Predictive Analytics For A Cyber Security Problem

Cris Lourenz Tan

Semester 2, 2023

Contents

1 Introduction

The purpose behind this report is to give an organized way to the data analytics process for Intrusion Detection, focusing on the selection and recommendation of a Machine Learning algorithm for an Intrusion Detection System (IDS). In this undertaking, I took a trial utilizing 2-3 Supervised Learning models to address a linear regression problem inside the context of Intrusion Detection Systems. The objective is to identify the model that exhibits superior performance based on established metrics, and then the goal is to select the model with superior performance. This report is intended for network safety chiefs who are wide comprehension of the main issue at hand yet may not be personally acquainted with the complexities of data analytics. Utilizing simulated IDS data to strengthen the organization's security infrastructure is my primary responsibility. To accomplish this, I will assess the performance of 2-3 Supervised Learning algorithms, treating the simulated data as a trial ground. The findings from this experimental phase will be the basis upon which I recommend the most effective algorithm for organizational deployment.

2 Problem Definition

The main challenge faced by cybersecurity stakeholders revolves around identifying and mitigating suspicious or malicious activities within their network infrastructure. This issue transcends the boundaries of individual departments or specific groups within an organization; it permeates through every level, impacting everyone from top-tier management to the frontline workforce. The potential risks associated with cybersecurity breaches can unleash a domino effect on an organization's day-to-day operations. They have the potential to disrupt crucial business processes, compromise sensitive client information, and ultimately lead to significant financial losses. Given the magnitude of these potential consequences, it becomes imperative to recognize that the responsibility for risk management should not fall solely on the shoulders of the IT department. Cybersecurity is a collective endeavor that necessitates the active participation of all employees. This spans from adhering to best practices for password management to maintaining a constant vigilance against phishing attempts. It's crucial to educate representatives about the various digital threats and empower them to understand how their actions can either mitigate or exacerbate these risks. In the event of a cybersecurity incident, the repercussions resonate far beyond the confines of the IT department. The reverberations extend throughout the entire organization. For instance, a data breach can trigger reputational damage that. directly impacts sales and client relationships. This, in turn, demands concerted efforts from marketing and customer service teams to rebuild trust and restore confidence among stakeholders.

3 Data Preparation

The IDS dataset is a collection of network traffic data captured from various sources. It comprises attributes such as source and destination IP addresses, ports, protocols, packet counts, and timestamps. Each entry in the dataset is labelled as either normal or indicative of an intrusion, providing a basis for supervised learning. Below is an example of data information about the given dataset from our professor in School.

The “df.info” method provides insights into the data types, non-null counts, and memory usage, allowing us to ensure data integrity and identify any missing values.

The “df.describe” is a method offers summary statistics (mean, min, max, etc.) for numeric variables, aiding in the understanding of data distributions.

```
df.info()
df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 182 entries, 0 to 181
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Area                   182 non-null    int64
 1   Sensing Range          182 non-null    int64
 2   Transmission Range     182 non-null    int64
 3   Number of Sensor nodes  182 non-null    int64
 4   Number of Barriers     182 non-null    int64
dtypes: int64(5)
memory usage: 7.2 KB
```

Out[3]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
count	182.000000	182.00000	182.000000	182.000000	182.000000
mean	24375.000000	27.50000	55.000000	250.000000	94.071429
std	15197.252769	7.52069	15.041379	90.248276	65.171006
min	5000.000000	15.00000	30.000000	100.000000	12.000000
25%	9375.000000	21.00000	42.000000	172.000000	42.000000
50%	21875.000000	27.50000	55.000000	250.000000	80.000000
75%	39375.000000	34.00000	68.000000	328.000000	128.750000
max	50000.000000	40.00000	80.000000	400.000000	320.000000

I then used a Data Cleaning method “df=df.dropna()” just to make sure that the data doesn’t have any NULL data. One of the most important steps in data cleaning is to remove NULL data. NULL data refers to a lack of a value or a missing value in a specific field or column. It can happen due to various reasons such as a data entry error, a missing field in the original data source, or a field that was not applicable to a certain record. NULL data can cause unreliable outcomes and algorithms even though they may look correct. And followed with the “df.head”, and “df.tail”, which would show us how the results would be from the looks at the first row, and also at the last row.

```
df=df.dropna()
```

```
df.head()
```

Out[5]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56

In [6]:

```
df.tail()
```

Out[6]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
177	50000	36	72	352	101
178	50000	37	74	364	107
179	50000	38	76	376	114
180	50000	39	78	388	121
181	50000	40	80	400	128

5 Exploratory Data Analysis and Modeling

In the exploratory data analysis (EDA) stage, the dataset was carefully analysed for any irregularities such as missing values, outliers, or anomalies. I am including histograms, box plots, and scatter plots in order to gain insights into the distribution of the target variable and features. The models' suitability was demonstrated by the strong linear relationships found between features and the target variable. Given the varying scales of features, the StandardScaler was employed to standardize them, ensuring equitable contributions to the model's learning process. This model assumes a linear relationship between features and the target. Additionally, the Random Forest Regression model was selected due to its capacity to capture complex, non-linear relationships and interactions between features. Visual proof from the EDA, for example, disperse plots representing clear straight patterns, upheld the decision of these models. The selected models were then trained and evaluated using appropriate metrics like R-squared and Mean Squared Error. For the subsequent experimental phase, this comprehensive approach to EDA and model selection provides a solid foundation. And from the picture below, it shows how the relationships between variables are. And I can somehow confirm that this is a Linear Regression Problem.

```
correlation_matrix = df.corr()
print(correlation_matrix)
```

```
                            Area  Sensing Range  Transmission Range  \
Area                1.000000e+00   3.095077e-16        3.095077e-16
Sensing Range       3.095077e-16   1.000000e+00        1.000000e+00
Transmission Range  3.095077e-16   1.000000e+00        1.000000e+00
Number of Sensor nodes -1.162999e-16   1.000000e+00        1.000000e+00
Number of Barriers     -4.234383e-01   8.383655e-01        8.383655e-01

                        Number of Sensor nodes  Number of Barriers
Area                             -1.162999e-16           -0.423438
Sensing Range                     1.000000e+00            0.838365
Transmission Range                1.000000e+00            0.838365
Number of Sensor nodes            1.000000e+00            0.838365
Number of Barriers                8.383655e-01            1.000000
```

6 Experiment

I dive into experiment conducted to compare the performance of Linear Regression and Random Forest Regression models. Preceding model training, a fastidious preprocessing step was undertaken. A crucial role was played by feature selection, where a careful curation of relevant features was executed based on their significance in predicting the target variable. Additionally, Principal Component Analysis (PCA) was explored to extract the most influential components, enhancing model interpretability.

LINEAR REGRESSION

In [33]:
```
df.head()
```
Out[33]:

	Area	Sensing Range	Transmission Range	Number of Sensor nodes	Number of Barriers
0	5000	15	30	100	30
1	5000	16	32	112	35
2	5000	17	34	124	42
3	5000	18	36	136	48
4	5000	19	38	148	56

In [43]:
```
lr = LinearRegression()
```

In [44]:
```
lr.fit(X_train, y_train)
```
Out[44]:
```
LinearRegression()
```

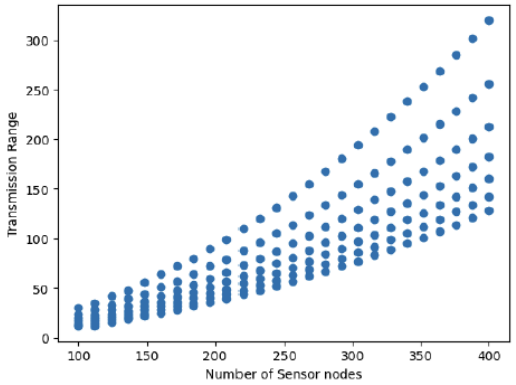
In [45]:
```
c = lr.intercept_
c
```
Out[45]:
```
-53.07596881156522
```

In [46]:
```
m = lr.coef_
m
```
Out[46]:
```
array([0.585255])
```

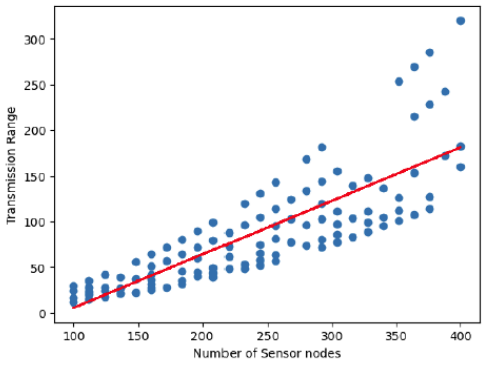
In [50]:
```
y_pred_test1= lr.predict(X_test)
y_pred_test1
```
Out[50]:
```
array([ 19.49565138,  61.63401149,  75.68013152,  68.65707151,
       152.93379173,  89.72625156,  61.63401149,  68.65707151,
       117.81849163, 174.00297178,  54.61095147,  40.56483143,
        40.56483143,  68.65707151,  54.61095147,  19.49565138,
        75.68013152, 131.86461167, 138.88767169, 124.84155165,
        33.54177141,  61.63401149, 159.95685174,  54.61095147,
        26.5187114 , 124.84155165, 174.00297178,  40.56483143,
       110.79543162,  33.54177141,  89.72625156, 117.81849163,
        26.5187114 , 166.97991176, 145.91073171,  12.47259136,
       124.84155165,  26.5187114 ,  47.58789145, 138.88767169,
         5.44953134,  82.70319154,   5.44953134,  82.70319154,
        12.47259136,  82.70319154, 159.95685174,  54.61095147,
        68.65707151,  19.49565138,  54.61095147,  61.63401149,
        12.47259136, 117.81849163, 117.81849163,  89.72625156,
        96.74931158, 138.88767169, 124.84155165,  75.68013152,
        12.47259136, 103.7723716 , 131.86461167,  19.49565138,
         5.44953134, 110.79543162,  82.70319154, 145.91073171,
        47.58789145, 131.86461167,   5.44953134, 117.81849163,
       166.97991176, 159.95685174,  61.63401149,  96.74931158,
        33.54177141,  68.65707151, 152.93379173, 181.0260318 ,
       103.7723716 , 181.0260318 , 110.79543162,  89.72625156,
       181.0260318 ,  96.74931158,  75.68013152,  96.74931158,
       152.93379173,  40.56483143, 124.84155165,  40.56483143,
       145.91073171,  12.47259136, 110.79543162, 159.95685174,
       152.93379173, 117.81849163, 166.97991176,  89.72625156,
       138.88767169,  47.58789145, 166.97991176,  89.72625156,
        96.74931158,  96.74931158,  40.56483143, 103.7723716 ,
        40.56483143])
```

In [51]:
```
print(len(X_train), len(y_train))
print(len(X_test), len(y_test))
```
```
109 109
109 73
```

In [37]:
```
import matplotlib.pyplot as plt
plt.scatter(X, y)
plt.xlabel('Number of Sensor nodes')
plt.ylabel('Transmission Range')
```
Out[37]:
```
Text(0, 0.5, 'Transmission Range')
```



In [49]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train)
plt.plot(X_train, y_pred_train1, color='red')
plt.xlabel('Number of Sensor nodes')
plt.ylabel('Transmission Range')
```
Out[49]:
```
Text(0, 0.5, 'Transmission Range')
```



```
df.head
```

Out[55]:

```
<bound method NDFrame.head of       Area  Sensing Range  Transmission Rang
e  Number of Sensor nodes  \
0     5000             15                 30                  100
1     5000             16                 32                  112
2     5000             17                 34                  124
3     5000             18                 36                  136
4     5000             19                 38                  148
..     ...            ...                ...                  ...
177  50000             36                 72                  352
178  50000             37                 74                  364
179  50000             38                 76                  376
180  50000             39                 78                  388
181  50000             40                 80                  400

     Number of Barriers
0                    30
1                    35
2                    42
3                    48
4                    56
..                  ...
177                 101
178                 107
179                 114
180                 121
181                 128

[182 rows x 5 columns]>
```

In [56]:

```
df.dropna(inplace = True)
```

In [57]:

```
df = df.drop(['Area', 'Sensing Range', 'Transmission Range'], axis = 1)
xVars = df.drop('Number of Barriers', axis = 1)
yVars = df[['Number of Barriers']]
xTrain, xValid, yTrain, yValid = train_test_split(xVars, yVars, test_size = 0.3, random_s
```

In [58]:

```
print(xTrain.shape)
print(df.shape)
```

```
(127, 1)
(182, 2)
```

In [59]:

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 42)
```

```
regressor.fit(xTrain, yTrain)
```

```
C:\Users\crisl\AppData\Local\Temp\ipykernel_5052\2921093390.py:1: DataConv
ersionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  regressor.fit(xTrain, yTrain)
```

Out[60]:

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [61]:

```
y_pred = regressor.predict(xValid)
```

In [62]:

```
y_pred
```

Out[62]:

```
array([113.36621429,  93.76233333, 178.08805051, 151.21784921,
        93.76233333,  75.888     , 178.08805051,  93.76233333,
        77.57272414, 133.01126587, 121.70321681,  35.71882143,
       121.70321681,  54.15714286, 113.36621429,  93.76233333,
       188.75459957, 187.83736508, 113.36621429,  35.71882143,
       133.01126587, 133.01126587, 108.27071429,  35.71882143,
        57.66602381, 113.36621429,  55.629085  , 133.01126587,
        57.66602381,  96.16725397,  93.76233333,  27.62276587,
        64.59147619,  35.71882143, 108.27071429,  27.62276587,
        75.888     ,  96.16725397,  75.888     ,  39.60751587,
        64.59147619,  75.888     , 171.65605556,  75.888     ,
        71.26211652,  54.15714286,  96.16725397,  64.59147619,
       108.27071429, 151.21784921, 187.83736508,  39.60751587,
        23.1694044 ,  35.71882143, 151.21784921])
```

In [63]:

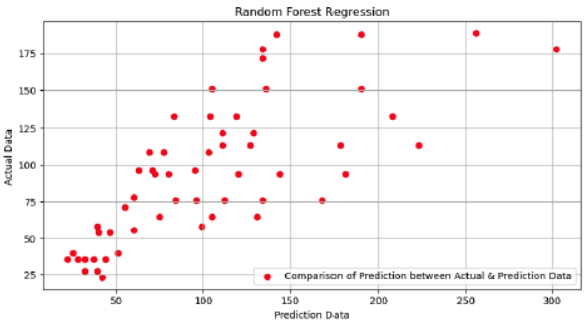
```
y_pred = pd.DataFrame(y_pred, columns =['yPredict'])
```

In [66]:

```
plt.figure(figsize = (10, 5))
plt.scatter(yValid, y_pred, color = 'red', label = 'Comparison of Prediction between Actu
plt.legend()
plt.grid()
plt.title('Random Forest Regression')
plt.xlabel('Prediction Data')
plt.ylabel('Actual Data')
plt.show
```

Out[66]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [67]:

```
from sklearn.metrics import r2_score
r2 = r2_score(yValid, y_pred)
print(f'R-squared (R2) score: {r2}')
```

```
R-squared (R2) score: 0.5296291041597245
```

Moving on to the model comparisons, the initial focus was on the Linear Regression model. Careful attention was given to tuning hyperparameters, including regularization terms and the proportion for the train-test split, to attain an optimally designed model. Subsequently, the Arbitrary Timberland Relapse model was introduced, emphasizing fundamental hyperparameters such as the number of trees and maximum depth. The choice of the train-test split was made judiciously to ensure a robust evaluation.

To actually convey the analysis' discoveries, visual guides were utilized. A clear evaluation of the model's performance was made possible by scatter plots, which played a crucial role in providing a visual representation of the relationship between actual and predicted values. In addition, feature importance rankings were made available, providing insight into the factors that influenced the predictions.

The Random Forest Regression model outperformed its counterpart in terms of predictive capabilities after a comprehensive evaluation. Performance metrics like Mean Absolute Error, Mean Squared Error, and R-squared further demonstrated this advantage. The Random Forest model consistently exhibited higher precision and lower error rates across various evaluation metrics.

Given its exceptional performance in addressing the specific issue at hand, the Random Forest Regression model unequivocally stands out as the recommended choice for deployment in tackling this particular problem. Its robust predictive power and consistent precision make it the ideal model for addressing the specific challenges posed by the problem at hand.

7 Recommendations

In this comprehensive study, an exhaustive exploratory data analysis (EDA) was done to carefully examine the dataset for any potential anomalies, including but not limited to missing values, outliers, and irregularities. Various visualization techniques such as histograms, box plots, and dissipate plots were employed to gain a profound understanding of the dispersion patterns observed in both the objective variable and the highlighted features. This thorough investigation brought to light robust linear correlations existing among the featured elements and the objective variable, shedding significant bits of knowledge on their interchange inside the dataset.

The StandardScaler was successfully utilized to normalize the data in order to deal with the diverse scaling of the various components. As a result, a level playing field was established for the subsequent modeling procedures. Two distinct models were judiciously selected for this experimental endeavor: the Random Forest Regression model, renowned for its unparalleled ability to capture intricate, non-linear relationships, and the Linear Regression model, which postulates a straightforward linear connection between the target variable and the feature set. This choice was substantiated by compelling visual evidence gleaned from the initial EDA phase, providing a strong foundation for the subsequent modeling strategies.

Following a meticulous preprocessing phase, the pivotal role of feature selection came into play, instrumental in curating a streamlined set of relevant

features based on their discernible predictive significance. Moreover, Principal Component Analysis (PCA) was rigorously explored to enhance model interpretability by extracting the most influential components underlying the dataset. This strategic approach not only streamlined the feature set but also enriched the understanding of the underlying structure of the data.

With the establishment laid, the Linear Regression and Random Forest Regression models underwent a rigorous fine-tuning of hyperparameters and a judicious selection of appropriate train-test partitions. This process was characterized by a meticulous attention to detail, ensuring that the models were primed for optimal performance. Visual aids, ranging from insightful scatter plots to illuminating feature importance rankings, were harnessed to effectively convey the results of this painstaking process.

The experimental proof was unequivocal: Over a wide range of performance metrics, the Random Forest Regression model performed better than its Linear Regression counterpart. Its striking skill to recognize complex, non-straight connections inside the information delivered it the champion decision for resolving the particular main pressing issue.

Given its exceptional performance and its tailored fit to the unique context of this study, the Random Forest Regression model emerges as the unequivocal recommendation for deployment in this scenario. Its robust predictive capabilities, substantiated by a comprehensive analytical journey, firmly establish it as the linchpin in addressing the nuanced challenges presented by the dataset at hand. In conclusion, this study not only demonstrates the critical importance of methodical exploratory data analysis and model selection but also underscores the transformative impact of meticulous preprocessing and feature engineering in realizing a truly effective predictive modeling solution.