0.00	14
accuracy			0.98	733705
macro avg	0.76	0.78	0.77	733705
weighted avg	0.98	0.98	0.98	733705

*Figure-2: Classification report for Logistic Regression*

### 3. SVM

	precision	recall	f1-score	support
0	0.89	0.93	0.91	385309
1	1.00	0.86	0.92	330112
2	1.00	0.99	1.00	107
3	0.38	0.93	0.54	18163
accuracy			0.90	733691
macro avg	0.82	0.93	0.84	733691
weighted avg	0.92	0.90	0.91	733691

Figure-3: Classification report for SVM

Model	Precision	Recall	F1 Score
XG Boost	0.98	0.90	0.93
Logistic Regression	0.98	0.98	0.98
SVM	0.92	0.90	0.91

Table-1: Classification report comparison

### Confusion Matrix

#### 1. XG Boost

[	[	378939	1481	0	4889]
	[	181	266694	0	63237]
	[	0	0	107	0]
	[	5	0	0	18158]]]

Figure-4: Confusion matrix for XGBoost

#### 2. Logistic Regression

```
[ [ 379950,    2001,      0,    3358,      0],
  [   8428, 321510,      0,    174,      0],
  [      0,      0,    107,      0,      0],
  [   384,   426,      0, 17353,      0],
  [      0,    13,      0,      1,      0]] )
```

Figure-5: Confusion matrix for Logistic Regression

### 3. SVM

```
[[359050      157         0 26102]
 [ 44781 283591         0   1740]
 [         0         0    106         1]
 [   339    933         0 16891]]
```

*Figure-6: Confusion matrix for Logistic Regression*

## Inferences

Out of the three models used i.e. **Logistic regression, XG Boost and SVM**, SVM performed best after hyper parameter tuning. **Accuracy** of the three models are

Logistic Regression: ~ 97%

SVM: ~ 99%

XGBoost: ~ 91%

For the **Time Analysis** we found that SVM took the most time. It took more time because if it takes the wrong parameters then it has to compute again. After hyperparameter tuning, when we input the exact parameters, time taken to execute the SVM model decreases sharply.

Logistic Regression: ~ 166.57 seconds

SVM: ~ 3138.75 seconds

XGBoost: ~ 32.2seconds

Our target column was **category & subcategory** column. So, I have made a new column in our dataset that contains concatenated values of category and subcategory. All the three models are trained and tested on the same pre-processed dataset.

I have **trained and validated our model** on **the training dataset** and then applied that model on the **testing dataset** to get the results.

The following results are thus for both training models as well as for testing models.

### Accuracy Score

Base Training model

XGBoost	0.9998
Logistic Regression	0.8727
SVM	0.9085

Table-2: Training Model accuracy comparison

Base Testing model

XGBoost	0.7929
Logistic Regression	0.9668
SVM	0.9647

Table-3: Training Model accuracy comparison

### K-Fold Cross Validation

XGBoost	0.9998
Logistic Regression	0.8797
SVM	0.9083

Table-4: K-Fold Cross Validation scores

Hyper Parameter Tuning  
(Training Model)

XGBoost	0.9998
Logistic Regression	0.9016
SVM	0.9915

Table-5: Hyper Parameter Tuning Scores  
(Training Model)

Hyper Parameter Tuning  
(Testing Model)

XGBoost	0.9224
Logistic Regression	0.9699
SVM	0.9964

Table-6: Hyper Parameter Tuning scores  
(Testing Model)

## Computational Time

### Base Model

XGBoost	78.64 seconds
Logistic Regression	190.63 seconds
SVM	2171.62 seconds

*Table-7: Computation Time comparison for training base model*

### Hyper Parameter Tuning

XGBoost	286 min (4 sets of tuning models)
Logistic Regression	53.2 mins
SVM	217.7 mins (2 sets of tuning models)

*Table-8: Time comparison for Hyper Parameter Tuning*

## Classification Report

### Training model

- **XG Boost**

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3000
1	1.00	1.00	1.00	14000
2	1.00	1.00	1.00	14000
3	1.00	1.00	1.00	4000
4	1.00	1.00	1.00	13000
5	1.00	1.00	1.00	14000
6	1.00	1.00	1.00	1600
7	1.00	1.00	1.00	6000
8	1.00	1.00	1.00	11719
accuracy			1.00	81319
macro avg	1.00	1.00	1.00	81319
weighted avg	1.00	1.00	1.00	81319

*Figure-7: Classification Report for XGBoost on Training model*

- **Logistic Regression**

	precision	recall	f1-score	support
0	0.60	0.59	0.59	3000
1	0.91	0.92	0.92	14000
2	1.00	1.00	1.00	14000
3	0.74	0.45	0.56	4000
4	0.86	0.96	0.91	13000
5	1.00	1.00	1.00	14000
6	1.00	1.00	1.00	1600
7	0.61	0.64	0.62	6000
8	0.76	0.73	0.74	11719
accuracy			0.87	81319
macro avg	0.83	0.81	0.82	81319
weighted avg	0.87	0.87	0.87	81319

*Figure-8: Classification Report for Logistic Regression on Training model*

- SVM

	precision	recall	f1-score	support
0	0.73	0.55	0.63	3000
1	0.91	0.91	0.91	14000
2	1.00	0.99	0.99	14000
3	0.76	0.86	0.81	4000
4	0.89	0.98	0.93	13000
5	0.99	1.00	0.99	14000
6	1.00	1.00	1.00	1600
7	0.95	0.53	0.68	6000
8	0.81	0.92	0.86	11719
accuracy			0.91	81319
macro avg	0.89	0.86	0.87	81319
weighted avg	0.91	0.91	0.90	81319

*Figure-9: Classification Report for SVM on Training model*

### Testing model

- XG Boost

	precision	recall	f1-score	support
0	0.64	1.00	0.78	202
1	1.00	0.97	0.99	195149
2	1.00	0.98	0.99	189948
3	0.41	1.00	0.59	300
4	0.98	0.62	0.76	123183
5	0.98	1.00	0.99	206620
6	1.00	1.00	1.00	98
7	0.07	0.59	0.13	3615
8	0.38	1.00	0.55	14530
accuracy			0.92	733645
macro avg	0.72	0.91	0.75	733645
weighted avg	0.97	0.92	0.94	733645

*Figure-10: Classification Report for XGBoost on Testing model*



## Confusion Matrix

### Training Model

	DDoS_HTTP	DDoS_TCP	DDoS_UDP	DoS_HTTP	DoS_TCP	DoS_UDP	Normal_Normal	OS_Fingerprint	Service_Scan
DDoS_HTTP	3000	0	0	0	0	0	0	0	0
DDoS_TCP	1	13993	0	0	6	0	0	0	0
DDoS_UDP	0	0	14000	0	0	0	0	0	0
DoS_HTTP	0	0	0	4000	0	0	0	0	0
DoS_TCP	0	0	0	0	13000	0	0	0	0
DoS_UDP	0	0	0	0	0	14000	0	0	0
Normal_Normal	0	0	0	0	0	0	1600	0	0
OS_Fingerprint	0	0	0	0	0	0	0	5999	1
Service_Scan	0	0	0	0	0	0	0	5	11714

*Figure-11: Data Frame of Training model*

### XG Boost: -

```
array([[78318, 1],
       [ 0, 3000]],

       [[67319, 0],
       [ 7, 13993]],

       [[67319, 0],
       [ 0, 14000]],

       [[77319, 0],
       [ 0, 4000]],

       [[68313, 6],
       [ 0, 13000]],

       [[67319, 0],
       [ 0, 14000]],

       [[79719, 0],
       [ 0, 1600]],

       [[75314, 5],
       [ 1, 5999]],

       [[69599, 1],
       [ 5, 11714]]], dtype=int64)
```

*Figure-12: Confusion Matrix for XGBoost on Training model*

### Logistic Regression

```
array([[77165, 1154],
       [ 1244, 1756]],

      [[66081, 1238],
       [ 1120, 12880]],

      [[67307, 12],
       [    0, 14000]],

      [[76675, 644],
       [ 2199, 1801]],

      [[66233, 2086],
       [  463, 12537]],

      [[67315, 4],
       [    5, 13995]],

      [[79719, 0],
       [    0, 1600]],

      [[72875, 2444],
       [ 2172, 3828]],

      [[66832, 2768],
       [ 3147, 8572]]])
```

*Figure-13: Confusion Matrix for Logistic Regression  
on Training model*

### SVM

```
array([[77700, 619],
       [ 1343, 1657]],

      [[66119, 1200],
       [ 1320, 12680]],

      [[67315, 4],
       [  152, 13848]],

      [[76246, 1073],
       [  550, 3450]],

      [[66695, 1624],
       [  322, 12678]],

      [[67167, 152],
       [    4, 13996]],

      [[79719, 0],
       [    0, 1600]],

      [[75141, 178],
       [ 2816, 3184]],

      [[67017, 2583],
       [  926, 10793]]])
```

*Figure-14: Confusion Matrix for SVM  
on Training model*

## Testing model

	DDoS_HTTP	DDoS_TCP	DDoS_UDP	DoS_HTTP	DoS_TCP	DoS_UDP	Normal_Normal	OS_Fingerprint	Service_Scan
DDoS_HTTP	202	0	0	0	0	0	0	0	0
DDoS_TCP	70	189777	0	36	1544	0	0	89	3633
DDoS_UDP	0	0	186482	0	0	3465	0	1	0
DoS_HTTP	0	0	0	299	0	0	0	0	1
DoS_TCP	46	32	0	386	76634	0	0	27759	18326
DoS_UDP	0	0	2	0	0	206616	0	2	0
Normal_Normal	0	0	0	0	0	0	98	0	0
OS_Fingerprint	0	2	0	0	2	0	0	2149	1462
Service_Scan	0	3	0	0	0	0	0	7	14520

Figure-15: Data Frame of Testing model

## XG Boost

```
array([[733327, 116],
       [ 0, 202]],

      [[538459, 37],
       [ 5372, 189777]],

      [[543695, 2],
       [ 3466, 186482]],

      [[732923, 422],
       [ 1, 299]],

      [[608916, 1546],
       [ 46549, 76634]],

      [[523560, 3465],
       [ 4, 206616]],

      [[733547, 0],
       [ 0, 98]],

      [[702172, 27858],
       [ 1466, 2149]],

      [[695693, 23422],
       [ 10, 14520]]], dtype=int64)
```

Figure-16: Confusion Matrix for XGBoost on Testing model

## Logistic Regression

```
array([[77189, 1130],
       [ 804, 2196]],

      [[66843, 476],
       [ 1017, 12983]],

      [[67315, 4],
       [ 0, 14000]],

      [[76381, 938],
       [ 1055, 2945]],

      [[66966, 1353],
       [ 450, 12550]],

      [[67319, 0],
       [ 8, 13992]],

      [[79719, 0],
       [ 0, 1600]],

      [[73786, 1533],
       [ 2610, 3390]],

      [[67016, 2584],
       [ 2074, 9645]]])
```

*Figure-17: Confusion Matrix for Logistic Regression  
on Testing model*

## SVM

```
array([[733408, 35],
       [ 0, 202]],

      [[538481, 15],
       [ 1366, 193783]],

      [[543679, 18],
       [ 4, 189944]],

      [[733334, 11],
       [ 1, 299]],

      [[609372, 1090],
       [ 30, 123153]],

      [[527024, 1],
       [ 22, 206598]],

      [[733547, 0],
       [ 98, 0]],

      [[729094, 936],
       [ 132, 3483]],

      [[718620, 495],
       [ 948, 13582]]])
```

*Figure-18: Confusion Matrix for SVM  
on Testing model*

## Inferences

Out of the three models used i.e. **Logistic regression, XG Boost and SVM**, SVM performed best after hyper parameter tuning. **Accuracy** of the three models after hyper parameter tuning are:

XGBoost: ~ 92.2%

Logistic Regression: ~ 96.9%

SVM: ~ 99.6%

For the Time Analysis it was found that SVM took the most time. It took more time because if it takes the wrong parameters then it has to compute again.

After hyperparameter tuning, when we input the exact parameters, time taken to execute the model decreases sharply.

XGBoost: ~ **78.6 seconds**

Logistic Regression: ~ **190.6 seconds**

SVM: ~ **2171.6 seconds**