ЛЕКЦИЯ 11

Изучение выявления Фишинга нейронными сетями

ЭТАПЫ ВЫЯВЛЕНИЯ ФИШИНГА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

Сбор данных
Источники данных:
•**PhishTank, OpenPhish, APWG** – базы фишинговых сайтов
•**VirusTotal, URLhaus** – списки вредоносных URL
•**Лог-файлы прокси-серверов и веб-браузеров**
•**DNS-запросы и сетевой трафик**
Признаки фишинговых сайтов:
•**Длина URL** (фишинговые сайты часто используют длинные ссылки)
•**Использование IP-адресов в URL**
•**Подозрительные домены** (.tk, .ml, .ga, .cf, .gq)
•**Подменные символы в домене** (например, g00gle.com, paypa1.com)
•**HTTPS vs. HTTP** (многие фишинговые сайты не используют HTTPS)
•**Перенаправления (редиректы)**
•**Избыточное использование вложенных <iframe>**
•**Частота встречаемости домена в WHOIS**

ЭТАПЫ ВЫЯВЛЕНИЯ ФИШИНГА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

```
phishing
```

	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
...	...	...
539341	paulasalamanca.com/g7cberv	bad
539342	trustcarts.com/g7cberv	bad
539343	rndled.com/g7cberv	bad
539344	rktest.net/g7cberv	bad
539345	unoldontal.com/g7cberv	bad

539346 rows × 2 columns

ЭТАПЫ ВЫЯВЛЕНИЯ ФИШИНГА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

Предобработка данных

Перед применением алгоритмов машинного обучения данные должны быть очищены и преобразованы.

Действия:

- **Очистка данных** (удаление дубликатов, исправление ошибок)
- **Токенизация URL** (разделение домена, поддоменов, пути)
- **One-Hot Encoding** (преобразование категориальных признаков)
- **Нормализация числовых признаков** (например, длина URL)
- **Преобразование данных с помощью TF-IDF**

ЭТАПЫ ВЫЯВЛЕНИЯ ФИШИНГА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

```
vectorizer = TfidfVectorizer(max_features = 100)
```

```
X = vectorizer.fit_transform(X.values)
```

```
X
```

```
<539346x100 sparse matrix of type '<class 'numpy.float64'>'
        with 1344974 stored elements in Compressed Sparse Row format>
```

```
X.shape
```

```
(539346, 100)
```

```
ros = RandomOverSampler(random_state=42)
```

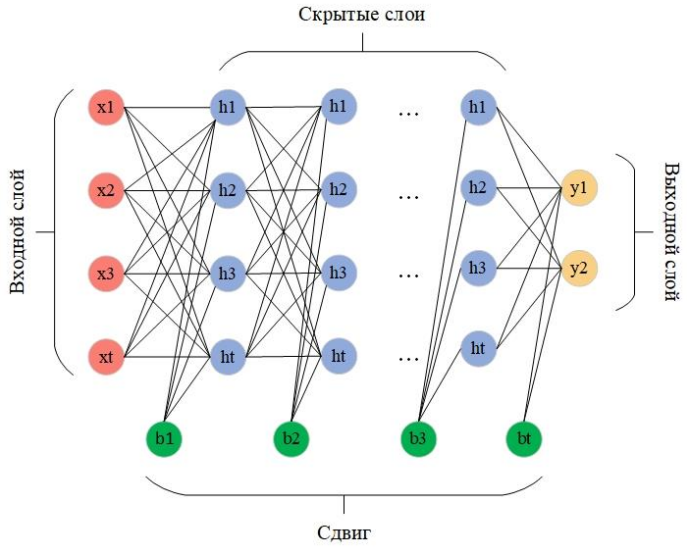
```
X_resampled, y_resampled = ros.fit_resample(X, y)
```

ЭТАПЫ ВЫЯВЛЕНИЯ ФИШИНГА С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

- **Разделение данных на обучающую и тестовую выборки**
- Данные разделяются следующим образом:
- **Обучающая выборка (Train Set)**: 70-80% данных
- **Тестовая выборка (Test Set)**: 20-30% данных
- Дополнительно можно использовать **кросс-валидацию (k-fold cross-validation)** для более точной оценки моделей.

ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ

- **Глубокие нейронные сети (DNN)** представляют собой модель нейронных сетей с двумя и более скрытыми слоями. Нейронная сеть состоит из входного слоя, содержащего входные данные, скрытых слоев, включающих узлы, называемые нейронами, и выходного слоя, содержащего один или несколько нейронов



ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ

- При этом $x = x_1, x_2, ..., x_f$ является входным вектором, $w_1, w_2, ..., w_i$ веса соединения каждого уровня,
- $b_1, b_2, ..., b_i$ - вектор смещения. Уровни от l_2 до l_{n-1} образуют скрытые слои, $y_1, y_2, ..., y_m$ является выходным вектором.
- Элементы скрытых и выходных слоев называются нейронами. Они представлены функциями активации, отвечающими за нелинейное функциональное отображение между входными данными и переменной отклика. Самыми популярными функциями активации являются сигмоидная функция, функция гиперболического тангенса (*tanh*), выпрямленная линейная единица (*ReLu*) и *softmax*. Сигмоидная функция в основном используется на выходном слое в бинарной классификации, так как определяет выходное значение как 0 или 1. Функция *tanh* – это улучшенная версия сигмоидной функции с разницей лишь в том, что в *tanh* выходные значения находятся в диапазоне от -1 до 1. В скрытых слоях чаще всего применяется функция активации *ReLu*. Это приводит к выходному значению 0, если он получает отрицательный вход *x*, иначе для положительных входов он возвращает *x* без изменений, подобно линейной функции.
- Функция *softmax* применяется в многоклассовой классификации, вычисляя вероятность того, что каждое вхождение принадлежит к заранее определенному классу, и корректирует выходные значения для каждого класса так, чтобы они находились в диапазоне от 0 до 1. Функция *softmax* обычно используется только для выходного слоя.

ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ

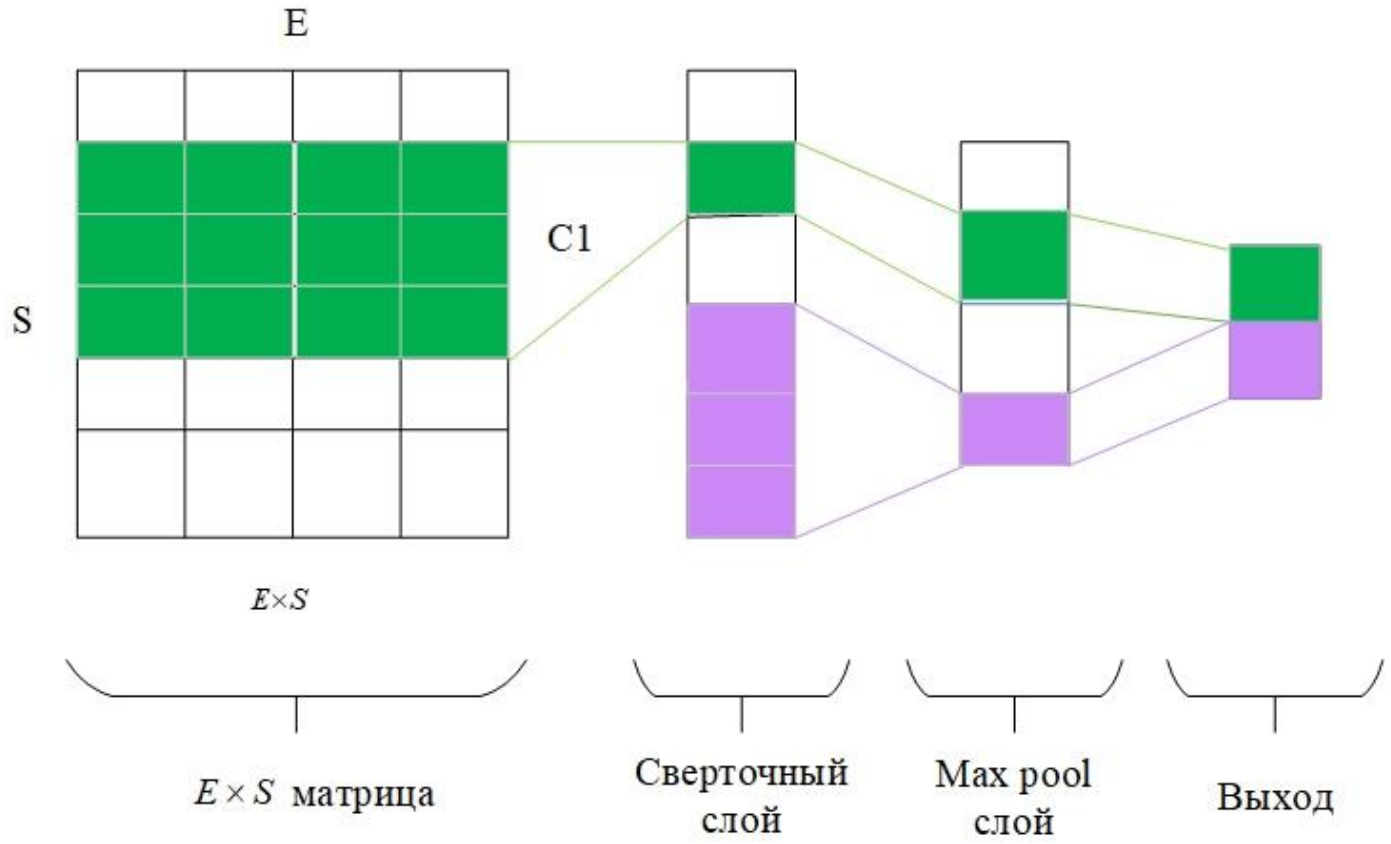
```
with strategy.scope():
    model_dnn = Sequential()
    model_dnn.add(Dense(32, input_dim=20))
    model_dnn.add(Activation('tanh'))
    model_dnn.add(Dropout(0.4))
    model_dnn.add(Dense(16))
    model_dnn.add(Activation('tanh'))
    model_dnn.add(Dense(1))
    model_dnn.add(Activation('sigmoid'))
    optimizer = Adam(learning_rate=0.000008)
    model_dnn.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy', Precision(), Recall(),
                                                                                 tfa.metrics.FBetaScore(num_classes=2,average="micro"
    model_dnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 32)                672

 activation (Activation)     (None, 32)                0

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 16)                528

 activation_1 (Activation)   (None, 16)                0

 dense_2 (Dense)             (None, 1)                 17

 activation_2 (Activation)   (None, 1)                 0

=================================================================
Total params: 1,217
Trainable params: 1,217
Non-trainable params: 0
_________________________________________________________________
```

СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

Сверточные нейронные сети (CNN) – один из популярных и часто используемых видов нейронных сетей, приобрётший большую популярность благодаря использованию в задачах классификации и распознавания изображений. Они применяются при работе с изображениями, в которых фильтр перемещается по самому изображению. Также сверточные нейронные сети получили распространение и в задачах по распознаванию речи, обработке естественных языков и анализу тональности. При работе с текстовыми данными необходимо учитывать, что слова имеют разную длину, и в векторном представлении их необходимо привести к одинаковой размерности. Для векторного преобразования обычно используются такие вхождения слов, как Word2vec, Glove и FastText.

СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ



СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

```
with strategy.scope():

    model_cnn = Sequential()
    model_cnn.add(Conv1D(filters=nb_filter, kernel_size=filter_length, activation='relu', input_shape=(20,1)))
    model_cnn.add(GlobalMaxPooling1D())
    model_cnn.add(Dense(hidden_dims))
    model_cnn.add(Dropout(0.4))
    model_cnn.add(Activation('relu'))
    model_cnn.add(Flatten())
    model_cnn.add(Dense(32, activation='relu'))
    model_cnn.add(Dropout(0.4))
    model_cnn.add(Dense(16, activation='relu'))
    model_cnn.add(Dropout(0.4))
    model_cnn.add(Dense(1, activation='sigmoid'))
    optimizer = Adam(learning_rate=0.000008)
    model_cnn.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy', Precision(), Recall(),
                                                                                 tfa.metrics.FBetaScore(num_classes=2,average="micro"
    model_cnn.summary()
```

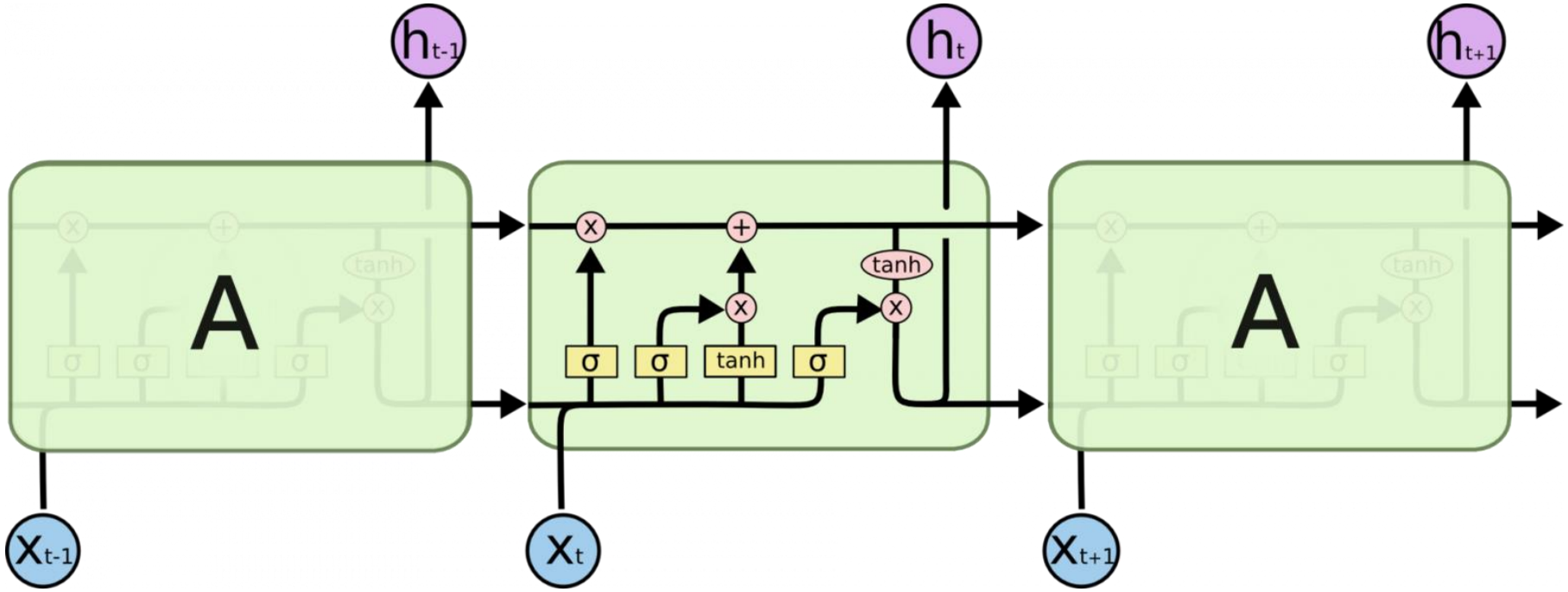
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 18, 250)           1000

 global_max_pooling1d (Globa  (None, 250)              0
 lMaxPooling1D)

 dense (Dense)               (None, 250)               62750

 dropout (Dropout)           (None, 250)               0

 activation (Activation)     (None, 250)               0

 flatten (Flatten)           (None, 250)               0

 dense_1 (Dense)             (None, 32)                8032

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 16)                528

 dropout_2 (Dropout)         (None, 16)                0

 dense_3 (Dense)             (None, 1)                 17

=================================================================
Total params: 72,327
Trainable params: 72,327
Non-trainable params: 0
```

LONG SHORT-TERM MEMORY - LSTM

- Долгая краткосрочная память (Long short-term memory - LSTM) – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Они были представлены Зеппом Хохрайтер и Юргеном Шмидхубером в 1997 году, а затем усовершенствованы и популярно изложены в работах многих других исследователей. Они прекрасно решают целый ряд разнообразных задач и в настоящее время широко используются.
- LSTM разработаны специально, чтобы избежать проблемы долговременной зависимости. Запоминание информации на долгие периоды времени – это их обычное поведение, а не что-то, чему они с трудом пытаются обучиться.
- Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети. В обычной RNN структура одного такого модуля очень проста, например, он может представлять собой один слой с функцией активации tanh (гиперболический тангенс).

LONG SHORT-TERM MEMORY - LSTM

- Структура LSTM также напоминает цепочку, но модули выглядят иначе. Вместо одного слоя нейронной сети они содержат целых четыре, и эти слои взаимодействуют особенным образом



LONG SHORT-TERM MEMORY - LSTM

```
with strategy.scope():

    model_lstm = Sequential()
    model_lstm.add(LSTM(64, input_shape=(20,1), return_sequences = True))
    model_lstm.add(SpatialDropout1D(0.25))
    model_lstm.add(LSTM(32, dropout=0.5, recurrent_dropout=0.5))
    model_lstm.add(Dropout(0.2))
    model_lstm.add(Dense(1, activation='sigmoid'))
    optimizer = Adam(learning_rate=0.000008)
    model_lstm.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy', Precision(), Recall(),
                                                                                  tfa.metrics.FBetaScore(num_classes=2,average="micro'
    model_lstm.summary()
```

```
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel
as fallback when running on GPU.
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 20, 64)            16896

 spatial_dropout1d_1 (Spatia  (None, 20, 64)           0
 lDropout1D)

 lstm_3 (LSTM)               (None, 32)                12416

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 29,345
Trainable params: 29,345
Non-trainable params: 0
_________________________________________________________________
```

СПАСИБО ЗА ВНИМАНИЕ!!!