Logistic Regression

Goal:
The goal of this assignment is to train various Logistic Regression models on KDDCup99 dataset. You can find more details and download the dataset here: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Tasks:

1. Become familiar with Scikit-Learn's logistic regression models. You can find more details and examples here: https://scikit- learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

2. Download KDD99.csv dataset (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html)

3. Using the Scikit-Learn Library, train the Logistic Regression model to classify Probe vs Not Probe classes using all available features. Probe class consists of all probing attacks, such as ipsweep probe, nmap probe, portsweep probe, etc. Non probe class consists of every other attack in the dataset. Make sure you split your data into train and test sets. In general, 80% for training 20% for testing is a good split.

4. Summarize your results. What is the training accuracy? What is the testing accuracy? Do you think your models overfits? How many iterations did it take to converge?

5. Take a look at the trained model parameters. Which features have the largest weights in the absolute value? These are potentially most informative features for the trained model. Do they make sense? Why?

6. Train model with L1 and L2 regularization. Compare the accuracies for L1 and L2 and plot the feature weights.

7. Try different strengths for L2 regularization. (You can use values .001, .01, .1, 1, 10 etc.) Show how the train and test accuracy varies with different regularization strengths.

8. Carry out all the tasks on a single Jupyter notebook. Discuss your findings when appropriate. Is what you observe expected? Why?

Installing the required libraries

```
In [1]: !pip3 install plotly
```

```
Requirement already satisfied: plotly in ./opt/anaconda3/lib/python3.8/site-packages (5.3.1)
Requirement already satisfied: six in ./opt/anaconda3/lib/python3.8/site-packages (from plotly) (1.15.0)
Requirement already satisfied: tenacity>=6.2.0 in ./opt/anaconda3/lib/python3.8/site-packages (from plotly) (8.0.1)
```

```
In [2]: !pip3 install seaborn
```

```
Requirement already satisfied: seaborn in ./opt/anaconda3/lib/python3.8/site-packages (0.11.1)
Requirement already satisfied: numpy>=1.15 in ./opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.20.1)
Requirement already satisfied: scipy>=1.0 in ./opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.6.2)
Requirement already satisfied: pandas>=0.23 in ./opt/anaconda3/lib/python3.8/site-packages (from seaborn) (1.2.4)
Requirement already satisfied: matplotlib>=2.2 in ./opt/anaconda3/lib/python3.8/site-packages (from seaborn) (3.3.4
)
Requirement already satisfied: pillow>=6.2.0 in ./opt/anaconda3/lib/python3.8/site-packages (from matplotlib>=2.2->
seaborn) (8.2.0)
Requirement already satisfied: python-dateutil>=2.1 in ./opt/anaconda3/lib/python3.8/site-packages (from matplotlib
>=2.2->seaborn) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in ./opt/anaconda3/lib/python3.8/site-packages (from matplotlib>=2
.2->seaborn) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in ./opt/anaconda3/lib/python3.8/site-packa
ges (from matplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: cycler>=0.10 in ./opt/anaconda3/lib/python3.8/site-packages (from matplotlib>=2.2->s
eaborn) (0.10.0)
Requirement already satisfied: six in ./opt/anaconda3/lib/python3.8/site-packages (from cycler>=0.10->matplotlib>=2
.2->seaborn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in ./opt/anaconda3/lib/python3.8/site-packages (from pandas>=0.23->seab
orn) (2021.1)
```

```
In [3]: import plotly.express as px
        import pandas as pd
        import matplotlib
        import  matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
```

Datafiles

```
In [4]: df = pd.read_csv('kddcup99.csv')
        df
```

Out[4]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_h
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9	1.0	
1	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
494015	0	tcp	http	SF	310	1881	0	0	0	0	...	255	1.0	
494016	0	tcp	http	SF	282	2286	0	0	0	0	...	255	1.0	
494017	0	tcp	http	SF	203	1200	0	0	0	0	...	255	1.0	
494018	0	tcp	http	SF	291	1200	0	0	0	0	...	255	1.0	
494019	0	tcp	http	SF	219	1234	0	0	0	0	...	255	1.0	

494020 rows × 42 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 494020 entries, 0 to 494019
Data columns (total 42 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   duration                     494020 non-null  int64  
 1   protocol_type                494020 non-null  object 
 2   service                      494020 non-null  object 
 3   flag                         494020 non-null  object 
 4   src_bytes                    494020 non-null  int64  
 5   dst_bytes                    494020 non-null  int64  
 6   land                         494020 non-null  int64  
 7   wrong_fragment               494020 non-null  int64  
 8   urgent                       494020 non-null  int64  
 9   hot                          494020 non-null  int64  
 10  num_failed_logins            494020 non-null  int64  
 11  logged_in                    494020 non-null  int64  
 12  lnum_compromised             494020 non-null  int64  
 13  lroot_shell                  494020 non-null  int64  
 14  lsu_attempted                494020 non-null  int64  
 15  lnum_root                    494020 non-null  int64  
 16  lnum_file_creations          494020 non-null  int64  
 17  lnum_shells                  494020 non-null  int64  
 18  lnum_access_files            494020 non-null  int64  
 19  lnum_outbound_cmds           494020 non-null  int64  
 20  is_host_login                494020 non-null  int64  
 21  is_guest_login               494020 non-null  int64  
 22  count                        494020 non-null  int64  
 23  srv_count                    494020 non-null  int64  
 24  serror_rate                  494020 non-null  float64
 25  srv_serror_rate              494020 non-null  float64
 26  rerror_rate                  494020 non-null  float64
 27  srv_rerror_rate              494020 non-null  float64
 28  same_srv_rate                494020 non-null  float64
 29  diff_srv_rate                494020 non-null  float64
 30  srv_diff_host_rate           494020 non-null  float64
 31  dst_host_count               494020 non-null  int64  
 32  dst_host_srv_count           494020 non-null  int64  
```

```
 33  dst_host_same_srv_rate       494020 non-null  float64
 34  dst_host_diff_srv_rate       494020 non-null  float64
 35  dst_host_same_src_port_rate  494020 non-null  float64
 36  dst_host_srv_diff_host_rate  494020 non-null  float64
 37  dst_host_serror_rate         494020 non-null  float64
 38  dst_host_srv_serror_rate     494020 non-null  float64
 39  dst_host_rerror_rate         494020 non-null  float64
 40  dst_host_srv_rerror_rate     494020 non-null  float64
 41  label                        494020 non-null  object 
dtypes: float64(15), int64(23), object(4)
memory usage: 158.3+ MB
```

Encoding string

In [6]:
```
df['label'].unique()
```

Out[6]:
```
array(['normal', 'buffer_overflow', 'loadmodule', 'perl', 'neptune',
       'smurf', 'guess_passwd', 'pod', 'teardrop', 'portsweep', 'ipsweep',
       'land', 'ftp_write', 'back', 'imap', 'satan', 'phf', 'nmap',
       'multihop', 'warezmaster', 'warezclient', 'spy', 'rootkit'],
      dtype=object)
```

In [7]:
```
df['label']=df['label'].replace(['ipsweep','nmap','portsweep'],1)
df['label']=df['label'].replace(['normal','buffer_overflow','loadmodule','perl','neptune','smurf','guess_passwd','po
```

In [8]:
```
df
```

Out[8]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_h
0	0	tcp	http	SF	181	5450	0	0	0	0	...	9	1.0	
1	0	tcp	http	SF	239	486	0	0	0	0	...	19	1.0	
2	0	tcp	http	SF	235	1337	0	0	0	0	...	29	1.0	
3	0	tcp	http	SF	219	1337	0	0	0	0	...	39	1.0	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	49	1.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
494015	0	tcp	http	SF	310	1881	0	0	0	0	...	255	1.0	
494016	0	tcp	http	SF	282	2286	0	0	0	0	...	255	1.0	
494017	0	tcp	http	SF	203	1200	0	0	0	0	...	255	1.0	
494018	0	tcp	http	SF	291	1200	0	0	0	0	...	255	1.0	
494019	0	tcp	http	SF	219	1234	0	0	0	0	...	255	1.0	

494020 rows × 42 columns

In [9]:
```
dummies1=pd.get_dummies(df.protocol_type)
dummies1.head()
```

Out[9]:

	icmp	tcp	udp
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0

In [10]:
```
dummies2=pd.get_dummies(df.service)
dummies2.head()
```

Out[10]:

	IRC	X11	Z39_50	auth	bgp	courier	csnet_ns	ctf	daytime	discard	...	telnet	tftp_u	tim_i	time	urh_i	urp_i	uucp	uucp_path	vmnet	whois
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 66 columns

In [11]:
```
dummies3=pd.get_dummies(df.flag)
dummies3.head()
```

Out[11]:

	OTH	REJ	RSTO	RSTOS0	RSTR	S0	S1	S2	S3	SF	SH
0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	1	0

In [12]:
```
df1=pd.concat([df,dummies1,dummies2,dummies3],axis='columns')
df1.drop(['protocol_type','service','flag'], axis='columns',inplace= True)
df1.head(5)
```

Out[12]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	lnum_compromised	...	REJ	RSTO	RSTOS0	RSTR	S0
0	0	181	5450	0	0	0	0	0	1	0	...	0	0	0	0	0
1	0	239	486	0	0	0	0	0	1	0	...	0	0	0	0	0
2	0	235	1337	0	0	0	0	0	1	0	...	0	0	0	0	0
3	0	219	1337	0	0	0	0	0	1	0	...	0	0	0	0	0
4	0	217	2032	0	0	0	0	0	1	0	...	0	0	0	0	0

5 rows × 119 columns

In [13]:
```
df1.isnull().sum()
```

Out[13]:
```
duration          0
src_bytes         0
dst_bytes         0
land              0
wrong_fragment    0
                 ..
S1                0
S2                0
S3                0
SF                0
SH                0
Length: 119, dtype: int64
```

In [14]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [15]:
```
numeric_cols = df1.select_dtypes(include=np.number).columns.tolist()
categorical_cols = df1.select_dtypes('object').columns.tolist()
categorical_cols
```

Out[15]:
```
[]
```

In [16]:
```
scaler = MinMaxScaler()
```

In [17]:
```
scaler.fit(df1[numeric_cols])
```

Out[17]:
```
MinMaxScaler()
```

```
In [18]: df1[numeric_cols] = scaler.transform(df1[numeric_cols])
         df1[numeric_cols]
```

Out[18]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	lnum_compromised	...	REJ	RSTO	RSTOS0	RSTF
0	0.0	2.610418e-07	0.001057	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
1	0.0	3.446905e-07	0.000094	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
2	0.0	3.389216e-07	0.000259	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
3	0.0	3.158461e-07	0.000259	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
4	0.0	3.129617e-07	0.000394	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
494015	0.0	4.470881e-07	0.000365	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
494016	0.0	4.067060e-07	0.000443	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
494017	0.0	2.927706e-07	0.000233	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
494018	0.0	4.196859e-07	0.000233	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(
494019	0.0	3.158461e-07	0.000239	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.(

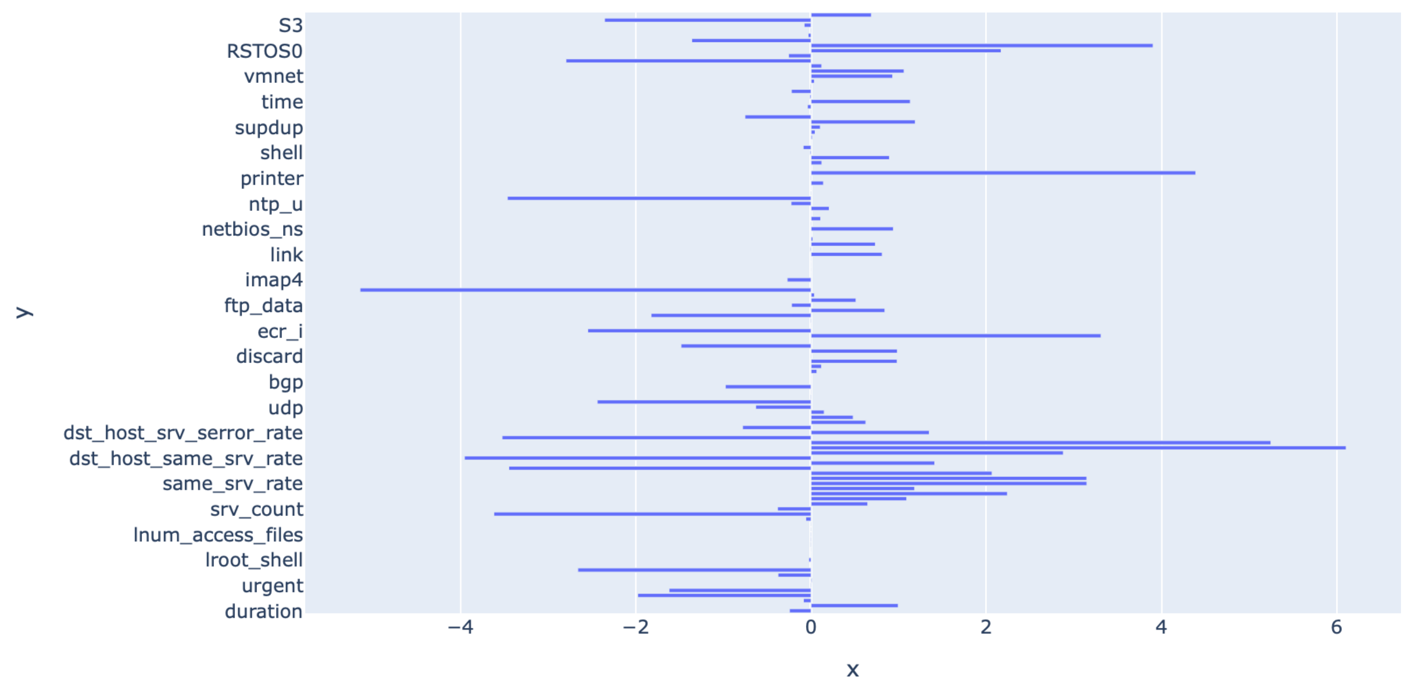
494020 rows × 119 columns

Splitting into training and testing data

```
In [19]: target=df1[numeric_cols].label
         df2=df1[numeric_cols].drop('label',axis='columns')

In [20]: from sklearn.model_selection import train_test_split
         X_train,X_test,y_train, y_test=train_test_split(df2,target,test_size=.2)
```

Fitting data

```
In [21]: from sklearn.linear_model import LogisticRegression

In [22]: model = LogisticRegression()

In [23]: model.fit(X_train,y_train)

Out[23]: LogisticRegression()

In [24]: print(model.n_iter_[0])

         94

In [25]: model.score(X_train,y_train)

Out[25]: 0.9997874579976519

In [26]: model.score(X_test,y_test)

Out[26]: 0.9997672159021902

In [27]: coefficients = model.coef_[0]

In [28]: scaled_weights=scaler.fit_transform(coefficients.reshape(118,1))

In [29]: fig = px.bar(
             x=coefficients,
             y=X_train.columns,
             orientation='h'
         )
         fig.show()
```



```
In [30]: model2_1 = LogisticRegression(solver='liblinear',penalty='l2',max_iter=12,C=0.01)

In [31]: model2_2 = LogisticRegression(solver='liblinear',penalty='l2',max_iter=20,C=10)

In [32]: model2_1.fit(X_train,y_train)

Out[32]: LogisticRegression(C=0.01, max_iter=12, solver='liblinear')

In [33]: model2_2.fit(X_train,y_train)

Out[33]: LogisticRegression(C=10, max_iter=20, solver='liblinear')
```

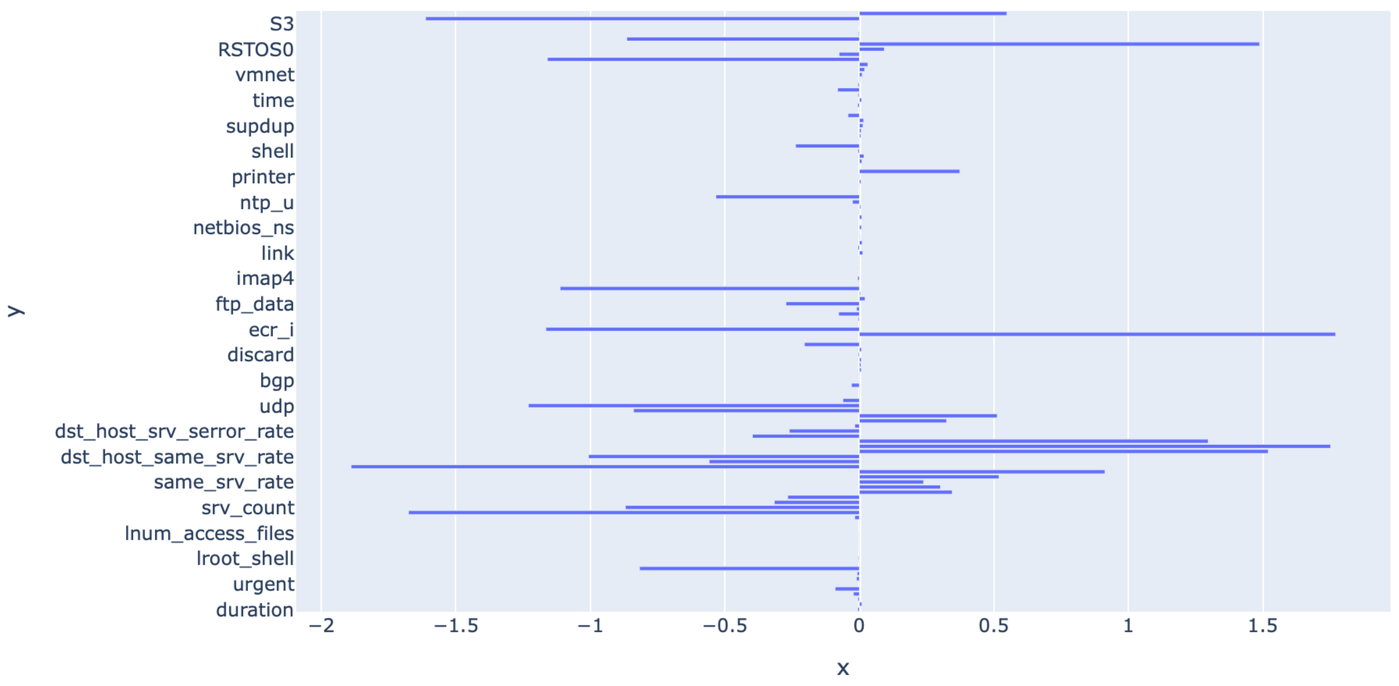
Training accuracy

```
In [34]: model2_1.score(X_train,y_train)

Out[34]: 0.998995486012712

In [35]: model2_2.score(X_train,y_train)

Out[35]: 0.9997823974737865
```

Testing accuracy

```
In [36]: model2_1.score(X_test,y_test)

Out[36]: 0.9987146269381807

In [37]: model2_2.score(X_test,y_test)

Out[37]: 0.9997570948544593
```

```
In [38]: coefficients = model2_1.coef_[0]

In [39]: scaled_weights=scaler.fit_transform(coefficients.reshape(118,1))

In [40]: fig = px.bar(
             x=coefficients,
             y=X_train.columns,
             orientation='h'
         )
         fig.show()
```


S3
RSTOS0
vmnet
time
supdup
shell
printer
ntp_u
netbios_ns
link
imap4
ftp_data
ecr_i
discard
bgp
udp
dst_host_srv_serror_rate
dst_host_same_srv_rate
same_srv_rate
srv_count
lnum_access_files
lroot_shell
urgent
duration
y
−2
−1.5
−1
−0.5
0
0.5
1
1.5
x