COMPUTER NETWORKS LAB PROJECT

NETWORK INTRUSION DETECTION USING MACHINE LEARNING

Group Members:

Vatsal Gupta (17104060)
Arjav Jain (17104070)
Ananya Sharma (17104038)

Title: Network Intrusion Detection using Machine Learning

Problem Statement

With the rapid development of information technology in the past two decades, computer networks are being widely used by industry, business and various fields of human life. Therefore, building reliable networks has become an essential task for IT administrators. However, the rapid development of the IT sector has brought to the forefront several challenges towards building reliable computer networks. There are many types of attacks threatening the availability, integrity and confidentiality of computer networks. The Denial of Service attacks (DoS), Remote to Local (R2L) and, User to Root (U2R) attacks are considered to be some of the most harmful attacks to any network.

Introduction and Proposed Model

With the advancement in technology, millions of people are now connected through one or other form of a network where they share lots of valuable data. Hence the need for security to safeguard data integrity and confidentiality is increased rapidly. Although effort has been made to secure data transmission at the same time, the attack technique for breaching the network continued to evolve. Thus it leads to the need for such a system which can adapt with this ever-changing attack technique. In this paper, we have purposed a system which is based on machine learning. We aim to find the most suitable machine learning algorithm which can predict the type of network attack with the highest accuracy. The algorithms which we have compared are Decision Tree, Naïve Bayes, K Nearest Neighbor (KNN), Random Forest and Logistic Regression. The dataset used for training the model is NSL - KDD revised dataset. Machine learning has been used to detect the intrusion attacks due to its flexibility; for example, if any new type of attack is developed in future, the system can be trained for predicting that attack. In our project, we have made a knowledge-based intrusion detection system which is also known as the anomaly-based system. It registers the anomalies and in future predicts such malicious network to send out an alert. This way, the network can disconnect to such a connection and then have only secured connections.

Hardware Requirements:

- Minimum memory requirement for smooth working of the code: 4 GB RAM
- Processor: Preferably intel i5 (atleast dual core) and later

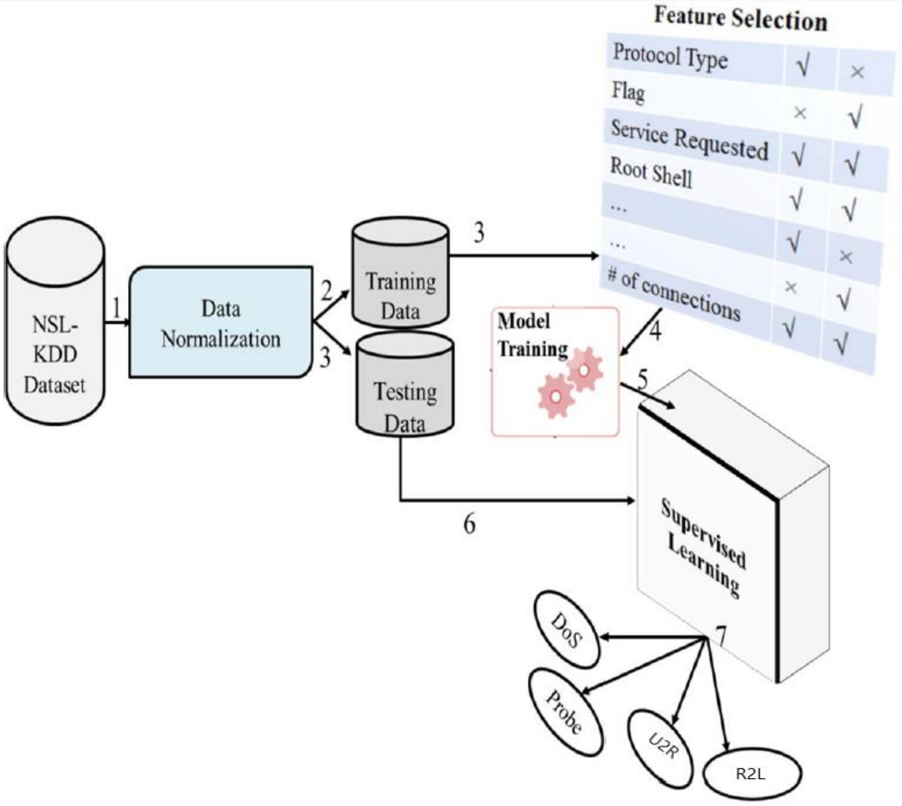
Software Requirements:

- Python 3.x
- Any python IDE (preferably Spyder IDE 3.3.2 or PyCharm)

Packages/Modules Used (can be installed using pip installer)

- warnings
- matplotlib
- pandas
- numpy
- seaborn
- sklearn
- imblearn

Data Flow Diagram



Dataset Description

The machine learning algorithms for intrusion detection have been applied on the freely available NSL – KDD revised dataset. It is an easily accessible dataset that is being relied upon by many researchers for their works on intrusion detection. The dataset contains many attack types like the DOS, U2R, R2L, Probe and normal (no attack). There are 21 types of attacks inside the main categories mentioned above.

Categories of Attack	Attack name	Number of instances
DOS	SMURF	2807886
	NEPTUNE	1072017
	Back	2203
	POD	264
	Teardrop	979
U2R	Buffer overflow	30
	Load Module	9
	PERL	3
	Rootkit	10
R2L	FTP Write	8
	Guess Passwd	53
	IMAP	12
	MulitHop	7
	PHF	4
	SPY	2
	Warez client	1020
	Warez Master	20
PROBE	IPSWEEP	12481
	NMAP	2316
	PORTSWEEP	10413
	SATAN	15892
normal		972781

The dataset contains a total of 41 attributes which could be used to determine if the attack is malicious or not at all an attack.

Description of flag values in the dataset is given in the following table:

Flag	Description
RSTOS0	Originator sent a SYN followed by a RST, never see a SYN ACK from the responder
RSTR	Established, responder aborted
RSTO	Connection established, originator aborted (sent a RST)
OTH	No SYN seen, just midstream traffic (a "partial connection" that was not later closed)
REJ	Connection attempt rejected
S0	Connection attempt seen, no reply
S1	Connection established, not terminated
S2	Connection established and close attempt by originator seen (but no reply from responder)
S3	Connection established and close attempt by responder seen (but no reply from originator)
SF	Normal establishment and termination
SH	Originator sent a SYN followed by a FIN (finish 'flag'), never saw a SYN ACK from the responder (hence the connection was "half" open)

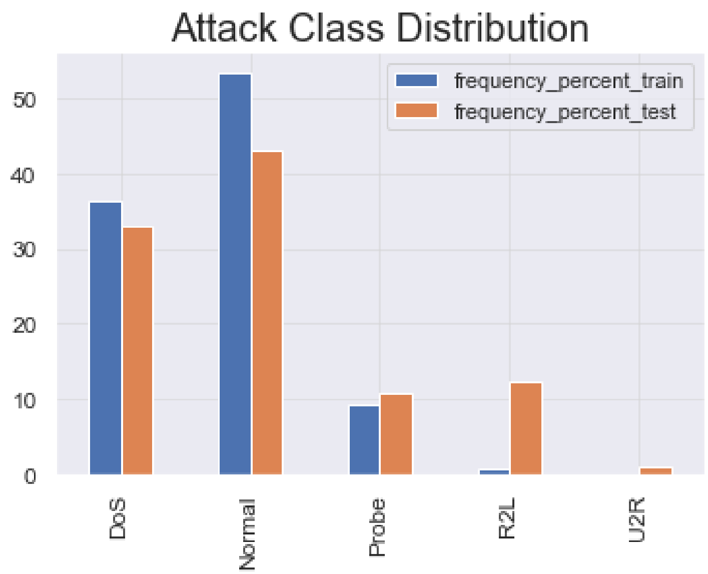
Working Model

The NSL - KDD is a huge dataset, and the one that we have used in our research is under the folder 'Dataset'. Our aim is to not only to find the best algorithm suited for the intrusion detection problem but also to implement it using the programming language python. The first process of applying machine learning is always pre-processing the data. This process involves various steps like 1) removing the redundant rows from the dataset 2) to see whether there are any missing values and then to remove those corresponding rows too.
The next process is to use this dataset and put it across various machine-learning algorithms that might give good results by correctly classifying the instances.

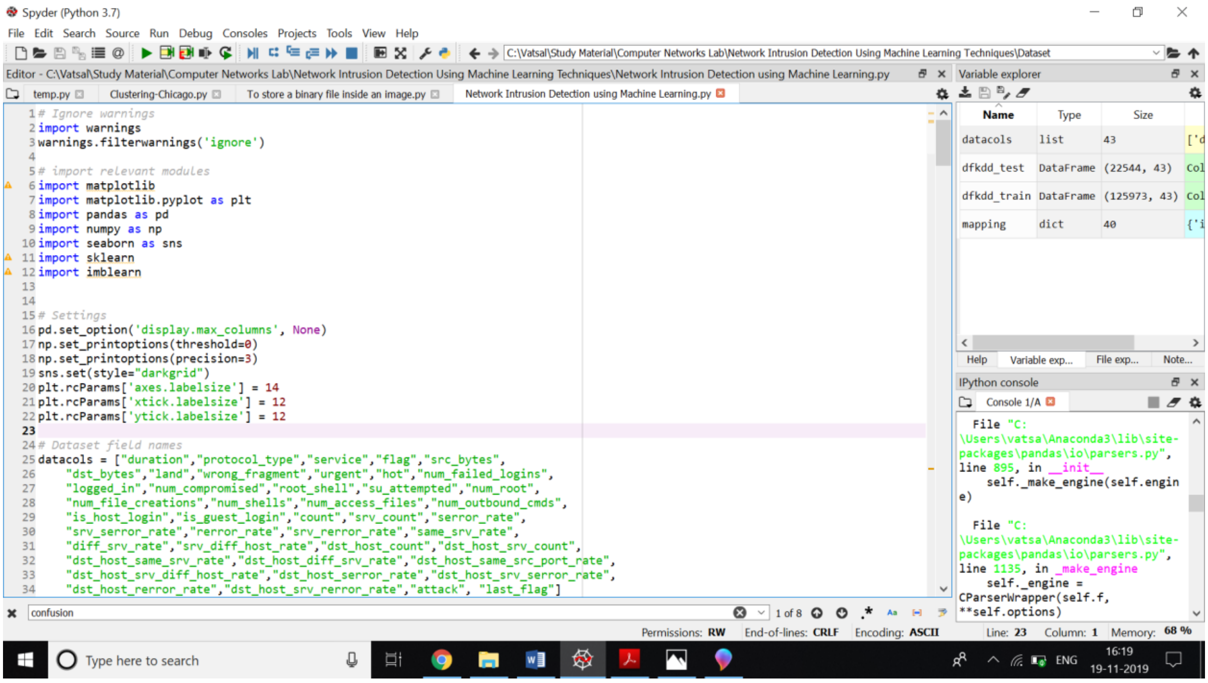
From our results, it can be inferred that the best algorithm that can be used for the network intrusion detection is the Random Forest. Random Forest algorithm is a classification algorithm based on ensemble learning. It works by building multiple decision trees at training, and the developed decision trees form the output function. The drawback of this algorithm is that is computation intensive and consequently has high computational time.

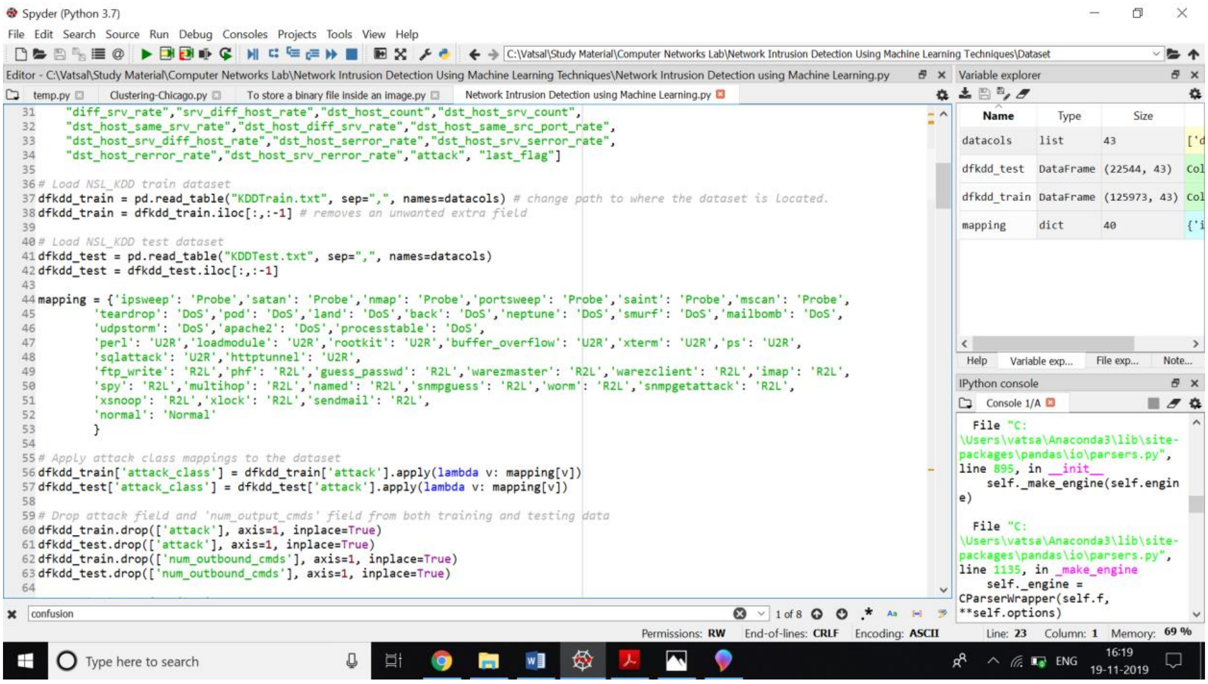
This program, when developed fully, will act as a filter to determine if a network is secure and will continuously learn from its own series of data making it better and stronger with each type of attack.

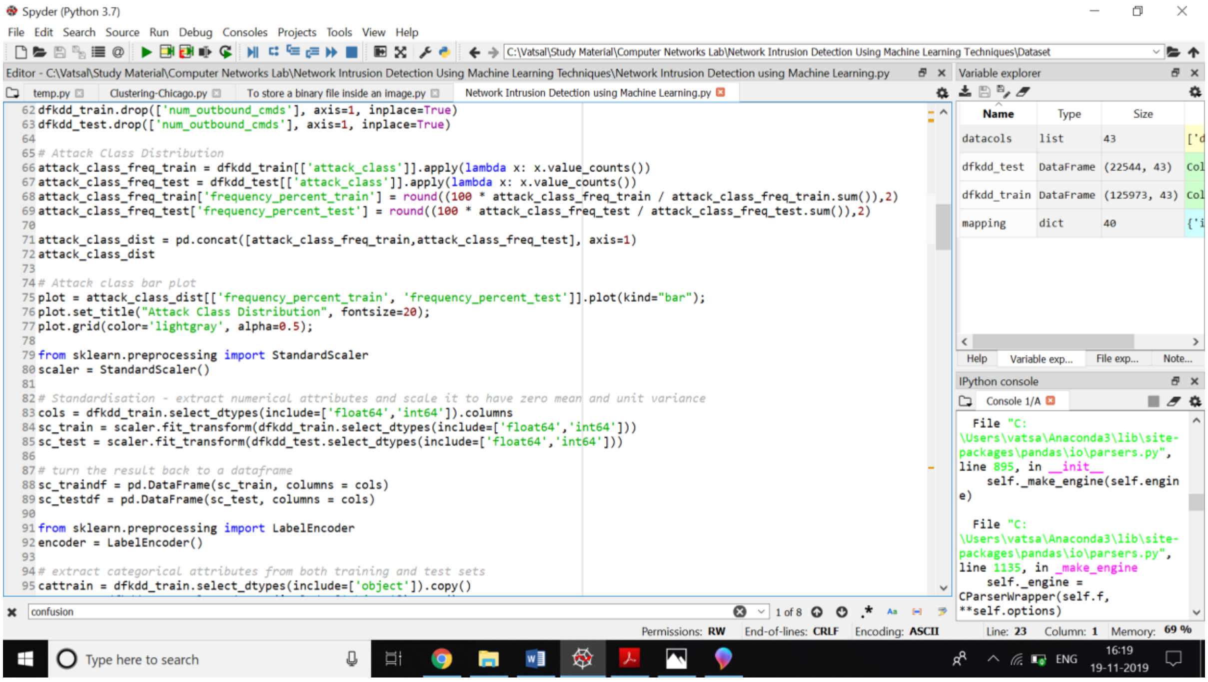
Attack Class Distribution

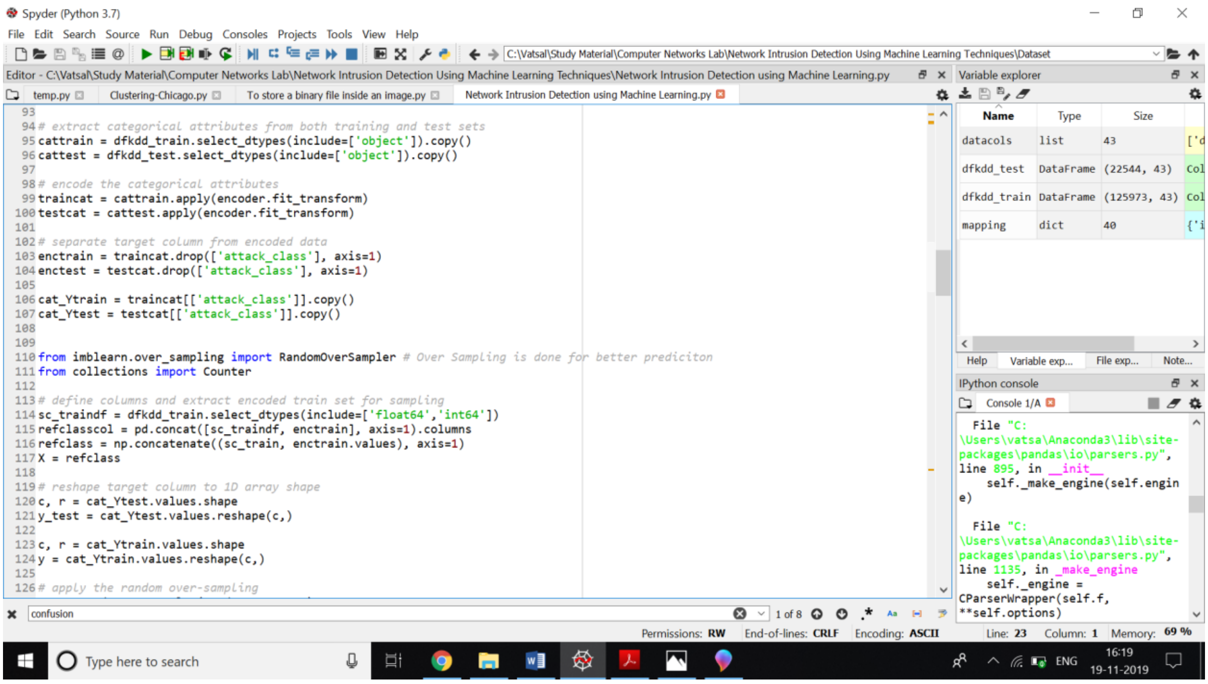


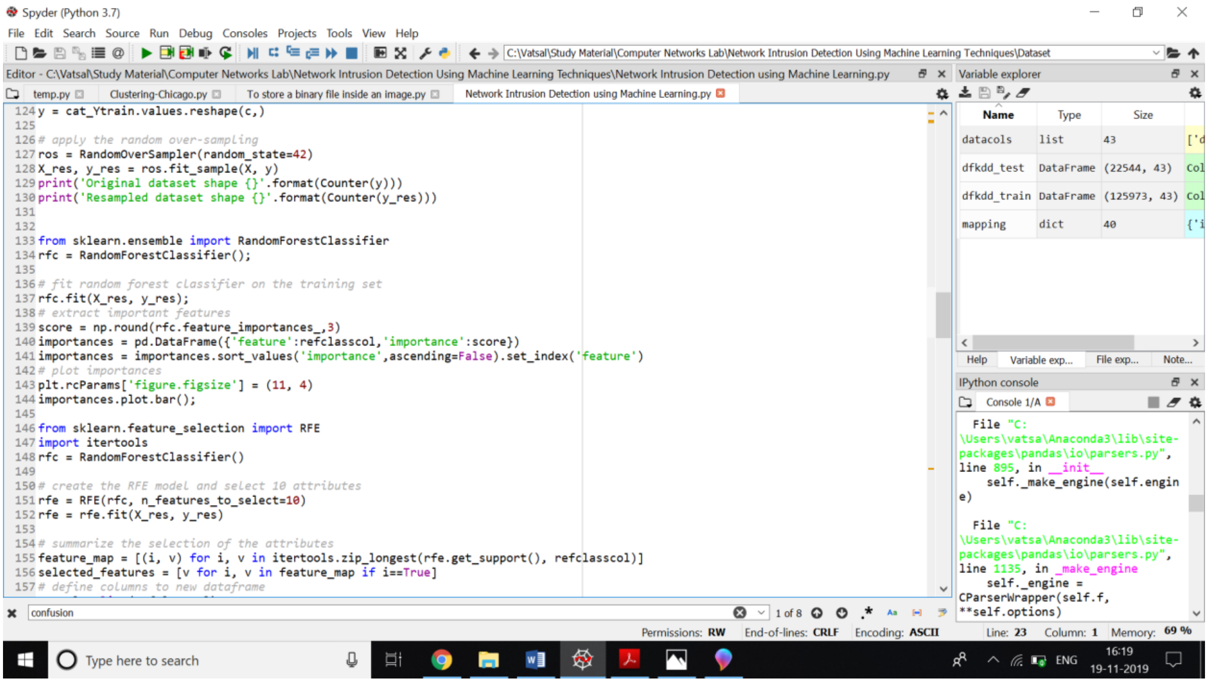
Screenshots of the code:

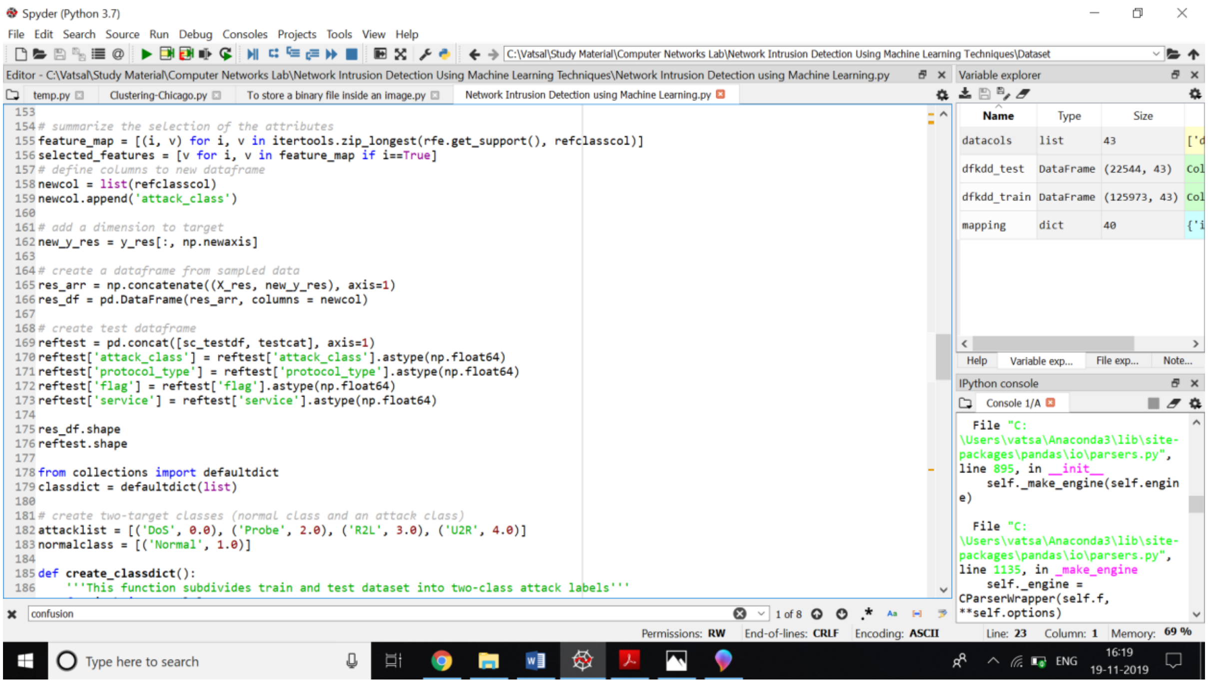

Spyder (Python 3.7)
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
# import relevant modules
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import imblearn
# Settings
pd.set_option('display.max_columns', None)
np.set_printoptions(threshold=0)
np.set_printoptions(precision=3)
sns.set(style="darkgrid")
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
# Dataset field names
datacols = ["duration","protocol_type","service","flag","src_bytes",
"dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
"logged_in","num_compromised","root_shell","su_attempted","num_root",
"num_file_creations","num_shells","num_access_files","num_outbound_cmds",
"is_host_login","is_guest_login","count","srv_count","serror_rate",
"srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","attack", "last_flag"]
Variable explorer
Name
Type
Size
datacols
list
43
dfkdd_test
DataFrame
(22544, 43)
dfkdd_train
DataFrame
(125973, 43)
mapping
dict
40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 23
Column: 1
Memory: 68 %
Type here to search
ENG
16:19
19-11-2019

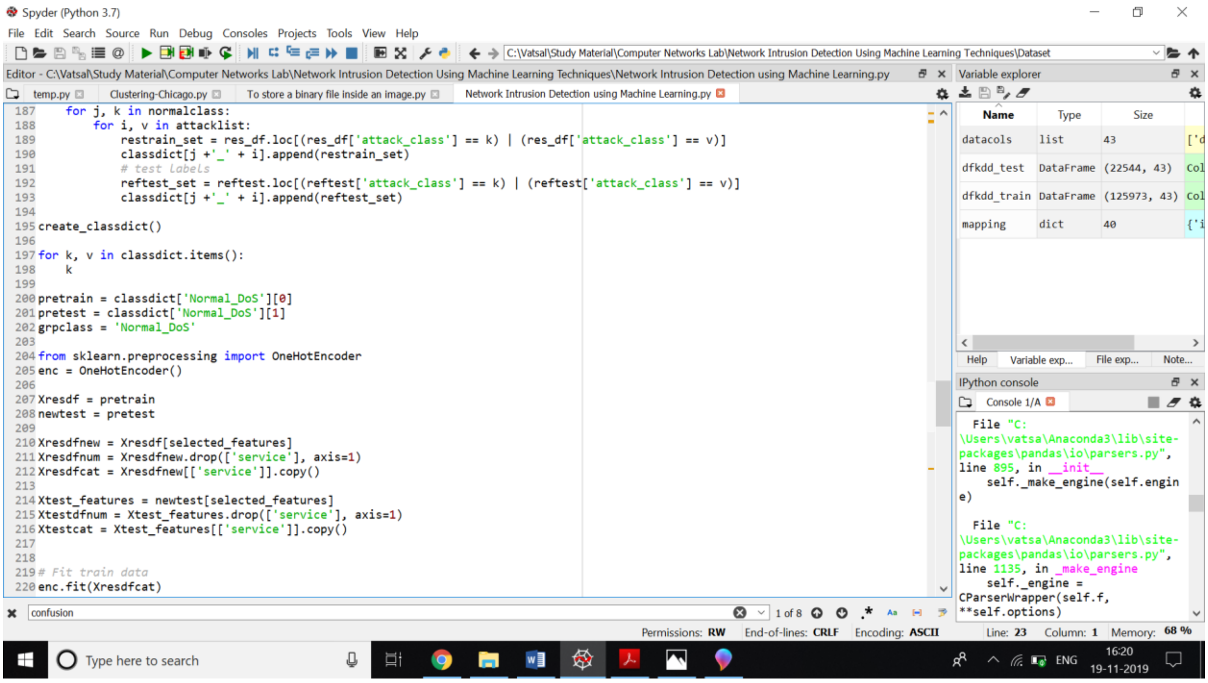

Spyder (Python 3.7)
"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
"dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
"dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
"dst_host_rerror_rate","dst_host_srv_rerror_rate","attack", "last_flag"]
# Load NSL_KDD train dataset
dfkdd_train = pd.read_table("KDDTrain.txt", sep=",", names=datacols) # change path to where the dataset is located.
dfkdd_train = dfkdd_train.iloc[:,:-1] # removes an unwanted extra field
# Load NSL_KDD test dataset
dfkdd_test = pd.read_table("KDDTest.txt", sep=",", names=datacols)
dfkdd_test = dfkdd_test.iloc[:,:-1]
mapping = {'ipsweep': 'Probe','satan': 'Probe','nmap': 'Probe','portsweep': 'Probe','saint': 'Probe','mscan': 'Probe',
'teardrop': 'DoS','pod': 'DoS','land': 'DoS','back': 'DoS','neptune': 'DoS','smurf': 'DoS','mailbomb': 'DoS',
'udpstorm': 'DoS','apache2': 'DoS','processtable': 'DoS',
'perl': 'U2R','loadmodule': 'U2R','rootkit': 'U2R','buffer_overflow': 'U2R','xterm': 'U2R','ps': 'U2R',
'sqlattack': 'U2R','httptunnel': 'U2R',
'ftp_write': 'R2L','phf': 'R2L','guess_passwd': 'R2L','warezmaster': 'R2L','warezclient': 'R2L','imap': 'R2L',
'spy': 'R2L','multihop': 'R2L','named': 'R2L','snmpguess': 'R2L','worm': 'R2L','snmpgetattack': 'R2L',
'xsnoop': 'R2L','xlock': 'R2L','sendmail': 'R2L',
'normal': 'Normal'
}
# Apply attack class mappings to the dataset
dfkdd_train['attack_class'] = dfkdd_train['attack'].apply(lambda v: mapping[v])
dfkdd_test['attack_class'] = dfkdd_test['attack'].apply(lambda v: mapping[v])
# Drop attack field and 'num_output_cmds' field from both training and testing data
dfkdd_train.drop(['attack'], axis=1, inplace=True)
dfkdd_test.drop(['attack'], axis=1, inplace=True)
dfkdd_train.drop(['num_outbound_cmds'], axis=1, inplace=True)
dfkdd_test.drop(['num_outbound_cmds'], axis=1, inplace=True)
Variable explorer
IPython console
Console 1/A
confusion
1 of 8
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 23
Column: 1
Memory: 69 %
Type here to search
ENG
16:19
19-11-2019

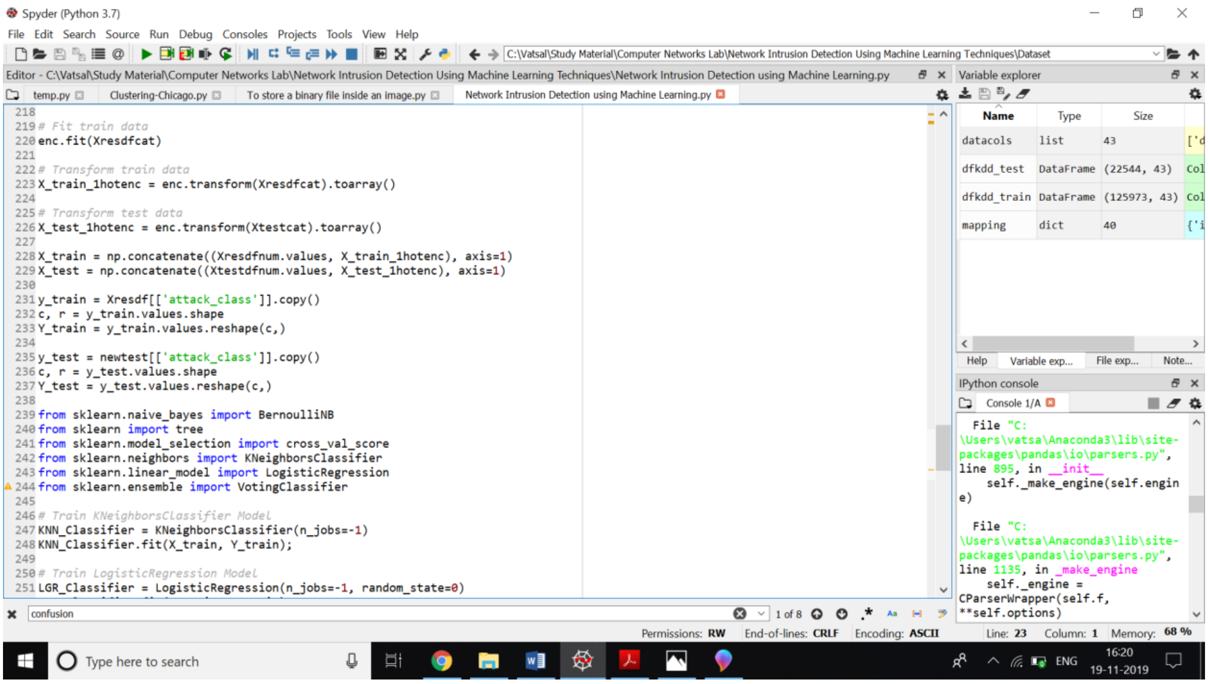

Spyder (Python 3.7)
dfkdd_train.drop(['num_outbound_cmds'], axis=1, inplace=True)
dfkdd_test.drop(['num_outbound_cmds'], axis=1, inplace=True)
# Attack Class Distribution
attack_class_freq_train = dfkdd_train[['attack_class']].apply(lambda x: x.value_counts())
attack_class_freq_test = dfkdd_test[['attack_class']].apply(lambda x: x.value_counts())
attack_class_freq_train['frequency_percent_train'] = round((100 * attack_class_freq_train / attack_class_freq_train.sum()),2)
attack_class_freq_test['frequency_percent_test'] = round((100 * attack_class_freq_test / attack_class_freq_test.sum()),2)
attack_class_dist = pd.concat([attack_class_freq_train,attack_class_freq_test], axis=1)
attack_class_dist
# Attack class bar plot
plot = attack_class_dist[['frequency_percent_train', 'frequency_percent_test']].plot(kind="bar");
plot.set_title("Attack Class Distribution", fontsize=20);
plot.grid(color='lightgray', alpha=0.5);
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# Standardisation - extract numerical attributes and scale it to have zero mean and unit variance
cols = dfkdd_train.select_dtypes(include=['float64','int64']).columns
sc_train = scaler.fit_transform(dfkdd_train.select_dtypes(include=['float64','int64']))
sc_test = scaler.fit_transform(dfkdd_test.select_dtypes(include=['float64','int64']))
# turn the result back to a dataframe
sc_traindf = pd.DataFrame(sc_train, columns = cols)
sc_testdf = pd.DataFrame(sc_test, columns = cols)
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
# extract categorical attributes from both training and test sets
cattrain = dfkdd_train.select_dtypes(include=['object']).copy()
Variable explorer
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
16:19
19-11-2019

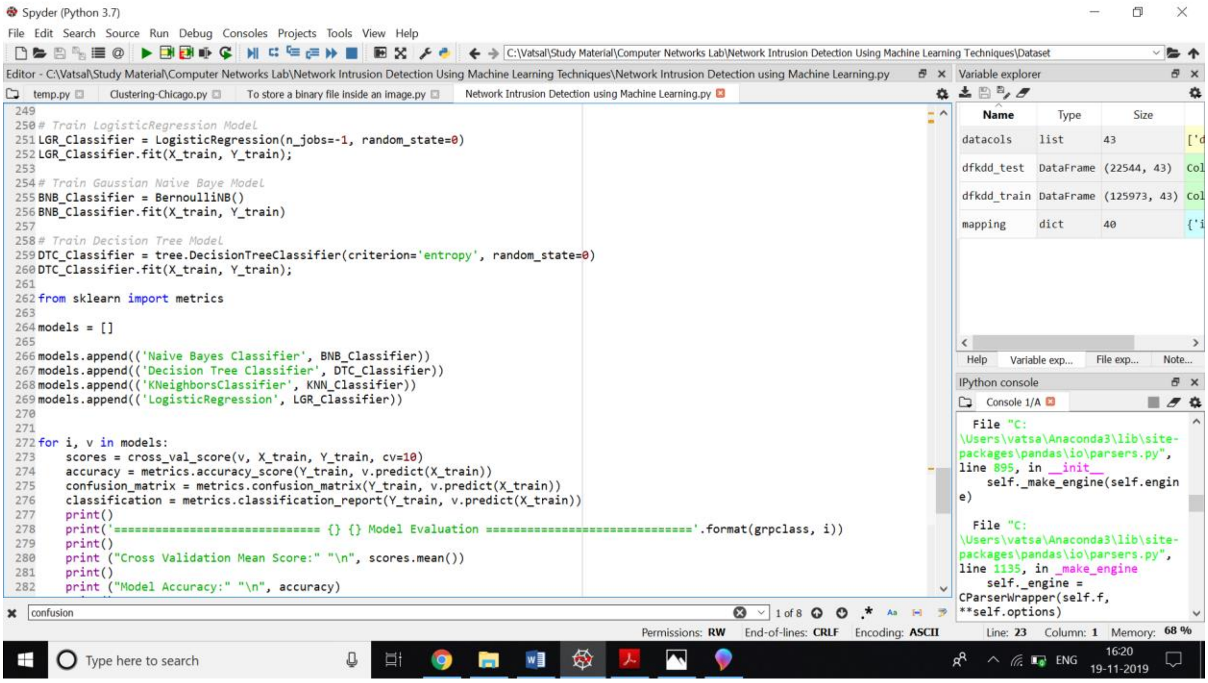

Spyder (Python 3.7)
# extract categorical attributes from both training and test sets
cattrain = dfkdd_train.select_dtypes(include=['object']).copy()
cattest = dfkdd_test.select_dtypes(include=['object']).copy()
# encode the categorical attributes
traincat = cattrain.apply(encoder.fit_transform)
testcat = cattest.apply(encoder.fit_transform)
# separate target column from encoded data
enctrain = traincat.drop(['attack_class'], axis=1)
enctest = testcat.drop(['attack_class'], axis=1)
cat_Ytrain = traincat[['attack_class']].copy()
cat_Ytest = testcat[['attack_class']].copy()
from imblearn.over_sampling import RandomOverSampler # Over Sampling is done for better prediciton
from collections import Counter
# define columns and extract encoded train set for sampling
sc_traindf = dfkdd_train.select_dtypes(include=['float64','int64'])
refclasscol = pd.concat([sc_traindf, enctrain], axis=1).columns
refclass = np.concatenate((sc_train, enctrain.values), axis=1)
X = refclass
# reshape target column to 1D array shape
c, r = cat_Ytest.values.shape
y_test = cat_Ytest.values.reshape(c,)
c, r = cat_Ytrain.values.shape
y = cat_Ytrain.values.reshape(c,)
# apply the random over-sampling
16:19
19-11-2019

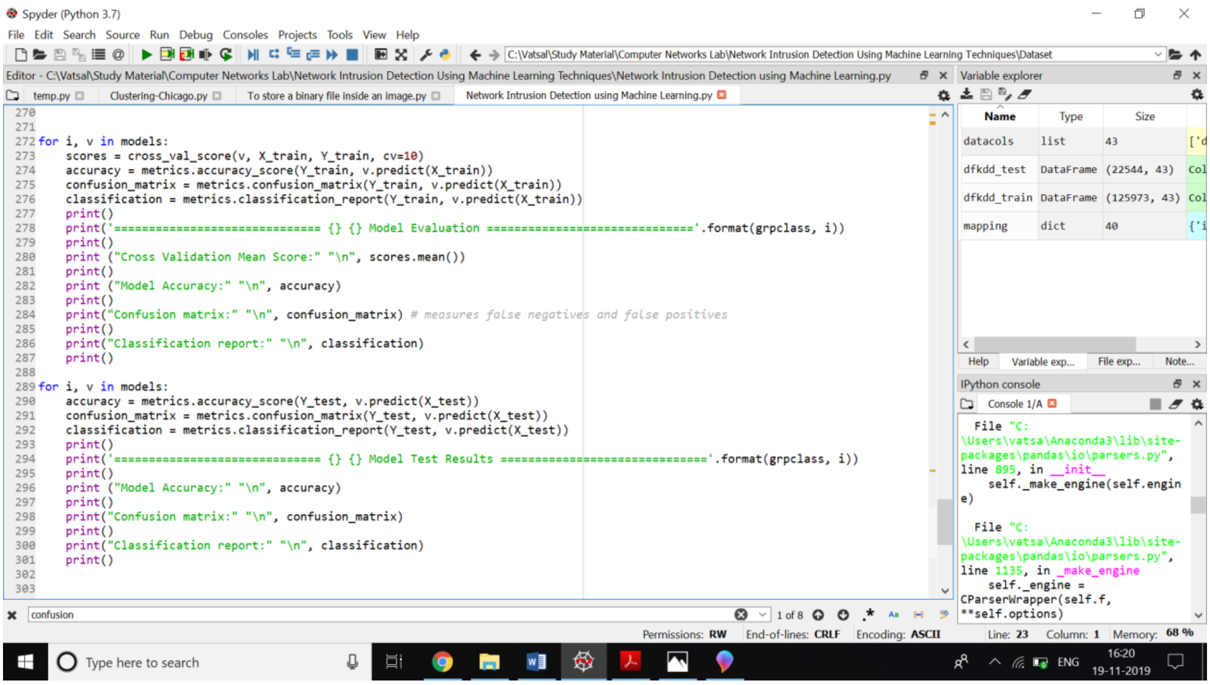

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
y = cat_Ytrain.values.reshape(c,)
# apply the random over-sampling
ros = RandomOverSampler(random_state=42)
X_res, y_res = ros.fit_sample(X, y)
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier();
# fit random forest classifier on the training set
rfc.fit(X_res, y_res);
# extract important features
score = np.round(rfc.feature_importances_,3)
importances = pd.DataFrame({'feature':refclasscol,'importance':score})
importances = importances.sort_values('importance',ascending=False).set_index('feature')
# plot importances
plt.rcParams['figure.figsize'] = (11, 4)
importances.plot.bar();
from sklearn.feature_selection import RFE
import itertools
rfc = RandomForestClassifier()
# create the RFE model and select 10 attributes
rfe = RFE(rfc, n_features_to_select=10)
rfe = rfe.fit(X_res, y_res)
# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), refclasscol)]
selected_features = [v for i, v in feature_map if i==True]
# define columns to new dataframe
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 69 %
Type here to search
16:19
19-11-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
# summarize the selection of the attributes
feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), refclasscol)]
selected_features = [v for i, v in feature_map if i==True]
# define columns to new dataframe
newcol = list(refclasscol)
newcol.append('attack_class')
# add a dimension to target
new_y_res = y_res[:, np.newaxis]
# create a dataframe from sampled data
res_arr = np.concatenate((X_res, new_y_res), axis=1)
res_df = pd.DataFrame(res_arr, columns = newcol)
# create test dataframe
reftest = pd.concat([sc_testdf, testcat], axis=1)
reftest['attack_class'] = reftest['attack_class'].astype(np.float64)
reftest['protocol_type'] = reftest['protocol_type'].astype(np.float64)
reftest['flag'] = reftest['flag'].astype(np.float64)
reftest['service'] = reftest['service'].astype(np.float64)
res_df.shape
reftest.shape
from collections import defaultdict
classdict = defaultdict(list)
# create two-target classes (normal class and an attack class)
attacklist = [('DoS', 0.0), ('Probe', 2.0), ('R2L', 3.0), ('U2R', 4.0)]
normalclass = [('Normal', 1.0)]
def create_classdict():
'''This function subdivides train and test dataset into two-class attack labels'''
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 69 %
Type here to search
16:19
19-11-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
    for j, k in normalclass:
        for i, v in attacklist:
            restrain_set = res_df.loc[(res_df['attack_class'] == k) | (res_df['attack_class'] == v)]
            classdict[j +'_' + i].append(restrain_set)
            # test labels
            reftest_set = reftest.loc[(reftest['attack_class'] == k) | (reftest['attack_class'] == v)]
            classdict[j +'_' + i].append(reftest_set)
create_classdict()
for k, v in classdict.items():
    k
pretrain = classdict['Normal_DoS'][0]
pretest = classdict['Normal_DoS'][1]
grpclass = 'Normal_DoS'
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()
Xresdf = pretrain
newtest = pretest
Xresdfnew = Xresdf[selected_features]
Xresdfnum = Xresdfnew.drop(['service'], axis=1)
Xresdfcat = Xresdfnew[['service']].copy()
Xtest_features = newtest[selected_features]
Xtestdfnum = Xtest_features.drop(['service'], axis=1)
Xtestcat = Xtest_features[['service']].copy()
# Fit train data
enc.fit(Xresdfcat)
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 68 %
Type here to search
ENG 16:20 19-11-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
# Fit train data
enc.fit(Xresdfcat)
# Transform train data
X_train_1hotenc = enc.transform(Xresdfcat).toarray()
# Transform test data
X_test_1hotenc = enc.transform(Xtestcat).toarray()
X_train = np.concatenate((Xresdfnum.values, X_train_1hotenc), axis=1)
X_test = np.concatenate((Xtestdfnum.values, X_test_1hotenc), axis=1)
y_train = Xresdf[['attack_class']].copy()
c, r = y_train.values.shape
Y_train = y_train.values.reshape(c,)
y_test = newtest[['attack_class']].copy()
c, r = y_test.values.shape
Y_test = y_test.values.reshape(c,)
from sklearn.naive_bayes import BernoulliNB
from sklearn import tree
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import VotingClassifier
# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);
# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 68 %
Type here to search
ENG 16:20 19-11-2019


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Dataset
Editor - C:\Vatsal\Study Material\Computer Networks Lab\Network Intrusion Detection Using Machine Learning Techniques\Network Intrusion Detection using Machine Learning.py
temp.py
Clustering-Chicago.py
To store a binary file inside an image.py
Network Intrusion Detection using Machine Learning.py
# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);
# Train Gaussian Naive Baye Model
BNB_Classifier = BernoulliNB()
BNB_Classifier.fit(X_train, Y_train)
# Train Decision Tree Model
DTC_Classifier = tree.DecisionTreeClassifier(criterion='entropy', random_state=0)
DTC_Classifier.fit(X_train, Y_train);
from sklearn import metrics
models = []
models.append(('Naive Bayes Classifier', BNB_Classifier))
models.append(('Decision Tree Classifier', DTC_Classifier))
models.append(('KNeighborsClassifier', KNN_Classifier))
models.append(('LogisticRegression', LGR_Classifier))
for i, v in models:
scores = cross_val_score(v, X_train, Y_train, cv=10)
accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
classification = metrics.classification_report(Y_train, v.predict(X_train))
print()
print('============================== {} {} Model Evaluation ==============================='.format(grpclass, i))
print()
print ("Cross Validation Mean Score:" "\n", scores.mean())
print()
print ("Model Accuracy:" "\n", accuracy)
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 68 %
Type here to search
ENG
16:20
19-11-2019


Spyder (Python 3.7)
Network Intrusion Detection using Machine Learning.py
for i, v in models:
scores = cross_val_score(v, X_train, Y_train, cv=10)
accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
classification = metrics.classification_report(Y_train, v.predict(X_train))
print()
print('============================== {} {} Model Evaluation ==============================='.format(grpclass, i))
print()
print ("Cross Validation Mean Score:" "\n", scores.mean())
print()
print ("Model Accuracy:" "\n", accuracy)
print()
print("Confusion matrix:" "\n", confusion_matrix) # measures false negatives and false positives
print()
print("Classification report:" "\n", classification)
print()
for i, v in models:
accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))
confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))
classification = metrics.classification_report(Y_test, v.predict(X_test))
print()
print('============================== {} {} Model Test Results ==============================='.format(grpclass, i))
print()
print ("Model Accuracy:" "\n", accuracy)
print()
print("Confusion matrix:" "\n", confusion_matrix)
print()
print("Classification report:" "\n", classification)
print()
Variable explorer
Name Type Size
datacols list 43
dfkdd_test DataFrame (22544, 43)
dfkdd_train DataFrame (125973, 43)
mapping dict 40
IPython console
Console 1/A
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 895, in __init__
self._make_engine(self.engine)
File "C:\Users\vatsa\Anaconda3\lib\site-packages\pandas\io\parsers.py", line 1135, in _make_engine
self._engine = CParserWrapper(self.f, **self.options)
confusion
1 of 8
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 23 Column: 1 Memory: 68 %
Type here to search
ENG
16:20
19-11-2019

Screenshots of the results:

```
============================== Normal_DoS LogisticRegression Model Test Results
==============================

Model Accuracy:
 0.8421573766672491

Confusion matrix:
 [[5963 1495]
 [1215 8496]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.83      0.80      0.81      7458
         1.0       0.85      0.87      0.86      9711

    accuracy                           0.84     17169
   macro avg       0.84      0.84      0.84     17169
weighted avg       0.84      0.84      0.84     17169
```

```
============================== Normal_DoS Naive Baye Classifier Model Evaluation
==============================

Cross Validation Mean Score:
 0.9737760721391011

Model Accuracy:
 0.9737686173767133

Confusion matrix:
 [[65346  1997]
 [ 1536 65807]]

Classification report:
               precision    recall  f1-score   support

         0.0       0.98      0.97      0.97     67343
         1.0       0.97      0.98      0.97     67343

    accuracy                           0.97    134686
   macro avg       0.97      0.97      0.97    134686
weighted avg       0.97      0.97      0.97    134686
```

```
============================== Normal_DoS Decision Tree Classifier Model Evaluation
==============================

Cross Validation Mean Score:
 0.9997698368967857

Model Accuracy:
 0.9999480272634127

Confusion matrix:
 [[67343     0]
 [    7 67336]]

Classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

```
============================== Normal_DoS KNeighborsClassifier Model Evaluation
==============================

Cross Validation Mean Score:
 0.9965698163470991

Model Accuracy:
 0.9977577476500898

Confusion matrix:
 [[67287    56]
 [  246 67097]]

Classification report:
               precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

References:

Research Papers Referred

- Kevric, J., Jukic, S., & Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, *28*(1), 1051-1058.

- Lee, C. H., Su, Y. Y., Lin, Y. C., & Lee, S. J. (2017, September). Machine learning based network intrusion detection. In *2017 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA)* (pp. 79-83). IEEE.

- Taher, K. A., Jisan, B. M. Y., & Rahman, M. M. (2019, January). Network Intrusion Detection using Supervised Machine Learning Technique with Feature Selection. In *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)* (pp. 643-646). IEEE.

For Dataset

https://www.unb.ca/cic/datasets/nsl.html