ЛЕКЦИЯ 6

Изучение выявления Man in the Middle атак моделями машинного обучения

КЛАССИФИКАЦИЯ

Алгоритмы машинного обучения:

- Дерево решений (Decision tree);
- Случайный лес (Random forest);
- XGBoost;
- CatBoost;
- AdaBoost

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Сбор данных

Для эффективного выявления атак MitM необходимо собрать сетевые данные, содержащие как **обычный трафик**, так и **трафик, характерный для MitM-атак**.

Источники данных:

- **Лог-файлы сетевых устройств** (роутеры, коммутаторы, серверы)
- **Пакеты трафика** (Wireshark, Tcpdump)
- **NetFlow и sFlow статистика**

Открытые датасеты (CICIDS2017, NSL-KDD, UNSW-NB15)

Ключевые признаки аномального поведения:

- **Несоответствие MAC-адресов и IP-адресов** (отравление ARP)
- **Высокая задержка в ответах** (proxy-перехват)
- **Изменения в SSL-сертификатах** (атакующий использует свой сертификат)
- **Необычные DNS-запросы** (перенаправление трафика)
- **Резкое увеличение трафика** на нестандартных портах

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
MitM
```

	0	1	2	3	4	5	6	7	8	9	...	10
0	1.000000	1294.000000	0.000000e+00	1.000000	1294.000000	0.000000e+00	1.000000	1294.000000	0.000000e+00	1.000000	...	0.000000e+0
1	1.000000	1514.000000	0.000000e+00	1.000000	1514.000000	0.000000e+00	1.000000	1514.000000	0.000000e+00	1.000000	...	0.000000e+0
2	1.999505	1294.000000	6.984919e-10	1.999703	1294.000000	2.328306e-10	1.999901	1294.000000	6.984919e-10	1.999990	...	0.000000e+0
3	2.998985	1294.000000	9.313226e-10	2.999391	1294.000000	4.656613e-10	2.999797	1294.000000	6.984919e-10	2.999980	...	6.984919e-1
4	3.998061	1294.000000	9.313226e-10	3.998836	1294.000000	2.328306e-10	3.999612	1294.000000	6.984919e-10	3.999961	...	2.328306e-1
...	...	...	...	...	...	...	...	...	...	...	...	
2504262	380.551133	1331.684771	1.895906e+05	650.148549	1336.357995	1.865743e+05	1973.033776	1339.428666	1.851917e+05	19664.853626	...	1.813631e+0
2504263	379.629067	1332.165017	1.891785e+05	649.176314	1336.631637	1.863354e+05	1972.036680	1339.517190	1.851133e+05	19663.862251	...	1.813090e+0
2504264	380.276647	1332.643182	1.887678e+05	649.814659	1336.904590	1.860970e+05	1972.670406	1339.605640	1.850348e+05	19664.496996	...	1.812549e+0
2504265	379.354717	1333.121248	1.883566e+05	648.842152	1337.177530	1.858584e+05	1971.672375	1339.694090	1.849564e+05	19663.504357	...	1.812009e+0
2504266	379.951197	1333.597306	1.879468e+05	649.427959	1337.449804	1.856203e+05	1972.252742	1339.782469	1.848780e+05	19664.085817	...	1.811468e+0

2504267 rows × 115 columns

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Предобработка данных

- Сетевые данные, полученные на предыдущем этапе, должны быть очищены и приведены в удобный формат для машинного обучения.

Действия:

- **Удаление дубликатов** и неинформативных записей
- **Очистка данных** (устранение отсутствующих значений, обработка выбросов)
- **Приведение категориальных данных** к числовому виду (например, с помощью One-Hot Encoding)
- **Нормализация и стандартизация** (Min-Max Scaling, Standard Scaling)

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
scaler.fit(MitM)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Feature names only support names th
at are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

```
MinMaxScaler()
```

```
MitM_scaled = pd.DataFrame(scaler.transform(MitM))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:1688: FutureWarning: Feature names only support names th
at are all strings. Got feature names with dtypes: ['int', 'str']. An error will be raised in 1.2.
  warnings.warn(
```

```
with open('MitM_scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```

```
MitM_scaled
```

	0	1	2	3	4	5	6	7	8	9	...	106	107	108	109	1
0	0.680594	0.811159	0.884536	0.368931	0.813469	0.880903	0.372045	0.586947	0.878381	0.371202	...	0.426234	0.954144	0.846794	0.904155	0.8812
1	0.618364	0.593748	0.874648	0.369892	0.710273	0.877792	0.364657	0.848117	0.878255	0.368929	...	0.429657	0.954144	0.846794	0.902173	0.8811
2	0.149055	0.761982	0.878075	0.376165	0.840070	0.879933	0.367722	0.938136	0.880932	0.362809	...	0.426239	0.954144	0.846794	0.647965	0.8807
3	0.045775	0.648012	0.876872	0.372446	0.761011	0.876963	0.374175	0.911456	0.877625	0.375386	...	0.425237	0.954144	0.846794	0.290110	0.8815
4	0.706189	0.466479	0.860038	0.261018	0.448934	0.855961	0.270552	0.442087	0.851410	0.281942	...	0.424160	0.954144	0.846794	0.909432	0.8773
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2504262	0.023012	0.580222	0.857112	0.428156	0.694649	0.864136	0.409059	0.872832	0.873982	0.383464	...	0.426494	0.954144	0.846794	0.159904	0.8814
2504263	0.259131	0.302684	0.843414	0.321658	0.349847	0.846513	0.305853	0.409280	0.848726	0.291985	...	0.430383	0.954144	0.846794	0.808057	0.8768
2504264	0.889435	0.705402	0.889849	0.331843	0.780692	0.888373	0.335660	0.910191	0.885616	0.346850	...	0.420949	0.954144	0.846794	0.906788	0.8811
2504265	0.658944	0.727583	0.881733	0.356368	0.796838	0.877444	0.372796	0.915024	0.876662	0.378450	...	0.428650	0.954144	0.846794	0.903414	0.8813
2504266	0.877150	0.651064	0.866764	0.396554	0.747148	0.870116	0.390237	0.896983	0.875249	0.379784	...	0.423274	0.954144	0.846794	0.906455	0.8811

2504267 rows × 116 columns

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Chi_square feature selection ¶

```
bestfeatures = SelectKBest(score_func=chi2, k=20)
```

```
fit_feat = bestfeatures.fit(MitM_scaled,MitM_labels)
```

```
MitM_scores = pd.DataFrame(fit_feat.scores_)
MitM_columns = pd.DataFrame(MitM_scaled.columns)
```

```
featureScores = pd.concat([MitM_columns,MitM_scores],axis=1)
featureScores.columns = ['Specs','Score']
```

```
print(featureScores.nlargest(20, 'Score'))
```

```
     Specs          Score
0        0  284570.859509
59      59   50057.824037
78      78   50057.824037
28      28   49969.049551
109    109   43181.048574
13      13   37499.951751
64      64   29401.928838
52      52    5751.688007
75      75    5751.688007
25      25    5723.137072
57      57    5114.790811
54      54    4722.101536
47      47    4672.799643
61      61    4633.207703
110    110    4563.681203
60      60    4563.208046
29      29    4553.725155
14      14    4553.498041
103    103    4543.808886
46      46    4543.692973
```

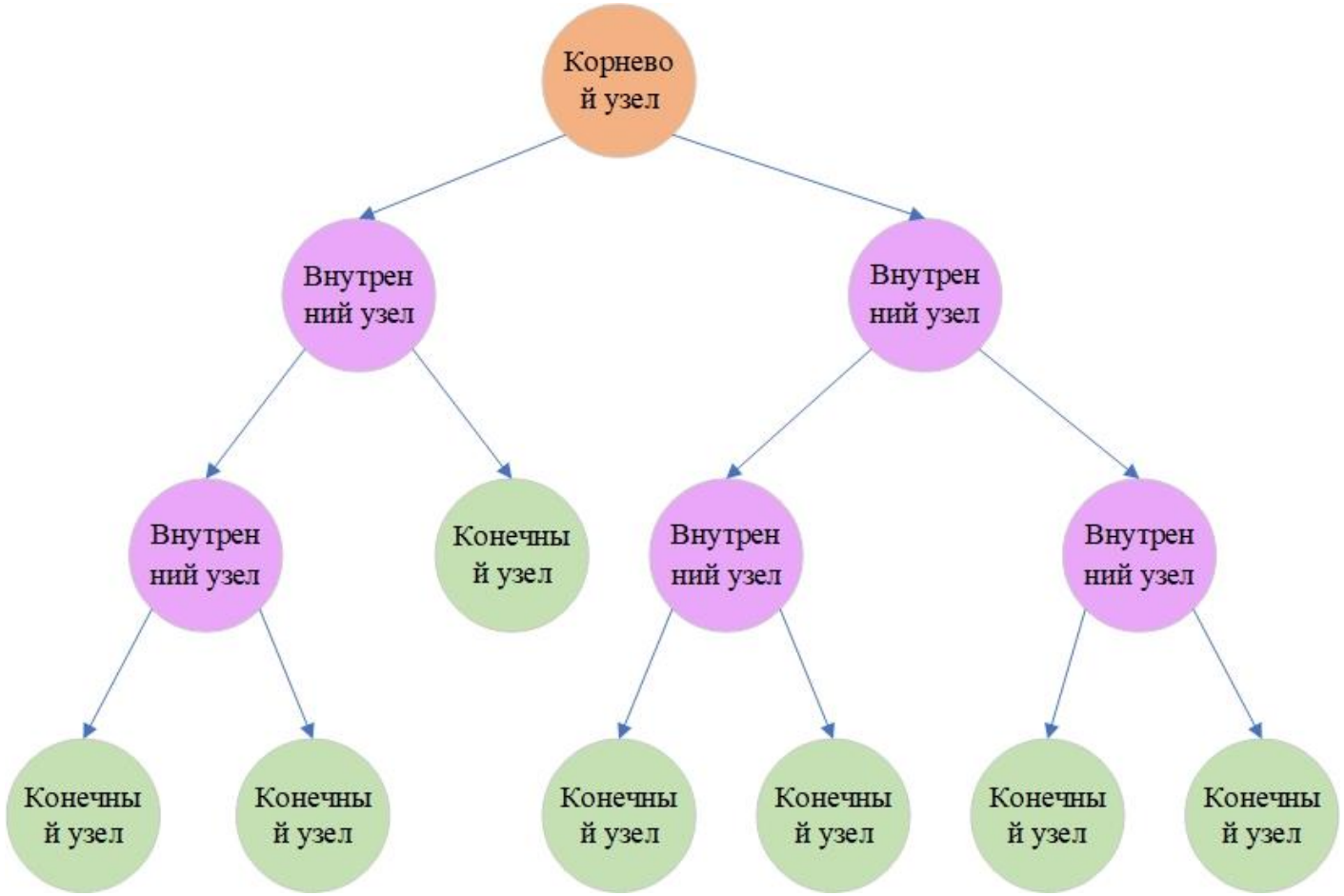
ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

- **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.

ДЕРЕВО РЕШЕНИЙ

Дерево решений– метод обучения с учителем, который использует набор правил для принятия решений подобно тому, как человек принимает решения. В данном методе данные разделяются на подмножества в зависимости от определенных признаков, отвечая на определенные вопросы до тех пор, пока все точки данных не будут принадлежать определенному классу. Таким образом, образуется древовидная структура с добавлением узла для каждого вопроса. Первый узел является корневым узлом (root node). При классификации документов на первом этапе выбирается слово, и все документы, содержащие его, помещаются в одну сторону, а документы, не содержащие его, помещаются в другую сторону. В результате образуются два датасета. После этого в этих датасетах выбирается новое слово, и все предыдущие шаги повторяются. Так продолжается до тех пор, пока весь датасет не будет разделен и присвоен конечным узлам. Если в конечном узле все точки данных однозначно соответствуют одному и тому же классу, то класс узла точно определен. В случае смешанных узлов алгоритм присваивает данному узлу класс с наибольшим числом точек данных, относящихся к нему.

ДЕРЕВО РЕШЕНИЙ

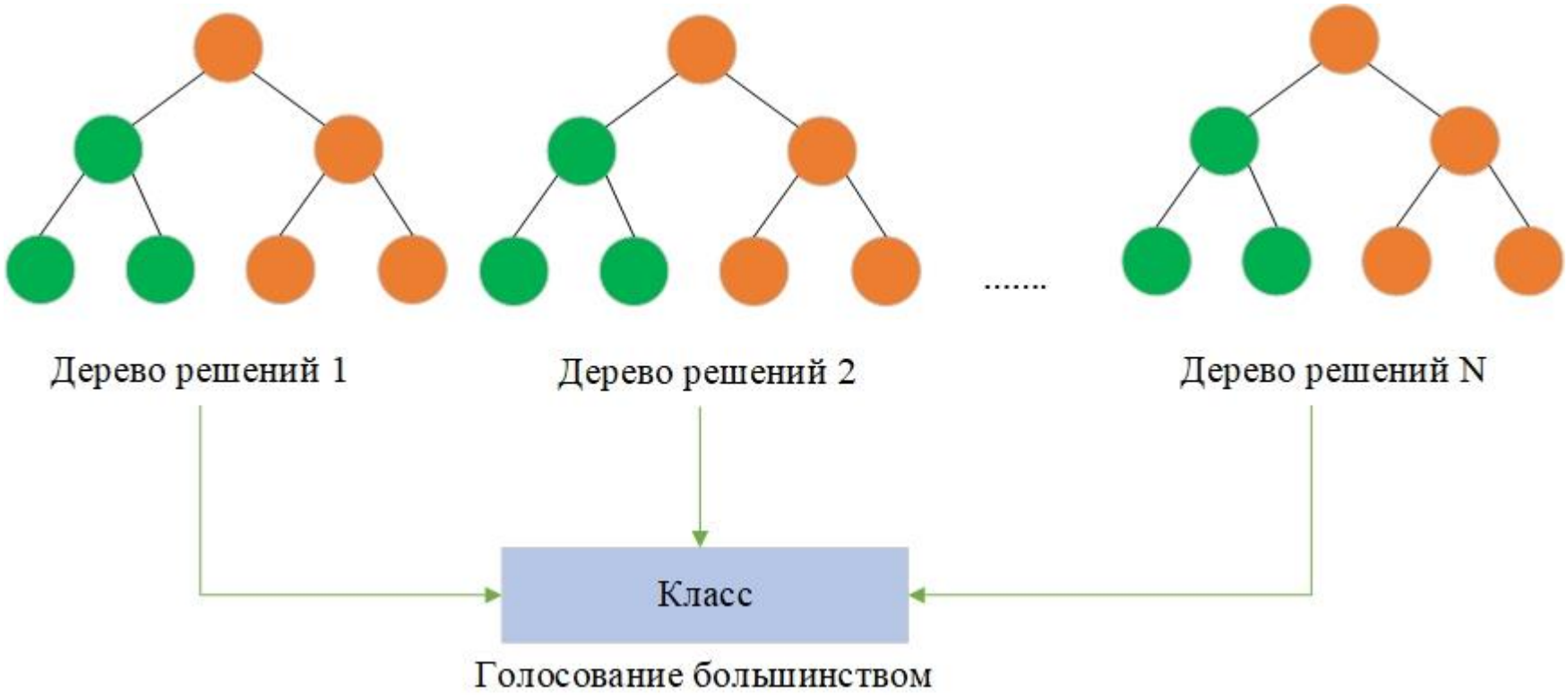


СЛУЧАЙНЫЙ ЛЕС

- Случайный лес– популярный алгоритм машинного обучения, основанный на концепции ансамблевого обучения. В данной концепции несколько классификаторов объединяются для улучшения производительности модели. Случайный лес состоит не из одного, а из множества деревьев решений. В задачах классификации каждый документ независимо кдассифицируется всеми деревьями. Класс документа определяется на основе наибольшего числа голосов среди всех деревьев.
- Алгоритм случайного леса имеет следующий ряд особенностей и преимуществ:

– Довольно быстро обучается.
– Эффективно обрабатывает датасеты с большим числом признаков.
– Выполняет предсказание данных с очень высокой точностью.
– Показывает хорошую эффективность даже при наличии большого числа пропусков данных.
– Хорошо обрабатываются как непрерывные, так и дискретные признаки.
– Обладает высокой масштабируемостью.

СЛУЧАЙНЫЙ ЛЕС



XGBOOST

- XGboost (eXtreme Gradient Boosting) – оптимизированный продвинутый алгоритм машинного обучения, использующий принцип бустинга. Он имеет хорошую производительность и решает большинство проблем регрессии и классификации. Использование ансамблевой техники подразумевает, что ошибки предыдущих шагов устраняются в новой модели. Отклонения прогнозов обученного ансамбля вычисляются на обучающем наборе на каждой итерации. Таким образом, оптимизация выполняется путем добавления новых древовидных прогнозов в ансамбль, уменьшая среднее отклонение модели. Эта процедура продолжается до тех пор, пока не будет достигнут требуемый уровень ошибки или критерий ранней остановки (максимальное количество деревьев или достижение заданной точности).

XGBoost

ЭТАПЫ ВЫЯВЛЕНИЯ MITM-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
metrics_list = []
```

```
classifiers = [MultinomialNB(), LogisticRegression(), DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None),
RandomForestClassifier(n_estimators = 10), xgb.XGBClassifier(random_state=42), CatBoostClassifier(task_type="GPU", devices='0'),
AdaBoostClassifier(n_estimators=10)]
```

```
k = 0
for i in classifiers:
    i.fit(x_train, y_train)
    y_pred = i.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred)
    roc_auc = metrics.auc(fpr, tpr)
    metrics_list.append({'Accuracy': accuracy,
                         'Precision': precision,
                         'Recall': recall,
                         'F1-score': f1,
                         'fpr': fpr,
                         'tpr': tpr})

    print("Evaluation metrics of " + algorithms[k]+" algorithm: ")
    print('Accuracy: ', accuracy)
    print('Precision: ', precision)
    print('Recall: ', recall)
    print('F1-score: ', f1)

    k = k + 1
```

СПАСИБО ЗА ВНИМАНИЕ!!!