МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Институт искусственного интеллекта
Кафедра компьютерной и информационной безопасности

КУРСОВАЯ РАБОТА
по дисциплине
«Интеллектуальные системы информационной безопасности»

Тема курсовой работы:

Интеллектуальная система обнаружения ботов в социальных сетях

Студент группы КТСО-03-22 С.Р. Ягодарова

Руководитель курсовой работы: А.Н. Чесалин

Работа представлена к защите «____»_____20__ г. _____
(подпись студента)

«Допущен к защите» «____»_____20__ г. _____
(подпись руководителя)

Москва 2024

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта

Кафедра компьютерной и информационной безопасности

Утверждаю

Заведующий
Кафедрой _____ А.Н. Чесалин

ЗАДАНИЕ
на выполнение курсовой работы
по дисциплине
«Интеллектуальные системы информационной безопасности»

Студент: _____ Группа: _____

1.Тема: Интеллектуальная система обнаружения ботов в социальных сетях

2. Исходные данные:
Информация о трёх тысячах пользователей социальной сети «ВКонтакте»
Статьи: Евсеева А.О., Гумерова Р.И., Катасёв А.С., Кирпичников «Идентификация ботов в социальных сетях на базе технологий интеллектуального анализа данных»
Анна Яковленко https://habr.com/ru/companies/solarsecurity/articles/700560/
3. Краткая аннотация работы (перечень подлежащих разработке вопросов):
- исследуются современные методы классификации
- производится анализ применимости методы классификации для обнаружения ботов в социальных сетях
- на исходных данных обучается модель для обнаружения ботов в социальных сетях
4. Срок представления к защите курсового проекта (работы): до 24 декабря 2024 г.

Руководитель курсовой работы: _____

Задание принял к исполнению _____

1. Теоретическая часть

1.1. Алгоритмы, подходящие для данной задачи

Пользователи в социальной сети “ВКонтакте” обладают специфическими характеристиками, которые позволяют разделять их на две основные группы: боты и не боты. Задачей данной курсовой работы является создание интеллектуальной системы, способной выполнять такую классификацию с использованием алгоритмов машинного обучения. Для достижения поставленной цели необходимо ответить на следующие ключевые вопросы:

1. Какие задачи машинного обучения решаются в данной работе?
2. Какие алгоритмы машинного обучения оптимальны для решения этой задачи?
3. Какие признаки пользователей “ВКонтакте” следует использовать в качестве входных данных для алгоритмов?

1.1.1. Задачи машинного обучения

В машинном обучении существует две основные категории задач, которые актуальны для данной курсовой работы:

1. Классификация
2. Кластеризация

Классификация

Классификация — это задача с учителем, где цель заключается в том, чтобы разделить объекты на несколько категорий или классов на основе имеющихся обучающих данных. В обучающей выборке каждому объекту заранее присвоен класс. В нашем случае, классы — это "бот" и "не бот".

1.1.2. Кластеризация

Кластеризация — это задача без учителя, где алгоритм самостоятельно группирует объекты на основе сходства признаков, не имея заранее определённых меток. Однако специфика наших данных такова, что метки

классов уже известны (есть информация, является ли пользователь ботом), поэтому обучение с учителем и задача классификации являются более подходящими для данной работы.

1.1.3. Выбор алгоритмов машинного обучения

Для решения задачи классификации существует ряд популярных алгоритмов машинного обучения, каждый из которых обладает своими особенностями, достоинствами и недостатками:

1. **RandomForestClassifier** (Случайный лес)
    - Принцип работы: состоит из множества деревьев решений. Каждое дерево строится на случайных подмножествах признаков и данных. Результатом является совокупное "голосование" деревьев, определяющее класс объекта.
    - Преимущества:
        - Высокая точность на больших данных.
        - Устойчивость к переобучению благодаря случайности.
        - Хорошо справляется с разнородными и пропущенными данными.
    - Недостатки:
        - Модели с большим числом деревьев могут быть медленными для обучения.
        - Не всегда легко интерпретировать результаты.
2. **Логистическая регрессия**
    - Принцип работы: рассчитывает вероятность принадлежности объекта к тому или иному классу с использованием логистической функции.
    - Преимущества:
        - Простая и быстрая реализация.
        - Хорошо работает на линейно разделимых данных.
    - Недостатки:

        - Ограничена в применении для сложных нелинейных зависимостей.

3. **SVM (Support Vector Machine)**
    - Принцип работы: строит гиперплоскость, которая максимально разделяет два класса.
    - Преимущества:
        - Эффективен при небольших выборках и высокой размерности данных.
    - Недостатки:
        - Медленная скорость на больших данных.
        - Трудности с выбором ядра для нелинейных данных.
4. **kNN (k ближайших соседей)**
    - Принцип работы: определяет класс объекта на основе классов k ближайших к нему соседей.
    - Преимущества:
        - Простая реализация.
        - Не требует долгого обучения (но требует ресурсов на предсказание).
    - Недостатки:
        - Чувствителен к масштабу данных и шумам.
        - Медленно работает на больших объёмах данных.
5. **Нейронные сети**
    - Принцип работы: моделирует работу нейронов мозга, обучаясь на большом количестве данных.
    - Преимущества:
        - Высокая гибкость и способность моделировать сложные зависимости.
    - Недостатки:
        - Требует большого количества данных и ресурсов.
        - Трудно интерпретировать результаты.

6. **XGBoost/LightGBM**
    - Принцип работы: использует метод градиентного бустинга, объединяя слабые деревья решений в сильную модель.
    - Преимущества:
        - Высокая скорость и точность.
        - Оптимизирован для больших данных.
    - Недостатки:
        - Может переобучаться на малых выборках.

1.1.4. Выбор метода для работы

В данной курсовой работе был выбран RandomForestClassifier по следующим причинам:

1. Высокая точность и надёжность.
2. Эффективность на больших данных и разнородных признаках.
3. Устойчивость к переобучению, что особенно важно для работы с новой, непроверенной выборкой.

1.1.5. Вывод

Таким образом, задача классификации с использованием RandomForestClassifier позволяет эффективно разделять пользователей “ВКонтакте” на ботов и реальных пользователей. Использование качественных признаков (пункт 2.1) и мощных алгоритмов машинного обучения обеспечивает высокую точность результатов, а выбранный метод случайного леса дополнительно гарантирует устойчивость и надёжность системы.

2. Практическая часть

2.1. Сбор данных

Для того, чтобы классификатор RandomForestClassifier работал, ему нужны обучающая и тестовая выборки. В контексте данной задачи это два больших списка пользователей: ботов и не ботов. Для того, чтобы их составить, было написано несколько программ.

1. get_list_bots.py – программа для составления списка ботов. Её последовательность действий можно описать примерно так:

a) Делает запрос на сайт botnadzor.org, чтобы получить html-страницу со списком найденных ботов. С помощью парсинга этой страницы можно получить список уникальных идентификаторов пользователей-ботов.

b) Делает запрос на сайт regvk.ru, чтобы по уникальному идентификатору найти дату регистрации пользователя.

c) С помощью VK API делает запрос пользователя, который возвращает список указанных полей (например, имя, фамилия, дата рождения). Для этого нужно было зарегистрировать приложение в ВК и получить специальный токен.

d) Делает запрос друзей пользователя и вычисляет процент ботов среди них.

e) Делает запрос постов пользователя и вычисляет дату первого поста и среднее время, проходящее между публикациями постов.

f) Всю полученную информацию программа записывает в файл bots_for_training.txt

Таким образом, в результате выполнения программы получается файл, содержащий записи примерно такого вида: {'ID': 848604856, 'Дата регистрации': '6 февраля 2024 года', 'Дата рождения': '5.1.2005', 'Есть фото': 1, 'Статус': 'RU', 'Количество подписчиков': 410, 'Пол': 1, 'Имя': 'Irina', 'Фамилия': 'Kamshina', 'Количество друзей': 328, 'Процент друзей-ботов': '2.44%', 'Количество постов': 102, 'Дата первого поста': '2024-02-07 13:57:41', 'Средний интервал между постами': '2.51 дней'}. Каждая такая запись соответствует

одному пользователю-боту. В общей сложности файл содержит информацию о 1736 ботах.

2. usernadzor.py – программа для составления списка пользователей. В целом, она выполняет те же самые действия, что и get_list_bots.py, за исключением добывания уникальных идентификаторов пользователей. В данной программе они достаются из файла users_id.txt. А заполняется этот файл с помощью кода get_uid.py, который запрашивает список пользователей групп, тем самым получая идентификаторы. Программа usernadzor.py записывает пользователей в файл users_for_training.txt. Вот пример записи в этом файле: {'ID': 140017668, 'Дата регистрации': '3 июля 2011 года', 'Дата рождения': '27.9', 'Есть фото': 1, 'Статус': '', 'Количество подписчиков': 542, 'Пол': 1, 'Имя': 'Мария', 'Фамилия': 'Бармина', 'Количество друзей': 0, 'Процент друзей-ботов': '0%', 'Количество постов': 0, 'Дата первого поста': 'Нет постов', 'Средний интервал между постами': 'Нет постов'}. Таких записей 1306. Таким образом у на есть два файла: в одном – список ботов, в другом – список пользователей. При этом собрано много информации обо всех пользователях.

Теперь можно ответить на вопрос, какие признаки пользователей “ВКонтакте” важны для классификации. Исходя из личных наблюдений и расчёта средних значений можно сказать, что бота от обычного человека отличает главным образом следующие 4 фактора:

1. Средний интервал времени между постами
2. Количество дней с момента регистрации
3. Процент ботов в друзьях
4. Количество дней с момента регистрации до выхода первого поста

2.2. Обучение модели

Дальнейший код создавался в среде Google Colab.

Итак, после получения файлов нужно использовать их содержимое для обучения модели и проверки. Для этого был написан код, последовательность действий которого можно описать примерно так:

1. Открытие файлов со списками пользователей и считывание оттуда данных. Результат – два списка: в одном – боты, во втором – обычные пользователи.

```
list_bots = []
with open("/content/bots_for_training.txt", "r") as file:
  for line in file:
        if line.strip():  # Убедимся, что строка не пустая
            list_bots.append(line.strip())

list_users = []
with open("/content/users_for_training.txt", "r") as file:
  for line in file:
        if line.strip():  # Убедимся, что строка не пустая
            list_users.append(line.strip())
```

Код 1. Заполнение списков

2. Поиск с помощью регулярных выражений в строке нужных данных: даты регистрации, процента друзей-ботов, даты первого поста, среднего времени между выходом постов.

```
pattern = r"Дата регистрации':\s*'(\d{1,2})\s+([а-яА-Я]+)\s+(\d{4})"
pattern1 = r"'Процент друзей-ботов':\s*'([\d.]+)%'"
pattern2 = r"'Дата первого поста':\s*'(\d{4}-\d{2}-\d{2}
\d{2}:\d{2}:\d{2})'"
pattern3 = r"'Средний интервал между постами':\s*'([\d.]+)\s+дней'"
```

Код 2. Паттерны для поиска

```
  match = re.search(pattern1, bot)
  if match:
    bot_friends_percentage = float(match.group(1)) # Сколько процентов
друзей - боты
    bot_friends_percentage_list.append(bot_friends_percentage)
    one.append(bot_friends_percentage)
    #print(bot_friends_percentage)
  else:
    continue
```

Код 3. Поиск процента друзей-ботов

3. Составление вектора признаков: высчитывание разностей между датой регистрации и датой первого поста/текущей датой, сохранение этих данных и процента друзей-ботов, среднего времени между выходом постов в вектор. Пример вектора: [355, 8.15, 18, 10.14].

```
match = re.search(pattern2, bot)
if match:
    first_post_date = match.group(1)
    first_post_date = datetime.strptime(first_post_date, '%Y-%m-%d
%H:%M:%S')
    days_until_first_post = (first_post_date - reg_date).days # Сколько
дней прошло до первого поста
    days_until_first_post_list.append(days_until_first_post)
    one.append(days_until_first_post)
    #print(days_until_first_post)
else:
  continue
```

Код 4. Пример: расчёт количества дней до первого поста

4. Подобный вектор создаётся для каждого аккаунта, все векторы добавляются в список. Таким образом получается два списка: в одном векторы ботов, в другом – обычных пользователей. Эти списки соединяются в один, для которого составляется список меток (0 – не бот, 1 - бот).

```
all_list = list_signs_bots + list_signs_users
labels = [1] * len(list_signs_bots) + [0] * len(list_signs_users)
print("Признаки: ",all_list)
print("Метки: ", labels)
```

Код 5. Создание общих списков признаков и меток

5. Оба списка синхронно перемешиваются, чтобы быть в хаотичном порядке.

```
combined = list(zip(all_list, labels))
random.shuffle(combined)
# Разделяем обратно
X, y = zip(*combined)
X = list(X)
y = list(y)
```

Код 6. Перемешивание списка

6. Эти данные уже можно использовать для обучения модели. С помощью функции train_test_split данные делятся на обучающую и тестовую выборки. Далее создаётся модель и обучается на тестовых данных с помощью

функции fit(). Затем создаётся y_pred – предсказанные для тестовой группы результаты.

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
# Оценка качества модели
accuracy = accuracy_score(y_test, y_pred)
print("Точность модели:", accuracy)
print("Отчёт о классификации:\n", classification_report(y_test, y_pred))
```

Код 7. Работа с моделью

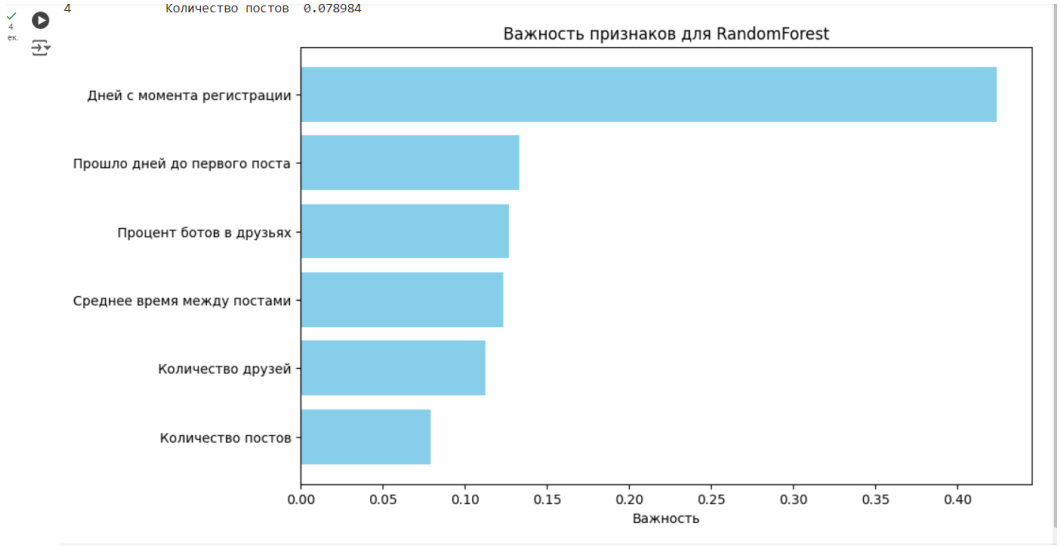
7. После этого программа выводит свой отчёт

```
Признаки:  [[315, 2.44, 1, 2.51], [312, 0.0, 48, 47.85],
Метки:  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
Перемешанные признаки: [[725, 16.13, 690, 2.46], [1460, 0
Перемешанные метки: [1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
Обучающая выборка: [[313, 0.0, 43, 6.51], [2686, 0.0, 260
Тестовая выборка: [[1515, 0.0, 1008, 5.08], [912, 28.81,
Точность модели: 0.9367311072056239
Отчёт о классификации:
              precision    recall  f1-score   support

           0       0.94      0.84      0.88       165
           1       0.94      0.98      0.96       404

    accuracy                           0.94       569
   macro avg       0.94      0.91      0.92       569
weighted avg       0.94      0.94      0.94       569
```

Скриншот 1. Отчёт о классификации



Скриншот 2. Диаграмма важности

2.3. Использование модели

Для использования модели бы написан отдельный блок кода. Его последовательность действий:

1. Для начала он запрашивает уникальный идентификатор пользователя. Это нужно для того, чтобы сделать необходимые запросы через VK API (также, как и в коде usernadzor.py).

2. Программа формирует строку вида {'ID': 15434011, 'Дата регистрации': '1 июля 2008 года', 'Дата рождения': '28.11', 'Есть фото': 1, 'Статус': '', 'Количество подписчиков': 273, 'Пол': 1, 'Имя': 'Виктория', 'Фамилия': 'Ефремова', 'Количество друзей': 61, 'Процент друзей-ботов': '0.00%', 'Количество постов': 14, 'Дата первого поста': '2016-07-16 11:15:29', 'Средний интервал между постами': '221.93 дней'}.

3. Далее, также, как и для обучения модели, формируется вектор из четырёх элементов. Этот вектор можно передать модели для предсказания результата.

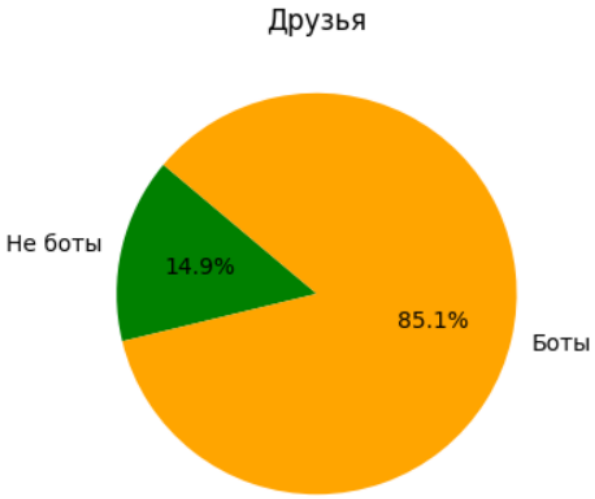
```
new_user = [one]
result = model.predict(new_user)
p = model.predict_proba(new_user)
```

Код 8. Предсказание результата и получение вероятностей

```
labels = ['Не бот', 'Бот']
colors = ['gold', 'yellowgreen']  # Цвета для сегментов
explode = (0, 0)
plt.figure(figsize=(4, 4))
plt.pie(p[0], explode=explode, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=False, startangle=140)
plt.title('Вероятность принадлежности к классам')
plt.show()
labels1 = ['Не боты', 'Боты']
colors1 = ['green', 'orange']
plt.figure(figsize=(4, 4))
plt.pie([100 - bot_friends_percentage, bot_friends_percentage],
explode=explode, labels=labels1, colors=colors1,
        autopct='%1.1f%%', shadow=False, startangle=140)
plt.title('Друзья')  # Заголовок
plt.show()
```

Код 9. Вывод диаграмм о количестве ботов в друзьях и вероятности принадлежности к классу

```
Введите уникальный идентификатор пользователя:
855404874
Информация о профиле:
ID :  855404874
Дата регистрации :  21 марта 2024 года
Дата рождения :  28.6.1972
Есть фото :  1
Статус :
Количество подписчиков :  300
Пол :  2
Имя :  Сергей
Фамилия :  Егоров
Количество друзей :  295
Процент друзей-ботов :  85.08%
Количество постов :  29
Дата первого поста :  2024-03-25 18:09:34
Средний интервал между постами :  1.45 дней
Дней с момента регистрации: 271
Процент ботов в друзьях:  85.08
Прошло дней с момента регистрации до выхода первого поста:  4
Среднее время между выходом постов:  1.45
[271, 85.08, 4, 1.45]
```




```
Результат предсказания: Бот
```

Скриншоты 2-4. Вывод программы

3. Заключение

В ходе выполнения курсовой работы была успешно разработана интеллектуальная система на основе алгоритмов машинного обучения для обнаружения ботов в социальной сети “ВКонтакте”. Основные результаты и достижения работы можно сформулировать следующим образом:

1. Постановка задачи и выбор подхода:

   Определена задача классификации, поскольку известны метки классов "бот" и "не бот". Использование обучения с учителем позволило создать точную модель на основе обучающих данных.

2. Выбор алгоритма:

   В ходе анализа различных алгоритмов машинного обучения, таких как логистическая регрессия, SVM, kNN, нейронные сети и градиентный бустинг, был выбран RandomForestClassifier. Данный метод продемонстрировал:

   - Высокую точность классификации;
   - Устойчивость к переобучению;
   - Способность эффективно обрабатывать большое количество признаков.

3. Выделение признаков пользователей:

   Были отобраны и обработаны ключевые признаки пользователей социальной сети:

   - Дата регистрации
   - Дата выхода первого поста
   - Статистика выходов постов
   - Боты в друзьях

   Эти признаки позволили модели точно различать реальных пользователей и ботов.

4. Результаты работы модели:

   Разработанная модель успешно обучилась на тренировочных данных и

показала высокие результаты на тестовой выборке. Достигнуты следующие показатели:

- Точность классификации (Accuracy): 0.94%;
- F1-мера: 0.88%.

Высокие значения метрик подтверждают эффективность и надёжность разработанной системы.