

ЛЕКЦИЯ 8

**Изучение выявления Вредоносного ПО моделями
машинного обучения**

КЛАССИФИКАЦИЯ

Алгоритмы машинного обучения:

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

ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Сбор данных

Для обучения модели необходима большая выборка файлов и сетевых данных, содержащих как **вредоносное ПО**, так и **чистые файлы**.

Источники данных:

- **VirusTotal, MalwareBazaar, Malicia Dataset** – базы вредоносного ПО
- **Логи антивирусов и IDS-систем** (Snort, Suricata)
- **Образцы исполняемых файлов** (PE-файлы в Windows, ELF-файлы в Linux)
- **Сетевой трафик** (Wireshark, NetFlow)

Типы данных:

- **Статические признаки** (размер файла, хеши, секции PE, импортированные библиотеки)
- **Динамические признаки** (поведение в виртуальной среде, анализ API-вызовов)
- **Сетевой анализ** (аномалии в HTTP, DNS-запросах, командно-контрольные сервера)

ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

malware

	index	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedData	Address
0	117796	224	8450	9.0	0	1024	1024	0	
1	23686	224	8450	8.0	0	35328	10752	0	
2	70211	224	271	6.0	0	12288	20480	0	
3	95900	224	8450	8.0	0	167936	8192	0	
4	8183	224	8482	14.0	0	4096	15360	0	
...	...	...	...	...	...	...	...	...	...
216346	210867	224	259	9.0	0	99328	1148928	0	
216347	176116	224	258	10.0	0	119808	375808	0	
216348	99211	224	8462	8.0	0	1536	2048	0	
216349	169002	224	258	10.0	0	28672	445952	16896	
216350	213842	224	783	2.0	56	29184	14848	110592	

216351 rows × 55 columns

ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Предобработка данных

Данные необходимо очистить и подготовить к обучению.

Действия:

- **Удаление дубликатов и выбросов**
- **Приведение категориальных данных к числовому виду**
- **Нормализация признаков** (например, MinMaxScaler)
- **Feature Engineering** – создание новых признаков (например, частота API-вызовов)

ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
from sklearn.preprocessing import MinMaxScaler  
scaler = MinMaxScaler()
```

```
scaler.fit(malware_data)
```

```
MinMaxScaler()
```

```
malware_scaled = pd.DataFrame(scaler.transform(malware_data))
```

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

```
malware_data
```

	index	SizeOfOptionalHeader	Characteristics	MajorLinkerVersion	MinorLinkerVersion	SizeOfCode	SizeOfInitializedData	SizeOfUninitializedData	Address
	0	117796	224	8450	9.0	0	1024	1024	0
	1	23686	224	8450	8.0	0	35328	10752	0
	2	70211	224	271	6.0	0	12288	20480	0
	3	95900	224	8450	8.0	0	167936	8192	0
	4	8183	224	8482	14.0	0	4096	15360	0
	...	...	...	...	...	...	...	...	...
216346	210867	224	259	9.0	0	99328	1148928	0	
216347	176116	224	258	10.0	0	119808	375808	0	
216348	99211	224	8462	8.0	0	1536	2048	0	
216349	169002	224	258	10.0	0	28672	445952	16896	
216350	213842	224	783	2.0	56	29184	14848	110592	

216351 rows × 54 columns

ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Chi_square feature selection

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

```
fit_feat = bestfeatures.fit(malware_scaled, malware['legitimate'])
```

```
malware_scores = pd.DataFrame(fit_feat.scores_)  
malware_columns = pd.DataFrame(malware_scaled.columns)
```

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

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

	Specs	Score
0	0	10610.997896
24	24	10285.818917
22	22	3190.340238
2	2	2987.899315
34	34	1301.058610
48	48	1282.168947
47	47	1070.573703
18	18	813.731644
53	53	610.707103
32	32	582.777707
4	4	428.536642
33	33	389.225729
23	23	347.613734
16	16	317.885149
17	17	290.498051
44	44	258.081053
43	43	138.408542
36	36	127.392942
39	39	103.853904
45	45	101.951695

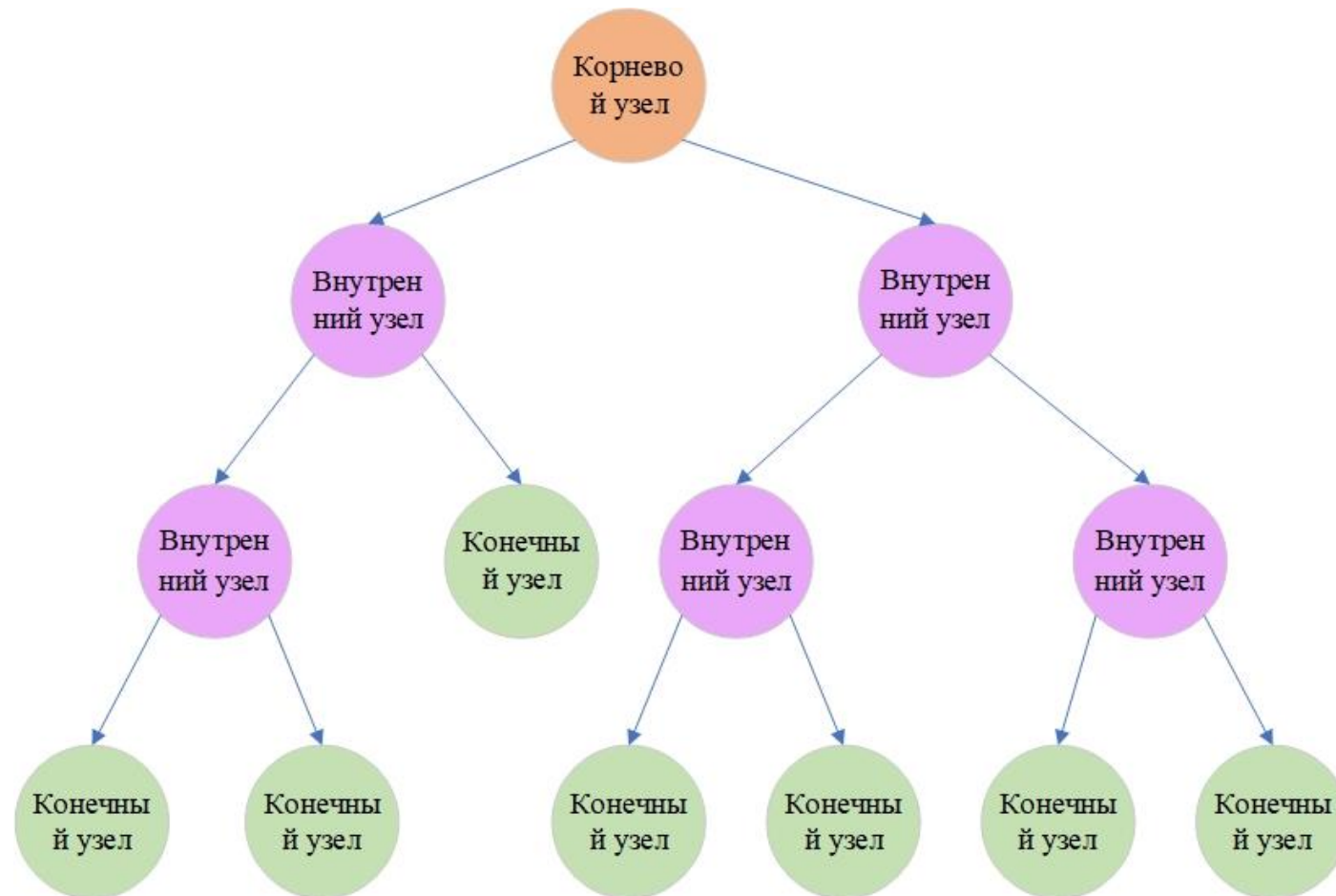
ЭТАПЫ ВЫЯВЛЕНИЯ MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

- **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.

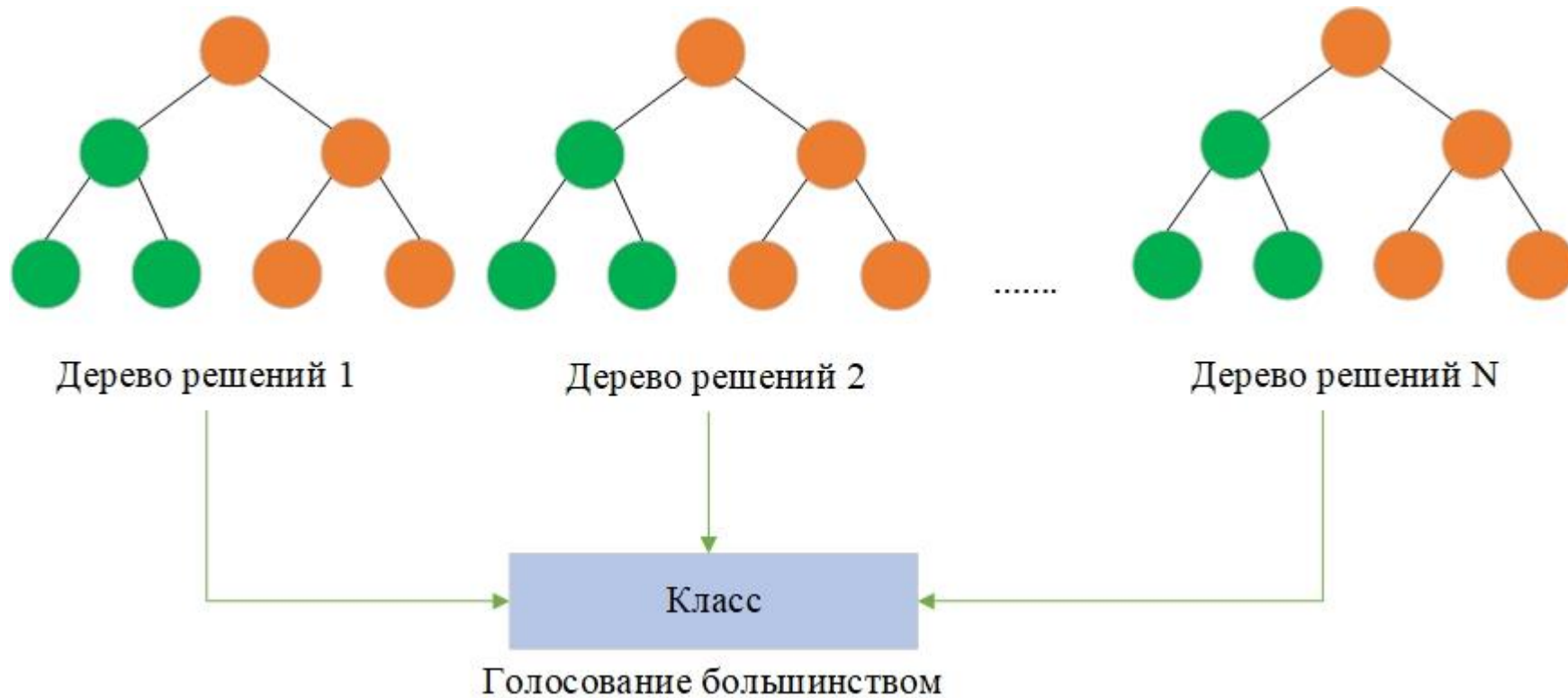
ДЕРЕВО РЕШЕНИЙ



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 классифицируется 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 MALWARE С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
algorithms = ['Naive Bayes', 'Logistic Regression', 'Decision tree', 'Random Forest', 'XGBoost', 'CatBoost', 'AdaBoost']
```

```
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!!!