ЛЕКЦИЯ 3

Обзор алгоритмов Naïve Bayes, Logistic Regression, Support Vector Machine, k-Nearest Neighbors для выявления DDoS атак

КЛАССИФИКАЦИЯ

Алгоритмы машинного обучения:

- Наивный байесовский классификатор (Naïve Bayes classifier);
- Логистическая регрессия (Logistic regression);
- Машина опорных векторов (Support vector machine);
- Метод k-ближайших соседей (k-nearest neighbors);
- Дерево решений (Decision tree);
- Случайный лес (Random forest);
- XGBoost.

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

Сбор данных

На этом этапе формируется датасет с трафиком сети, который включает как нормальный, так и вредоносный трафик (DDoS-атаки).
Источники данных:

- Лог-файлы серверов
- Захват трафика с помощью **Wireshark, Tcpdump**
- Датасеты, например, **CICDDoS2019, NSL-KDD, CAIDA DDoS**
- Мониторинг в реальном времени через **SNMP, NetFlow, sFlow**

Параметры трафика:

- IP-адреса отправителя и получателя
- Число пакетов за единицу времени
- Размер пакетов
- Количество соединений с одним IP
- Используемые протоколы (TCP, UDP, ICMP)
- Время жизни соединений (TTL)

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

	index	Unnamed: 0	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	...	Fwd Seg Size Min	Act Me
0	1886844	1811706	172.31.0.2-172.31.67.82-53-59931-17	172.31.67.82	59931	172.31.0.2	53	17	20/02/2018 10:44:37	20349	...	8	
1	1667897	4243700	172.31.66.81-13.89.190.129-49673-443-6	172.31.66.81	49673	13.89.190.129	443	6	20/02/2018 12:01:12	77452	...	20	
2	2590662	2634346	172.31.64.87-209.85.203.132-50315-443-6	209.85.203.132	443	172.31.64.87	50315	6	20/02/2018 09:31:35	21010	...	20	
3	2619695	25116	172.217.6.193-192.168.10.12-80-41588-6	192.168.10.12	41588	172.217.6.193	80	6	03/07/2017 06:16:17 PM	5436329	...	0	
4	2387220	3687623	172.31.0.2-172.31.66.28-53-51056-17	172.31.66.28	51056	172.31.0.2	53	17	20/02/2018 09:04:25	70607	...	8	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3186677	1081031	1473462	172.31.69.25-18.219.193.20-80-40358-6	18.219.193.20	40358	172.31.69.25	80	6	16/02/2018 11:25:29 PM	13964	...	0	
3186678	1649871	3439156	172.31.66.116-64.30.228.118-49756-443-6	172.31.66.116	49756	64.30.228.118	443	6	20/02/2018 08:33:35	128223	...	20	
3186679	1621073	4592346	172.31.67.12-216.58.198.66-49851-443-6	172.31.67.12	49851	216.58.198.66	443	6	20/02/2018 08:42:07	52905037	...	20	
3186680	1886020	372111	192.168.10.25-23.194.182.12-59968-443-6	192.168.10.25	59968	23.194.182.12	443	6	03/07/2017 09:33:39 PM	3	...	0	
3186681	992532	1119466	172.31.69.25-18.219.193.20-80-47790-6	18.219.193.20	47790	172.31.69.25	80	6	16/02/2018 11:22:50 PM	15017	...	0	

3186682 rows × 86 columns

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-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 DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

```
cat_cols = ['Flow ID', 'Src IP', 'Dst IP', 'Timestamp']
numerical_cols = list(set(DDoS.columns.values.tolist()) - set(cat_cols) - set(['Label']) - set(['Label_val']))
```

```
DDoS_cat = DDoS[['Flow ID', 'Src IP', 'Dst IP', 'Timestamp']]
DDoS_num = DDoS[numerical_cols]
```

```
DDoS['Flow ID']
```

```
0            172.31.0.2-172.31.67.82-53-59931-17
1         172.31.66.81-13.89.190.129-49673-443-6
2        172.31.64.87-209.85.203.132-50315-443-6
3         172.217.6.193-192.168.10.12-80-41588-6
4            172.31.0.2-172.31.66.28-53-51056-17
                          ...
99995       204.46.57.85-192.168.0.2-9440-5000-6
99996    83.190.207.124-192.168.0.2-54034-5000-6
99997    104.125.75.232-192.168.0.2-59426-5000-6
99998     176.33.53.116-192.168.0.2-40697-5000-6
99999    159.101.32.120-192.168.0.2-52542-5000-6
Name: Flow ID, Length: 3286682, dtype: object
```

```
le = LabelEncoder()
```

```
le_list = []
```

```
le.fit(DDoS['Flow ID'])
DDoS_cat['Flow ID'] = le.transform(DDoS_cat['Flow ID'])
le_list.append(le)
le = LabelEncoder()
le.fit(DDoS['Src IP'])
DDoS_cat['Src IP'] = le.transform(DDoS_cat['Src IP'])
le_list.append(le)
le = LabelEncoder()
le.fit(DDoS['Dst IP'])
DDoS_cat['Dst IP'] = le.transform(DDoS_cat['Dst IP'])
le_list.append(le)
le = LabelEncoder()
le.fit(DDoS['Timestamp'])
DDoS_cat['Timestamp'] = le.transform(DDoS_cat['Timestamp'])
le_list.append(le)
```

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
scaler = MinMaxScaler()
```

```
scaler.fit(DDoS_combined)
```

```
MinMaxScaler()
```

```
DDoS_scaled = pd.DataFrame(scaler.transform(DDoS_combined))
```

```
DDoS_scaled
```

	0	1	2	3	4	5	6	7	8	9	...	73	74	75	76	77	
0	0.349130	0.295863	0.221782	0.769459	0.080872	0.000057	0.0	0.000000	0.0	1.661833e-04	...	0.0	1.965699e-05	2.567790e-05	0.000000	0.0	0.0008
1	0.628769	0.294474	0.143003	0.822352	0.061017	0.000323	0.0	0.000000	0.0	6.454333e-04	...	0.0	5.164489e-06	8.336979e-09	0.000000	0.0	0.0067
2	0.468142	0.483225	0.225665	0.719018	0.000000	0.000088	0.0	0.000000	0.0	1.750833e-04	...	0.0	1.903855e-05	8.336979e-09	0.000000	0.0	0.7677
3	0.116046	0.403976	0.218061	0.022607	0.000000	0.015101	0.0	0.000000	0.0	4.499773e-02	...	0.0	7.357906e-08	3.045999e-04	0.000000	0.0	0.0012
4	0.251855	0.293873	0.221782	0.700256	0.044068	0.000196	0.0	0.000000	0.0	5.848583e-04	...	0.0	5.665161e-06	4.333062e-04	0.000000	0.0	0.0008
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3286677	0.984616	0.466377	0.365153	0.953876	0.020272	0.000002	1.0	0.172233	0.0	6.298930e-07	...	0.0	8.459306e-04	3.033699e-06	0.000358	1.0	0.0762
3286678	0.998066	0.938667	0.365153	0.953887	0.090211	0.000004	1.0	0.015220	1.0	6.515482e-07	...	1.0	7.757472e-04	5.974297e-06	0.000230	1.0	0.0762
3286679	0.000688	0.023537	0.365153	0.953899	0.033378	0.000003	0.0	0.037385	0.0	8.097211e-06	...	0.0	2.398819e-04	7.416420e-06	0.000329	0.0	0.0762
3286680	0.934107	0.311630	0.365153	0.953910	0.002239	0.000004	1.0	0.049674	1.0	4.030078e-06	...	1.0	5.670139e-05	5.992349e-06	0.000215	1.0	0.0762
3286681	0.040526	0.241005	0.365153	0.953922	0.065551	0.000002	0.0	0.166072	1.0	8.023003e-06	...	0.0	7.473625e-04	5.004525e-06	0.000259	0.0	0.0762

3286682 rows × 83 columns

Chi_square feature selection

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

```
fit_feat = bestfeatures.fit(DDoS_scaled,DDoS['Label_val'])
```

```
DDoS_scores = pd.DataFrame(fit_feat.scores_)
DDoS_columns = pd.DataFrame(DDoS_scaled.columns)
```

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

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

```
    Specs          Score
71     71  388598.076703
8       8  370244.162533
66     66  220029.050536
0       0  193953.300683
78     78  145341.568751
67     67  136127.520106
9       9  128360.427367
56     56  113455.768657
12     12   87027.910555
68     68   82895.984496
33     33   82196.174561
51     51   80343.230693
49     49   74135.051586
58     58   70569.787748
47     47   57870.084500
62     62   37909.910790
40     40   35396.278029
70     70   27414.719814
4       4   21301.278943
```

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-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.

ЭТАПЫ ВЫЯВЛЕНИЯ DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

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

```
x_train, x_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.3, random_state=42)
```

```
matrix_labels = ['ddos','Benign']
```

НАИВНЫЙ БАЙЕС

- Наивный Байес – один из самых простых и часто применяемых алгоритмов машинного обучения для классификации текстов, использующий вероятностный подход, основанный на теореме Байеса с сильными предположениями о независимости данных. Наивный Байес рассматривает каждый признак независимо от других признаков и оценивает вероятность влияния каждого из них на итоговый результат. В контексте классификации текстов он обучается на документах каждого класса и вычисляет условную вероятность того, что документ d относится к классу c

$$P(c \mid d) = \frac{P(c) \times P(d \mid c)}{P(d)}$$

где $d = \{x_1, x_2, \ldots, x_n\}$, x_i – вес i^{th} слова в документе d, и c – класс документа.

МАШИНА ОПОРНЫХ ВЕКТОРОВ

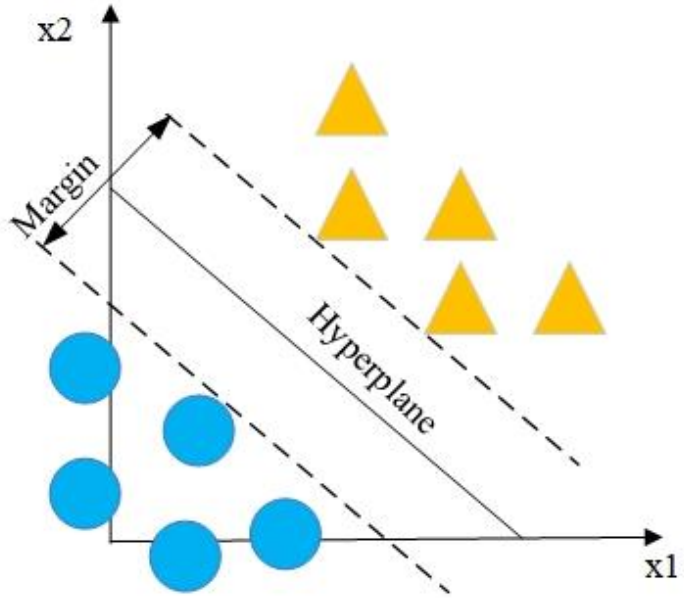
- Машина опорных векторов (Support vector machine) – еще один популярный алгоритм машинного обучения. Алгоритм использует пространство признаков, разделяемое гиперплоскостью, расположенной на максимальном расстоянии от ближайших точек двух классов обучающих данных. Чем шире граница, тем меньше ошибка классификатора, и достигается более эффективное разделение данных.
- Уравнение гиперплоскости записывается в следующем виде:

$$y_i(\vec{w} \times \vec{x} + b) \geq 0$$

где $\vec{x} = (x_1, x_2, ..., x_n)$ – вектор признаков; $\vec{w} = (w_1, w_2, ..., w_n)$ – вектор весов; y_i – выходное значение; b – сдвиг. Если значение больше или равно нулю, оно принадлежит к положительному классу. В противном случае относится к отрицательному классу.

МАШИНА ОПОРНЫХ ВЕКТОРОВ

- Разделяющая гиперплоскость Машины опорных векторов применяется преимущественно для двухклассовой классификации. Тем не менее, она без проблем адаптируется и для многоклассовой классификации с использованием метода «один против всех».

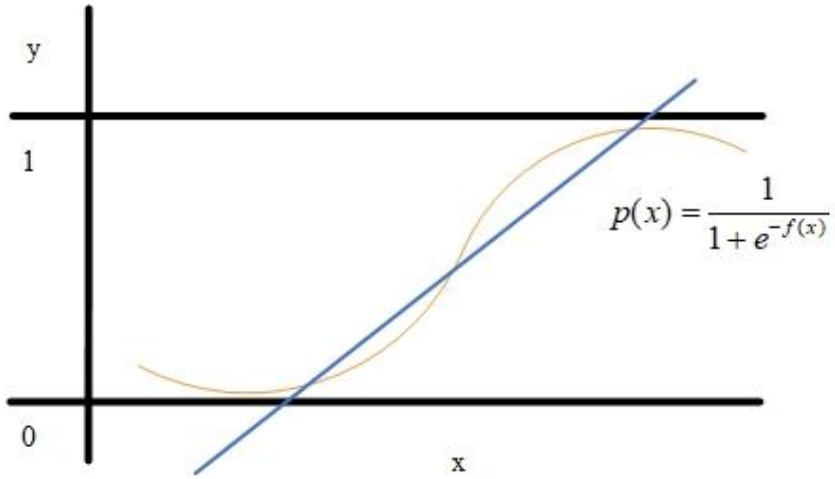


ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

- Логистическая регрессия предсказывает результат с использованием логистической функции

$$p(x) = \frac{1}{1 + e^{-f(x)}}$$

где $f(x) = w_0 + w_1 x_1 + \ldots + w_r x_r$ – линейная функция классификатора, $\vec{x} = (x_1, x_2, \ldots, x_n)$ – вектор признаков; $\vec{w} = (w_1, w_2, \ldots, w_n)$ – вектор весов. Логистическая функция $p(x)$ имеет вид сигмоида (рисунок 1.5) со значениями вероятности от 0 до 1. Документ d принадлежит к первому классу, если значение $p(x)$ близко к нулю. В противном случае его помещают во второй класс.



- В случае многоклассовой классификации используется подход «один против одного» (OvO), чтобы идентифицировать конкретный класс. В этом подходе датасеты с несколькими классами разбивается на несколько задач двоичной классификации, где каждый двоичный классификатор обучается на экземплярах, принадлежащих одному классу, и экземплярах, принадлежащих другому классу. Также используется метод «один против всех» (OvA), где множество двоичных классификаторов обучаются отличать экземпляры одного класса от всех других экземпляров. Преимущество OvO перед OvA заключается в том, что датасеты всех отдельных классификаторов сбалансированы, когда сбалансирован весь мультиклассовый датасет.

МЕТОД K-БЛИЖАЙШИХ СОСЕДЕЙ

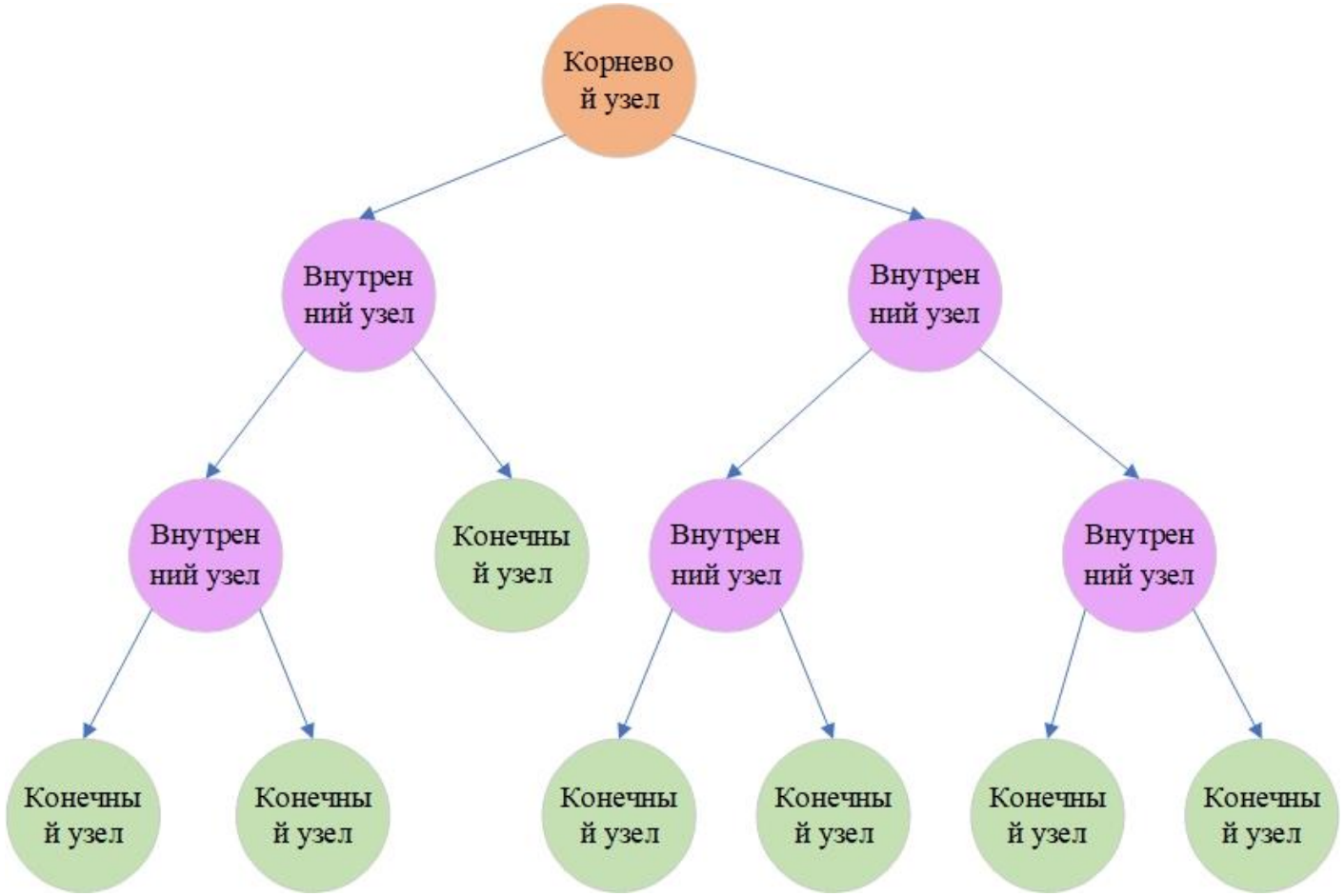
- Метод k-ближайших соседей – один из самых простых алгоритмов классификации данных. Он вычисляет расстояния между векторами и присваивает точки классу своих k ближайших соседних точек. В данном алгоритме вычисляется расстояние каждого объекта Для каждого объекта из тестовой выборки до всех объектов из обучающей выборки в пространстве признаков. Этот алгоритм обычно классифицирует документы с помощью наиболее широко используемой меры расстояния, называемой евклидовым расстоянием, которая определяется как $d(x,y)=\sqrt{\sum_{i=1}^{N}(a_{ix}-a_{iy})^2}$

где $d(x,y)$ расстояние между двумя документами; a_{ix} и a_{iy} веса i терма в документе x и y, соответственно; N номер уникального слова в списке документов. Метод k-ближайших соседей в процессе обучения запоминает векторы признаков и их метки классов. Там, где метки классов неизвестны, определяется расстояние между новым наблюдением и ранее запомненными векторами. Затем выбирается k ближайших векторов, и новый объект относится к классу, к которому принадлежит большинство. Выбор значения параметра k неоднозначен и требует экспериментальных подходов. При его увеличении улучшается точность классификации, но границы между классами становятся менее четкими. Данный метод показывает хорошие результаты классификации, однако основным его недостатком является высокая вычислительная трудоемкость при увеличении размера обучающей выборки.

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

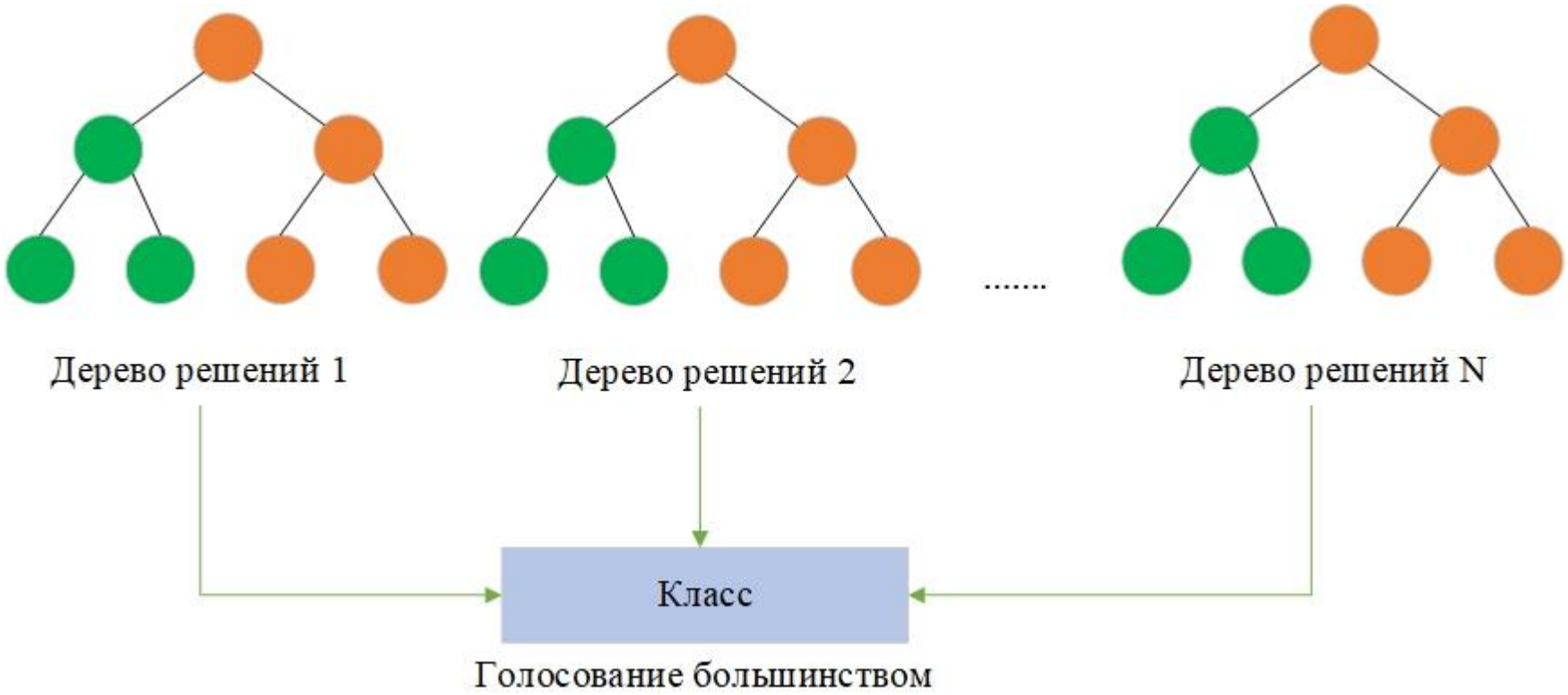


СЛУЧАЙНЫЙ ЛЕС

- 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 DDOS-АТАК С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

```
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='0:1')
```

```
k = 0
for i in classifiers:
    i.fit(x_train, y_train)
    if k==4:
        pickle.dump(i, open('XGBoost_ddos.sav', 'wb'))
    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!!!