БАЛТИЙСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМ. И. КАНТА ИНСТИТУТ ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт по индивидуальному практическому заданию на тему:

«Разработка алгоритма и программного средства выявления фишинговых сайтов с использованием методов машинного обучения» по дисциплине

«Информационная безопасность корпоративных сетей».

Выполнил студент 4 курса  
направления информационная безопасность:

Бочкарев Владислав Витальевич

Проверил: Бугаев Юрий Николаевич

г. Калининград 2024

**Оглавление**

[**Введение** 3](#_Toc185064797)

[**Разбор кода** 6](#_Toc185064798)

[**Перспективы и вывод** 10](#_Toc185064799)

# **Введение**

Цель данного индивидуального практического задания является создание программы, которая на основе машинного обучения будет выявлять фишинговые сайты. Основой обучения станет модель XGBoost и база данных признаков фишинговых и безопасных сайтов. Модель **XGBoost** (eXtreme Gradient Boosting) — это популярный алгоритм машинного обучения, основанный на методе градиентного бустинга. Она используется для задач классификации, регрессии и ранжирования. Разработан XGBoost с целью повышения скорости, точности и масштабируемости.

**Градиентный бустинг** — это метод машинного обучения, используемый для повышения точности предсказания моделей. Алгоритм строится на последовательной комбинации слабых моделей (обычно деревьев решений), чтобы создать сильную предсказательную модель.

Принцип работы градиентного бустинга:

1. **Инициализация модели:**  
   Обучение начинается с простой модели, которая может просто предсказать среднее значение целевой переменной (например, средний класс или среднее значение для регрессии).
2. **Обучение слабых моделей:**  
   На каждом шаге вносится новая слабая модель, которая обучается на ошибках (остатках) предыдущей модели.

Остаток — это разница между истинным значением и предсказанием текущей модели.

Новая модель пытается уменьшить эти остатки, "подгоняя" себя под те данные, где предыдущие модели ошибались.

1. **Обновление итоговой модели:**  
   Итоговый результат складывается из результатов всех слабых моделей с учетом их вклада. Каждая новая модель добавляется так, чтобы минимизировать ошибку всей модели.
2. **Использование градиента:**  
   Название "градиентный" связано с использованием метода оптимизации — **градиентного спуска**, который минимизирует функцию потерь.

Функция потерь показывает, насколько сильно предсказание отличается от истинного значения.

На каждом шаге вычисляется градиент функции потерь, чтобы определить направление улучшения.

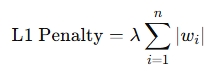
Основные особенности XGBoost:

1. XGBoost спроектирован для высокой скорости вычислений и эффективного использования памяти. Он поддерживает многопоточность, распределённые вычисления и использует оптимизированные алгоритмы.
2. XGBoost может автоматически обрабатывать пропуски в данных, интерпретируя их как отдельные ветви дерева решений.
3. Алгоритм использует L1 (lasso) и L2 (ridge) регуляризацию, чтобы уменьшить вероятность переобучения.

**L1 и L2 регуляризация** — это методы, используемые для предотвращения переобучения моделей машинного обучения. Они добавляют штраф к функции потерь, чтобы ограничить значения коэффициентов модели и сделать её более устойчивой.

**L1 (Lasso)**:

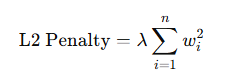
**L1 регуляризация** добавляет к функции потерь штраф в виде суммы абсолютных значений коэффициентов.



где Wi — это коэффициенты модели, а λ — гиперпараметр, регулирующий силу регуляризации.

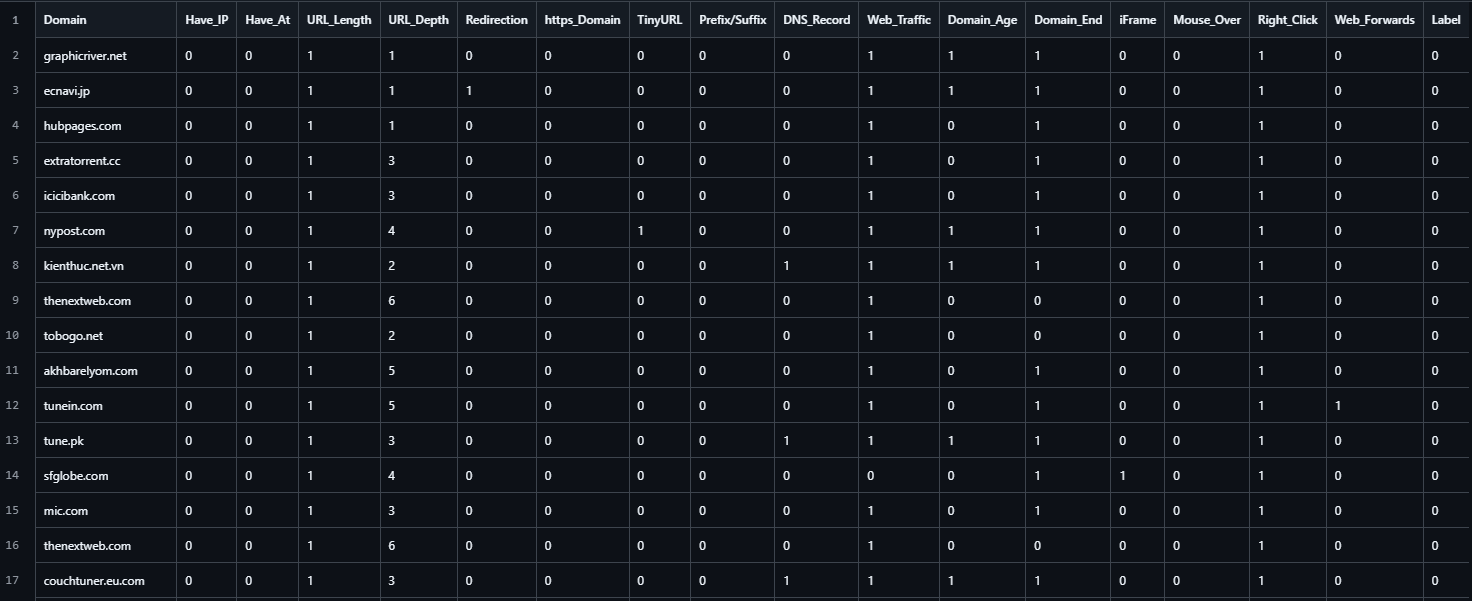
**L2 (Ridge):**

**L2 регуляризация** добавляет штраф в виде суммы квадратов коэффициентов.



1. **Скорость обучения**  
   За счёт использования методов, таких как предварительная сортировка данных и блочные структуры, XGBoost обучается быстрее, чем традиционные реализации градиентного бустинга.

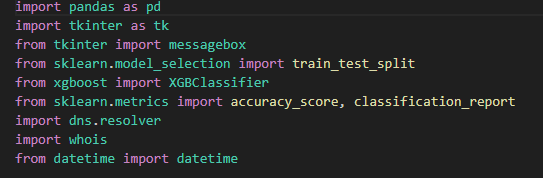
База данных фишинговых и безопасных сайтов состоит из 18 столбцов, из которых 16 – это признаки сайтов, первый столбец – это domain и последний столбец (Label) – это определение фишингового или безопасного сайта (0 – безопасный сайт, 1 – фишинговый сайт).



Сама программа была реализована на Python с добавление библиотек, таких как: pandas, numpy, tkinter, sklearn, xgboost, whois, dns.resolver, datetime.

# **Разбор кода**

1. **Импорт нужных библиотек для реализации алгоритма**



Библиотека pandas — это библиотека, которая служит для обработки и анализа данных

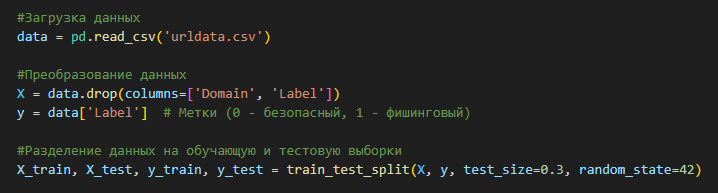
Библиотека tkinter - библиотека для организации диалогов в программе с помощью оконного графического интерфейса GUI

Библиотека sklearn ( Scikit-learn) – служит для задач машинного обучения и оценки качества модели.

Классификатор Xgboost – алгоритм машинного обучения

Dns.resolver – для работы с DNS-записями доменов

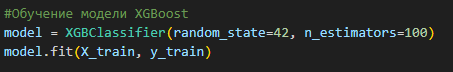
Whois – для получения информации о домене, а именно дата создания  
datetime – для работы с датами

1. **Загрузка и обработка данных**  
   

**data = pd.read\_csv('urldata.csv')** – загрузка данных с базы данных признаков, осуществляется с помощью библиотеки pandas.  
**X = data.drop(columns=['Domain', 'Label'])** – удаление столбцов Domain и Label, чтобы оставить только признаки

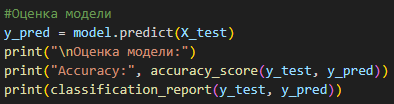
**y = data['Label']** – извлекаем метки обозначающие фишинговые (1) и безопасные сайты (0).

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)** – Разделение данных на обучающую и тестовую выборки в соотношение 70:30.

1. **Обучение модели**  
   

**model = XGBClassifier(random\_state=42, n\_estimators=100)** – Создаёт модель XGBoost с фиксированным генератором случайных чисел и 100 деревьями решений  
**model.fit(X\_train, y\_train)** – Обучение модели по данным

1. **Оценка модели**



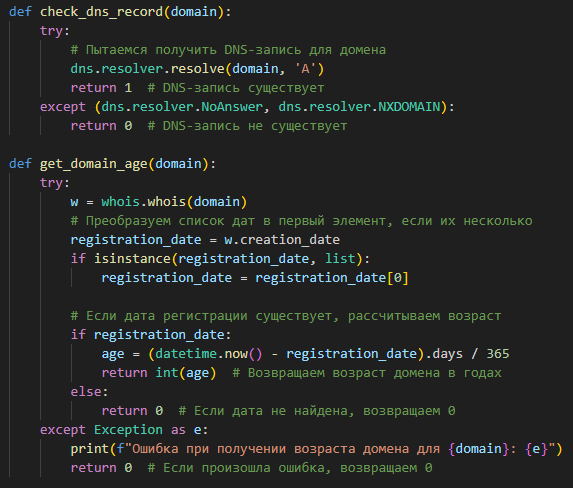
**y\_pred = model.predict(X\_test)** - прогнозы делаются на тестовой выборке.

**print ("Accuracy:", accuracy\_score(y\_test, y\_pred)) -** выводится точность модели.

**print (classification\_report(y\_test, y\_pred)) -** выводится полный отчёт о классификации.

1. **Функции для извлечения признаков  
   **

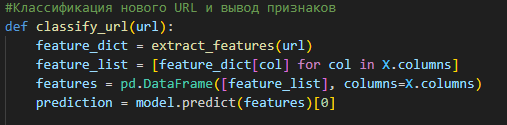
Функция extract\_features(url) – извлекает признаки из введённого url. Извлекает такие признаки как: содержание домен IP-адрес вместо доменного имени, наличие символа @, длина URL, глубина URL ( число сегментов), содержит ли URL больше одного направления, наличие протокола https, является ли домен коротким, содержание дефиса в домене, DNS-записи домена, web-traffic, возраст домена в годах, заканчивается ли домен популярными зонами, iFrame, изменение действия мыши, блокировка правого клика, количество веб-переходов.



Check\_dns\_record(domain) – Проверка существования DNS-записи для домена. Возвращает 1 если есть запись и 0 если нет записи.

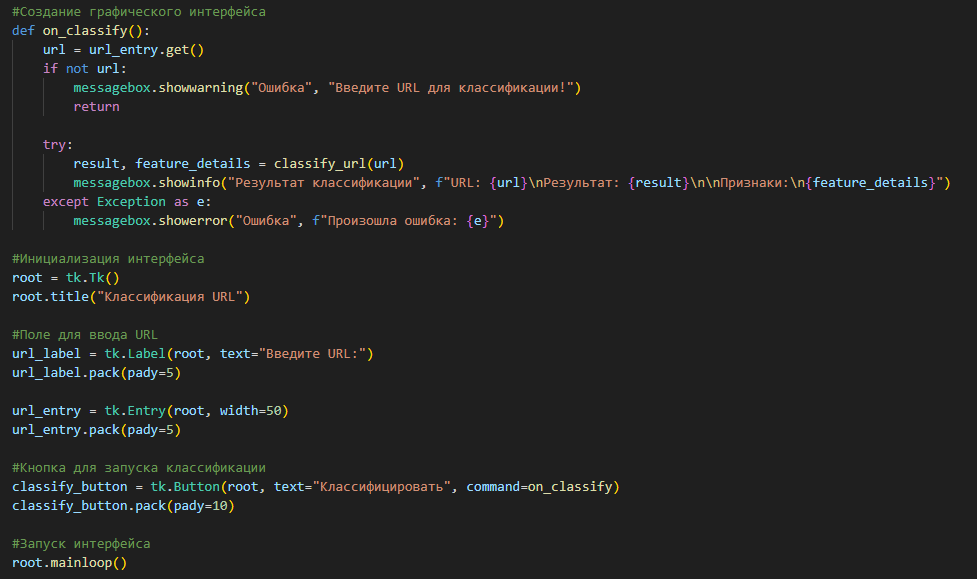
Get\_domain\_age(domain) – Рассчитывает возраст домена в годах, используя данные из whois

1. **Классификация нового URL**



Classify\_url(url) – определяет фишинговый или безопасный сайт. Сначала извлекаются признаки URL через extract\_features, дальше формируется датафрейм признаков и после модель прогнозирует результат. В конце функция возвращает результат и подробности по признакам.

1. **Создание интерфейса**

****

def on\_classify (): - Обрабатывает нажатие кнопки и выводит результат классификации в окне сообщения.

root = tk.Tk () - Инициализация графического интерфейса.

url\_entry = tk.Entry (root, width=50) - Поле для ввода URL.

classify\_button = tk.Button (root, text="Классифицировать", command=on\_classify) - Кнопка для запуска классификации.

root.mainloop () - Запуск интерфейса.

# **Перспективы и вывод**

Алгоритм и сам код требуют доработки и дальнейшего улучшения, чтобы прогнозы были более точными и верными. На данный момент защита от фишинговых сайтов является одной из главных задач защиты обычных пользователей. Так как не многие соблюдают правила и рекомендации по защите своих данных в интернете и часто заходят на фишинговые сайты даже не понимая этого. И всё же, даже если человек заподозрит что-то не ладное в сайте, то без должных навыков ему самому сложно будет понять фишинговый это сайт или безопасный. Именно поэтому существует множество сервисов, которые ясно и точно покажут вам, что на самом деле скрывается под безобидной обложкой якобы безопасного сайт. В дальнейшей перспективе в данный код можно внедрить:  
**1. Новые признаки**

1. Проверка whois-данных

Проверить наличие информации о владельце домена (почта, организация, страна). Количество DNS-серверов, указанных для домена.

b) Состояние сертификата SSL

Проверить, используется ли SSL-сертификат, его уровень верификации (EV, DV, OV) и срок действия.  
Библиотека: ssl, certifi.

c) Количество поддоменов

Подсчитать количество поддоменов в URL, так как злоумышленники часто используют длинные поддомены.

d) Проверка на совпадение с известными брендами

Добавить проверку на наличие имени известного бренда в домене (например, goog1e.com вместо google.com).

**2. Улучшения модели**

a) Добавить кросс-валидацию

Для более точной оценки качества модели вместо простого разбиения на обучающую и тестовую выборки.

b) Поиск оптимальных гиперпараметров

Использовать GridSearchCV или RandomizedSearchCV для подбора параметров модели XGBoost.

**3. Улучшение обработки данных**

a) Обработка категориальных данных

Если появляются категориальные признаки (например, тип SSL-сертификата), их можно перекодировать с помощью One-Hot Encoding.

b) Нормализация признаков

Многие алгоритмы лучше работают с нормализованными данными. Примените MinMaxScaler или StandardScaler

**4. Улучшение интерфейса**

a) История классификации

Добавить таблицу или список в интерфейсе для сохранения истории введённых URL и их результатов.

b) Показать важность признаков

Выводить список признаков, которые модель считает наиболее важными (доступно в XGBoost)

c) Подсказки пользователю

Добавить всплывающие подсказки для объяснения признаков.