University of Dayton

CPS 592: Machine Learning in Cybersecurity– Spring 2023

Title: Detecting Phishing URLs

Bonus Assignment

Date: 24/Apr/2023
To: Professor. Zhongmei Yao

Students:
Student ID: **101744371** || Name: Kevin Richard

1. Introduction

This is a report to compare the LSTM model with SVM, Logistic Regression and Deep Neural Network models to distinguish Phishing URLs. With TensorFlow, data in a dataset containing only information about good and malicious emails is pre-processed to extract features that can be used by the models to learn. Finally, a comprehensive comparison is rendered using the three models.

a. Long Short-Term Memory

It is a type of recurrent neural network (RNN) that is capable of handling long-term dependencies in sequence data. It overcomes the vanishing gradient problem as it is designed to retain information over a longer period using memory cells and gates to selectively input, output, and forget information.

b. Logistic Regression

Logistic Regression is a simple and fast algorithm that is well suited for linearly separable data and is often used for binary classification problems.

c. Support Vector Model

SVM is usually used for complex and non-linearly data. It achieves good accuracy by transforming the data into higher dimensional spaces where the classes are more separable.

d. Deep Neural Networks

Neural Networks consists of several underlying layers that are interconnected to each other. Neural Networks can find any kind of relation/function regardless of its complexity, provided it is deep/optimized enough, that is how much potential it has.

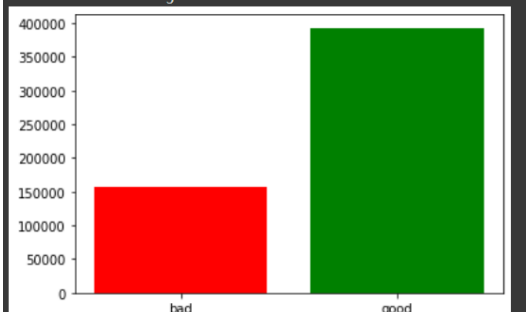
1.1 Input dataset:

```
df= pd.read_csv("phishing_site_urls.csv")
print(df.shape)
df.head()
```

```
(549346, 2)
```

	URL	Label
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad
3	mail.printakid.com/www.online.americanexpress....	bad
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad

```
['bad' 'good']
[156422 392924]
<BarContainer object of 2 artists>
```



1.2 Data pre-processing: Check for null Characters:

```
# Check for null values in the dataset

if df.isnull().any().any():
    print("There are null values in the dataset")
else:
    print("There are no null values in the dataset")
```

```
There are no null values in the dataset
```

2. Task 1: LSTM to implement RNN for detecting phishing URLs

```
n_timesteps = 10
# pad the sequences to have a fixed length
padded_sequences = pad_sequences(url_sequences, maxlen=n_timesteps, padding='post', truncating='post')

# convert the labels to a one-hot encoded array
label_map = {label: index for index, label in enumerate(set(y_label))}
num_classes = len(label_map)
label_sequences = [label_map[label] for label in y_label]
one_hot_labels = tf.keras.utils.to_categorical(label_sequences, num_classes)


# y_LSTM=df.Label
# y_LSTM=np.where(y_LSTM=='bad',0,1)


LSTM_x_train,LSTM_x_test,LSTM_y_train,LSTM_y_test =  train_test_split(padded_sequences,one_hot_labels,random_st

train_size, num_features = x_train.shape
test_size, test_features = x_test.shape
```

```
# Define the LSTM model architecture


# create the LSTM model
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=32, input_length=n_timesteps))
model.add(tf.keras.layers.LSTM(128, dropout=0.2))
# model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
model.add(tf.keras.layers.Dense(num_classes, activation='sigmoid'))


model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# train the model
model.fit(LSTM_x_train, LSTM_y_train, epochs=10, batch_size=32)

# Evaluate the model
loss, accuracy = model.evaluate(LSTM_x_test, LSTM_y_test)

# Make predictions
predictions = model.predict(LSTM_x_test)
```

3. Task 2: Report of Findings

When the URL data was tokenized directly into char level, embedded, and fit into the data, the accuracy was only around 86%. Tokenizing the URLs to words first and then lemmatizing before tokenizing into characters increased the accuracy to approx 99.87% during training phase.

a.1) One hot encoding

The phishing URL Data is first tokenized and lemmatized.

Tokenizing: It is the process of breaking down a sentence or text document into individual words or tokens.

Word Lemmatizing: It is the process of reducing a word to its base or root form.

```
# Tokenizing and Lemmatizing the URL

tok= RegexpTokenizer(r'[A-Za-z0-9]+')
df['list_url']=df.clean_url.map(lambda x: tok.tokenize(x))

wnl = WordNetLemmatizer()
df['lem_url'] = df['list_url'].map(lambda x: [wnl.lemmatize(word) for word in x])
```

```
[140] df.head()
```

	URL	Label	clean_url	list_url	lem_url
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad	nobell.it/70ffb52d079109dca5664cce6f317373782/...	[nobell, it, 70ffb52d079109dca5664cce6f3173737...	[nobell, it, 70ffb52d079109dca5664cce6f3173737...
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad	serviciosbys.com/paypal.cgi.bin.get-into.herf....	[serviciosbys, com, paypal, cgi, bin, get, int...	[serviciosbys, com, paypal, cgi, bin, get, int...
3	mail.printakid.com/www.online.americanexpress....	bad	mail.printakid.com/www.online.americanexpress....	[mail, printakid, com, www, online, americanex...	[mail, printakid, com, www, online, americanex...
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad	thewhiskeydregs.com/wp-content/themes/widescre...	[thewhiskeydregs, com, wp, content, themes, wi...	[thewhiskeydregs, com, wp, content, theme, wid...

One hot labels:

The Lemmatized data is further tokenized into character level and the labels are mapped categorically.

```
x_LSTM=df['lem_url'].tolist()
y_label = y.tolist()

# create a Tokenizer object to encode the URLs as sequences of integers
tokenizer = Tokenizer(char_level=True)
tokenizer.fit_on_texts(x_LSTM)
url_sequences = tokenizer.texts_to_sequences(x_LSTM)
```

```
n_timesteps = 10
# pad the sequences to have a fixed length
padded_sequences = pad_sequences(url_sequences, maxlen=n_timesteps, padding='post', truncating='post')

# convert the labels to a one-hot encoded array
label_map = {label: index for index, label in enumerate(set(y_label))}
num_classes = len(label_map)
label_sequences = [label_map[label] for label in y_label]
one_hot_labels = tf.keras.utils.to_categorical(label_sequences, num_classes)
```

a.2) Embedding step

```
model.add(tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=32, input_length=n_timesteps))
```

a.3) LSTM structure

It has one Embedding layer, one LSTM layer and one Sigmoid layer.

```
# create the LSTM model
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=32, input_length=n_timesteps))
model.add(tf.keras.layers.LSTM(128, dropout=0.2))
# model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
model.add(tf.keras.layers.Dense(num_classes, activation='sigmoid'))
```

b.1) Sample runs

A single epoch took approx. 1 hour and 10 minutes. At the third epoch the training accuracy reached 99.87% accuracy. As of now this is being considered as the final accuracy. The accuracy might increase when all 10 epochs are executed completely. Please note that the achieved accuracy is at-least 1% greater than the research paper, which had 98.76% accuracy as shown in below table.

TABLE III. RESULTS LSTM NETWORK

Fold	AUC	Accuracy	Recall	Precision	F1-score
0	0.999044	0.9871	0.991114	0.983203	0.987143
1	0.999106	0.987921	0.989549	0.986359	0.987952
2	0.999141	0.987844	0.98716	0.988506	0.987833
Average	**0.999097**	**0.987622**	**0.989274**	**0.986023**	**0.987642**
Std dev	**4e-05**	**0.00037**	**0.001626**	**0.002178**	**0.000357**

```
# Define the LSTM model architecture


# create the LSTM model
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(input_dim=len(tokenizer.word_index)+1, output_dim=32, input_length=n_timesteps))
model.add(tf.keras.layers.LSTM(128, dropout=0.2))
# model.add(tf.keras.layers.Dense(num_classes, activation='softmax'))
model.add(tf.keras.layers.Dense(num_classes, activation='sigmoid'))


model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# train the model
model.fit(LSTM_x_train, LSTM_y_train, epochs=10, batch_size=32)

# Evaluate the model
loss, accuracy = model.evaluate(LSTM_x_test, LSTM_y_test)

# Make predictions
predictions = model.predict(LSTM_x_test)
```

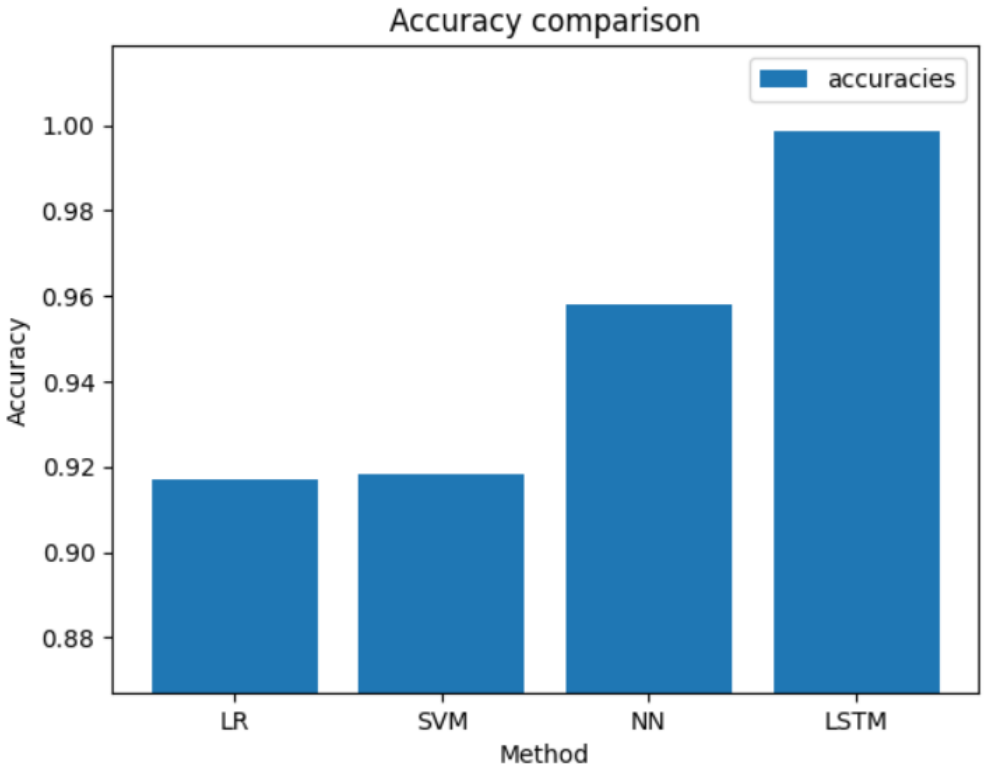
```
Epoch 1/10
12680/12680 [==============================] - 4361s 344ms/step - loss: 0.0949 - accuracy: 0.9649
Epoch 2/10
12680/12680 [==============================] - 4353s 343ms/step - loss: 0.0134 - accuracy: 0.9958
Epoch 3/10
 1749/12680 [===>..........................] - ETA: 1:02:30 - loss: 0.0043 - accuracy: 0.9987
```

b.2) Accuracy Comparison

LSTM Training Accuracy

```
Epoch 1/10
12680/12680 [==============================] - 4361s 344ms/step - loss: 0.0949 - accuracy: 0.9649
Epoch 2/10
12680/12680 [==============================] - 4353s 343ms/step - loss: 0.0134 - accuracy: 0.9958
Epoch 3/10
 1862/12680 [===>..........................] - ETA: 1:01:52 - loss: 0.0042 - accuracy: 0.9987
```

The accuracy is compared with the Logistic Regression, SVM and DNN models from the previous assignments



c.1) Challenges and difficulties encountered

The main challenge of this assignment was to figure out the one hot encoding and the embedding setup. The long training time remained an obstacle until the end. Much faster training time could be obtained through a stronger machine. An hour of training time for a single epoch shows a glimpse into the scope of equipment and resources needed to analyse huge and complex data sets.