DOI: https://doi.org/10.15276/ict.02.2025.49
УДК 004.8:004.75:004.056.5

Вебсервіс для виявлення шахрайських транзакцій за допомогою технологій штучного інтелекту

Шилін Михайло Андрійович[1)]
Студент каф. Інженерії програмного забезпечення
ORCID: https://orcid.org/0009-0003-6527-9356; 10387444@stud.op.edu.ua
Пригожев Олександр Сергійович[1)]
Канд. техніч. наук, доцент каф. Інженерії програмного забезпечення
ORCID: https://orcid.org/0000-0001-8532-8897; o.s.prygozhev@op.edu.ua.
[1)] Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У даних тезах описано етапи програмної реалізації програмної системи для виявлення шахрайських транзакцій за допомогою технологій машинного навчання та штучного інтелекту. Опис етапів програмної реалізації даної системи містить у собі: аналіз аналогічних програмних систем, проектування архітектури програмної системи, порівняння алгоритмів машинного навчання та вибір найбільш ефективного або ефективних, навчання та тестування моделі, вибір стеку технологій для програмної реалізації, а також основні кроки програмної реалізації. Етап «аналіз аналогічних програмних систем» містить детальний опис метрик, характеристик та особливостей на які варто звернути увагу для реалізації подібної системи. Даний етап допомагає виявити сильні та слабкі сторони аналогічних систем. Етап «проектування архітектури програмної системи» містить опис проектування архітектури програмної системи з обґрунтуванням ефективності даної архітектури. Етап «порівняння алгоритмів машинного навчання та вибір найбільш ефективного або ефективних» містить опис існуючих метрик ефективності моделей машинного навчання, а також найважливіших метрик ефективності саме для тих моделей, які призначені для виявлення шахрайських транзакцій. Етап «навчання та тестування моделі» містить опис підготовки набору даних до навчання моделі, проектування моделі машинного навчання, навчання моделі, а також її тестування. Етап «вибір стеку технологій для програмної реалізації» містить опис вибору стеку технологій для програмної реалізації програмної системи для виявлення шахрайських транзакцій з обґрунтуванням того, чому був обраний саме цей стек технологій. Етап «основні кроки програмної реалізації» містить опис основних кроків програмної реалізації подібної програмної системи. Проектування та програмна реалізація програмної системи для виявлення шахрайських транзакцій є актуальною на сьогоднішній день, оскільки є актуальною проблема шахрайства у транзакціях та адаптації шахраїв до вже існуючих програмних систем, які призначені для протидії шахрайству у транзакціях. Метою написання даних тез є опис етапів програмної реалізації системи для виявлення шахрайських транзакцій за допомогою технологій машинного навчання та штучного інтелекту, яка порівняла б у собі сильні сторони вже існуючих аналогічних систем та мінімізувала б їх недоліки. Також, тези містять опис підготовчих дій без яких розробка програмної системи для виявлення шахрайських транзакцій за допомогою технологій машинного навчання та штучного інтелекту є неможливою.

Ключові слова: програмна система; транзакції; проектування; реалізація; навчання; тестування; модель; виявлення; протидія

Актуальність. Статистика показує, що проблема шахрайських транзакцій є актуальною на сьогоднішній день. Більш того, є актуальною проблема адаптації шахраїв для обходу існуючих програмних системи для виявлення шахрайських транзакцій. Тому є актуальною, реалізація програмної системи для виявлення шахрайських транзакцій за допомогою технологій машинного навчання, яка поєднає у собі максимальну кількість сильних сторін існуючих систем та мінімізує їх недоліки.

Мета роботи. Метою даної роботи є опис етапів програмної реалізації системи для виявлення шахрайських транзакцій за допомогою технологій машинного навчання та штучного інтелекту. Також, крім самої програмної реалізації, описуються підготовчі етапи без яких програмна реалізація подібної програмної системи неможлива.

Аналіз аналогічних програмних систем. Аналізуючи аналогічні програмні системи необхідно звертати увагу на такі метрики, як відсоток виявлених шахрайських транзакцій, відсоток помилкових спрацьовувань, здатність моделі адаптуватися для успішної протидії новим шахрайським схемам, наявність можливості налаштування системи правил, а також швидкість обробки транзакції. Відсоток виявлених шахрайських транзакцій відображає відсоток виявлених шахрайських транзакцій серед усіх шахрайських транзакцій. Відсоток помилкових спрацьовувань відображає відсоток нормальних транзакцій, які помилково були

позначені системою шахрайськими. Здатність моделі адаптуватися для протидії новим шахрайським схемам відображає можливість або неможливість моделі донавчатися для протидії новим шахрайським схемам. Наявність можливості налаштування системи правил відображає здатність або нездатність програмної системи до налаштування системи правил перевірки транзакцій. Швидкість обробки транзакцій відображає можливість або неможливість системи здійснювати обробку транзакцій у режимі реального часу.

Проектування архітектури програмної системи. Оптимальною, на мій погляд була б програмна система, яка поєднує у собі можливість перевірки транзакцій шляхом завантаження файлу з даними транзакцій для подальшої перевірки та можливість перевірки транзакцій за допомогою API. Для поєднання цих можливостей у одній програмній системі оптимальним рішенням є розробка веб-сервісу з формою для перевірки транзакцій у файлах, API для перевірки транзакцій за наявності згенерованого API ключа, а також системою правил перевірки транзакцій.

Програмна система буде містити наступні мікро-сервіси:

1. Система реєстрації та авторизації: Даний мікро-сервіси відповідає за реєстрацію та авторизацію користувачів у програмній системі.

2. Особистий профіль користувача: Даний мікро-сервіс відповідає за сторінку профілю користувача, редагування облікового запису та видалення облікового запису.

3. Перевірка транзакцій за допомогою форми: Даний мікро-сервіс відповідає за перевірку транзакцій за допомогою форми та формування звіту за результатами перевірки.

4. Робота з API ключами: Даний мікро-сервіс відповідає за генерацію API ключів, видалення API ключів, а також їх зберігання.

5. Перевірка транзакцій за допомогою API: Даний мікро-сервіс відповідає за перевірку транзакцій за допомогою API та формування звіту за результатами перевірки.

6. Система правил перевірки транзакцій: Даний мікро-сервіс відповідає за налаштування додаткових правил перевірки транзакцій. Система правил перевірки транзакцій робить перевірку більш зручною та гнучкою.

7. Адміністративна панель: Даний мікро-сервіс відповідає за модерацію програмної системи.

Порівняння алгоритмів машинного навчання та вибір найбільш ефективного або ефективних. Виявлення шахрайських транзакцій є задачею для рішення якої найкраще підходять алгоритми класифікації. Ефективність моделей машинного навчання для задач класифікації оцінюється наступними метриками [1]:

1. Точність (Accuracy [1,2]): Даний показник відображає частку всіх правильних класифікацій, як позитивних так і негативних. Розраховується за наступною формулою:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

де TP – True Positive;
TN – True Negative;
FP – False Positive;
FN – False Negative.

2. Повнота (Recall): Даний показник відображає частку всіх фактичних позитивних результатів, які були правильно класифіковані як позитивні. Розраховується за наступною формулою:

$$Recall = \frac{TP}{TP + FN}.$$

3. Коефіцієнт хибно позитивних результатів (False Positive Rate [1]): Даний показник відображає частку всіх фактичних негативних результатів, які були неправильно класифіковано як позитивні.

4. Розраховується за наступною формулою:

$$FPR = \frac{FP}{FP + TN}.$$

5. Влучність (Precision): Даний показник відображає частку всіх результатів, які були класифіковані моделлю як позитивні і які справді є такими. Розраховується за наступною формулою:

$$Precision = (TP)/(TP + FP).$$

6. F1-score: Даний показник відображає середнє гармонійне влучності й повноти. Розраховується за наступною формулою:

$$\text{F1} = (2TP)/(2TP + FP + FN).$$

Необхідно відмітити, що найбільш важливими для визначення ефективності виявлення шахрайських транзакцій є такі показники, як Recall та Precision. Recall відображає частку успішно виявлені шахрайські транзакції серед усіх транзакцій, Precision відображає частку операцій відмічених системою шахрайськими та які дійсно є такими. Показник Recall безумовно є більш приоритетним у зв'язку з тим, що пропуск шахрайської транзакції значно гірше помилкового блокування нормальної транзакції.

TP (True Positive), FP (False Positive), FN (False Negative) та TN (True Negative) є складовими матриці помилок, яка є інструментом оцінки ефективності моделі машинного навчання. Приклад матриці помилок зображено у вигляді таблиці 1.

Таблиця 1. **Приклад матриці помилок**

	Фактично позитивні	**Фактично негативні**
Прогнозовані позитивні	Істинно позитивний результат (TP). Модель правильно передбачила позитивний клас. Приклад: модель виявила шахрайську транзакцію і вона дійсно є такою	Хибно позитивний результат (FP). Модель неправильно передбачила позитивний клас. Приклад: модель виявила, що транзакція є шахрайською, але насправді це не так
Прогнозований негативний клас	Хибно негативний результат (FN). Модель неправильно передбачила негативний клас. Приклад: модель виявила, що транзакція не є шахрайською, хоча насправді вона шахрайська	Істинно негативний результат (TN). Модель правильно передбачила негативний клас. Приклад: модель виявила, що транзакція є нормальною і вона дійсно є такою

Навчання та тестування моделі. Даний етап включає: підготовку набору даних до навчання моделі, налаштування моделі машинного навчання, навчання моделі, а також її тестування.

Невід'ємною частиною підготовки набору даних до навчання моделі є аналіз даних, який включає в себе перевірку форми набору даних, перевірки типів даних кожної колонки, перевірки кожної колонки на наявність пропущених значень.

У випадку виявлення пропущених значень у наборі даних, їх необхідно замінити на середнє для колонки значення або видалити рядки з пропущеними значеннями. Наявність пропущених значень у наборі даних може негативно вплинути на результати навчання моделі, тому даний крок є дуже важливим.

Після перевірки набору даних його необхідно розділити на тренувальну та тестову вибірки у пропорції 80 та 20 або 70 та 30 відсотків відповідно.

Після розділення набору даних на тренувальну та тестову вибірки необхідно масштабувати ознаки. Масштабування ознак допомагає підвищити точність та ефективність моделей, які залежать від масштабу даних. Масштабування ознак необхідно проводити окремо для тренувальної та тестової вибірок.

Після масштабування ознак наступним кроком підготовки набору даних до навчання є балансування класів. Балансування класів використовується в тих випадках, коли один клас значно переважає інший. Балансування класів застосовується тільки для тренувальної вибірки.

Налаштування моделі для подальшого навчання залежить від алгоритму. Наприклад, для таких алгоритмів, як GradientBoostingClassifier, DecisionTreeClassifier, RandomForestClassifier та LogisticRegression важливо фіксувати випадковість при розбитті даних за допомогою параметру random_state [3]. Це гарантує однакове розбиття на тренувальну та тестову вибірки при кожному запуску.

Порівняння таких алгоритмів, як GradientBoostingClassifier, DecisionTreeClassifier, RandomForestClassifier та LogisticRegression, а також алгоритму KNeighboursClassifier навчати який можна без додаткових налаштувань, виявило, що GradientBoostingClassifier є найбільш ефективним [9]. Показник Recall алгоритму GradientBoostingClassifier склав 0.8776, тобто виявляє 87,76 % усіх шахрайських транзакцій було успішно виявлено. Показник Precision алгоритму GradientBoostingClassifier склав 0.7890, тобто 78,90 % транзакцій позначених моделлю шахрайськими дійсно є такими. Дані показники, особливо показник Recall потребує підвищення. Досягти цього можливо за допомогою ансамблювання алгоритмів.

Ансамблювання алгоритмів. Ансамблювання алгоритмів машинного навчання являє собою об'єднання двох або більше алгоритмів з метою отримання більш точних передбачень. Було проведено експеремент у рамках якого за допомогою класу VotingClassifier Python бібліотеки scikit-learn було поєднано алгоритм GradientBoostingClassifier з такими алгоритмами, як RandomForestClassifier, LGBMClassifier, XGBClassifier, а також BalancedBaggingClassifier [10, 11, 12]. За результатом ансамблювання даних алгоритмів показник Recall моделі машинного навчання склав 0.9081, а показник Precision – 0.4635. Тобто, вдалося підвищити критично важливий показник Recall з 0.8776 до 0.9081. Крім того, Recall та Precision можуть бути додатково підвищені за допомогою системи правил та підбору кращих параметрів для навчання ансамблю алгоритмів.

При поєднанні моделі, яка містить подібний ансамбль алгоритмів з поясненням рішень моделі та системою правил дозволить створити програмну систему:

З високим показником Recall, підвищеним не тільки ансамблюванням алгоритмів машинного навчання, а й системою правил перевірки транзакцій. Система правил перевірки транзакцій та підбір кращих параметрів для навчання ансамблю алгоритмів забезпечить значне зростання Recall та Precision до 0.95+ та 0.80+ відповідно. Програмна система з такими показниками Recall та Precision не тільки не поступається, ай перевершує оціночні аналогічні показники аналогічних систем. Крім того, наявність системи правил робить програмну систему більш гнучкою порівняно з аналогами, які подібної системи не мають..З можливістю отримання аргументованого пояснення щодо рішення моделі;

З можливістю донавчання та як наслідок адаптації моделі для протидії новим шахрайським схемам.

З двома інтерфейсами замість одного, а саме інтерфейсом для перевірки транзакцій за допомогою форми, а також інтерфейсом для перевірки транзакцій за допомогою API. У свою чергу, такі аналогічні програмні системи як FICO Falcon Fraud Manager, ACI Worldwide Fraud Management та ARIC Featurespace можливості перевірки транзакцій за допомогою форми.

Вибір стеку технологій для програмної реалізації. На даному етапі обирається стек технологій для програмної реалізації системи. Оптимальним, на мій погляд, рішенням для програмної реалізації подібної системи, на мій погляд, є Python фреймворки Django, Django Rest Framework, а також СКБД PostgreSQL [4, 5, 6]. Для асинхронної обробки великої кількості транзакцій можна використати фреймворк для асинхронного виконання задач Celery та in-memory базу даних Redis [7, 8].

Фреймворк Django являє собою високорівневий фреймворк, призначений для веб-розробки.

Даний фреймворк має такі переваги.

1. Вбудована адміністративна панель: Фреймворк Django має вбудовану адміністративну панель, яка дозволяє зручно керувати даними. Серед можливостей, які надає дана адміністративна панель варто відмітити створення записів, редагування записів, перегляд записів, видалення записів, пошук записів та фільтрування записів. Також, варто відмітити можливість налаштування відображуваних полів таблиці, а також полів за якими відбувається пошук та фільтрування записів.

2. ORM для роботи з БД: ORM (Object Relation Mapping) робить можливим написання запитів для БД без використання SQL. Серед переваг Django ORM варто відмітити такі, як сумісність з різними СКБД, універсальність запитів для різних СКБД, можливість оптимізації запитів, інтеграція з системою міграцій, захист від SQL injections, а також можливість написання SQL запиту.

3. Модульна архітектура: Фреймворк Django має модульну архітектуру, що значно спрощує масштабування проекту. Також, варто відмітити, що проекти у Django базуються на патерні MVT. У патерні MVT (Model View Template) компонент «Model» відповідає за роботу з даними, компонент «View» - за обробку запитів від користувачів, компонент «Template» - за відображення даних.

4. Система міграцій: Система міграцій у Django дозволяє керувати змінами у схемі бази даних з можливістю повернутися до однієї з попередніх версій схеми бази даних, якщо це необхідно.

5. Вбудована система автентифікації: Django надає користувачам вбудовану систему автентифікації, яка включає стандартну модель User, можливість розширення стандартної моделі User, автентифікацію, авторизацію, а також забезпечує зв’язок користувача з запитом. Крім того, вбудована система автентифікації Django надає стандартну форму авторизації, стандартну форму реєстрації, стандартну форму зміни пароля, а також стандартну форму скидання пароля. Також, варто відмітити наявність вбудованих Class-Based Views для входу до облікового запису, виходу з облікового запису, зміни пароля, скидання пароля, а також декораторів, які забезпечують обмеження доступу до функції або Class-Based Views.

6. Зручна обробка форм: Фреймворк Django надає вбудовані класи для обробки форм, клас Form – для простих форм, які не пов’язані з моделями та клас ModelForm – для для форм, які пов’язані з моделями. Варто відмітити можливість валідації форм, як на рівні окремих полів, так в на рівні всієї форми. Також, надається можливість детального налаштування будь-якого поля форми за допомогою віджетів. Крім того, передбачено вбудований шаблонний тег, який забезпечує захист від CSRF-атак.

7. Class-Based Views: Class-Based Views являють собою класи, які наслідуються від стандартного класу View та описують обробку запитів за допомогою методів. У Django реалізовано такі стандартні Class-Based Views, як TemplateView - для відображення шаблону, ListView – для обробки списку об’єктів, DetailView - для обробки об’єкту, CreateView - для створення об’єкту, UpdateView – для редагування об’єкту, DeleteView – для видалення об’єкту. Крім того, є можливість додавати додаткові налаштування Class-Based Views за допомогою Mixin класів, наприклад, обов’язкова авторизація для використання.

8. Вбудована пагінація: Класи Paginator та Page значно спрощують реалізацію пагінації у Django. Клас Paginator приймає послідовність об’єктів та кількість об’єктів на сторінці. Клас Page містить список об’єктів сторінки, номер сторінки, інформацію про наявність або відсутність наступної або попередньої сторінки, а також індекси об’єктів списку.

Фреймворк Django Rest Framework сумісний з фреймворком Django, він є оптимальним вибором для реалізації API частини програмної системи. Django Rest Framework має такі переваги.

1. Система сериалізації: Система сериалізації у Django Rest Framework дозволяє працювати з моделями, виконувати валідацію даних, а також конвертувати моделі Django у JSON та JSON у моделі Django.

2. Вбудована система автентифікації: Вбудована система автентифікації Django Rest Framework сумісна з Django, але має свої особливості. Дана система містить такі стандартні

класи, як BasicAuthentication - передає логін та пароль у кожному запиті, SessionAuthentication - є аналогом стандартного класу LoginView у Django, TokenAuthentication - відповідає за унікальний токен користувача. Крім того, Django Rest Framework передбачає можливість реалізації кастомного класу автентифікації.

3. Робота з різними форматами: Django Rest Framework підтримує роботу з різними форматами. Серед цих форматів – JSON, HTML, multipart/form-data, application/x-www-form-urlencoded, XML та YAML. Для роботи з форматом JSON використовуються класи JSONRenderer та JSONParser, для роботи з форматом HTML – веб-інтерфейс BrowsableAPIRenderer, для роботи з формами – класи FormParser та MultiPartParser, для роботи з XML – класи XMLRenderer та XMLParser, для роботи з YAML – YAMLRenderer. Для XML та YAML необхідно підключити rest_framework_xml та rest_framework_yaml у settings.py, за замовчуванням вони не підключені. Крім того, є можливість реалізації кастомних класів для інших форматів.

4. Class-Based Views: Class-Based Views у Django Rest Framework базуються на класі APIView, який є аналогом класу View у Django. Також, передбачені стандартні класи для базових операцій. Клас ListAPIView - для роботи зі списком об’єктів, клас CreateAPIView - для створення об’єкту, клас UpdateAPIView - для редагування об’єкту, клас DestroyAPIView - для видалення об’єкту, клас RetrieveAPIView - для отримання об’єкту.

5. Вбудоване фільтрування: Django Rest Framework має вбудований пошуковий фільтр SearchFilter, призначений для текстових полів. Крім того, завдяки класу FilterSet можна реалізувати кастомний фільтр.

6. Вбудоване сортування: Вбудований клас OrderingFilter дозволяє сортування за вказаними полями.

7. Вбудована пагінація: Django Rest Framework передбачає вбудоване розбиття результатів API запиту на сторінки. Можливі різні такі варіанти пагінації, як PageNumberPaggination - посторінкова пагінація, LimitOffsetPaggination – з можливістю вказати зміщення та кількість об’єктів, а також CursorPaggination – курсорна пагінація, яка базується на значенні унікального поля.

Серед основних переваг СКБД PostgreSQL варто відмітити.

1. Відповідність ACID: ACID являє собою набір властивостей транзакцій, які гарантують надійність при роботі з даними. A (Atomicity) – у транзакції виконуються всі операції або не одна, C (Consistency) - дані зберігають допустимий стан за результатами транзакції, I (Isolation) - ізольованість транзакцій робить можливими одночасні транзакції, D (Durability) – після підтвердження транзакції дані гарантовано зберігаються.

2. Швидкість роботи: Однією з головних переваг СКБД PostgreSQL є висока швидкість роботи;

3. Розширюваність: PostgreSQL надає можливість створення кастомних типів даних, операторів, агрегатних функцій, алгоритмів індексації. Крім того, PostgreSQL підтримує створення функцій на різних мовах програмування.

4. Підтримка індексів: PostgreSQL підтримує індекси. Підтримуються такі індекси, як B-tree – стандартний, Hash – для рівності, GIN (Generalized Inverted Index) – для швидкого пошуку у масивах, BSON та повнотекстового пошуку, GiST – для геометричних задач, BRIN (Block Range Index) – для великих за об’ємом таблиць.

5. Безпечність: SSL/TLS забезпечує безпеку з’єднань. Рольова модель забезпечує розмежування прав користувачів.

6. Відмовостійкість: Відмовостійкість забезпечується наявністю таких механізмів, як асинхронна реплікація, PITR (Point-in-Time Recovery), WAL (White-Ahead Log), а також кластеризація та шардування. Асинхронна реплікація – гарантує збереження даних на двох серверах, PITR (Point-in-Time Recovery) – дозволяє відновити стан бази даних на певний момент часу, WAL (White-Ahead Log) – зміни стану бази даних фіксуються у журналі, кластеризація та шардування – забезпечують можливість створення масштабних кластерів.

7. Підтримка JSONB: Підтримка формату JSONB (JSON Binary) надає можливість працювати з документами, а рівно використовувати PostgreSQL, як гібрид SQL та NoSQL. Важливо відмітити, що формат JSONB підтримує швидкий пошук та індексацію;

Серед переваг фреймворку Celery варто відмітити [7]:

1. Масштабованість: Celery передбачає можливість додавання нових воркерів для обробки задач, що робить Celery ефективним для мікро-сервісної архітектури.

2. Асинхронність: Фреймворк Celery [7] асинхронно виконує задачі у фоні, що дозволяє не блокувати роботу програмної системи та позитивно впливає на продуктивність.

3. Планування задач: Вбудований планувальник Celery Beat надає можливість виконання задач за розкладом.

4. Надійність: Надійність Celery забезпечується повторним виконанням задач у випадку виникнення помилок.

5. Висока продуктивність: Асинхронність виконанная задач, а також виконання у RAM забезпечує високу продуктивність фреймворку Celery.

6. Підтримка різних брокерів: Celery підтримує взаємодії з різними брокерами, такими, як Redis [8], Amazon SQS, RabbitMQ.

7. Можливість інтеграції з різними фреймворками: Celery надає можливість інтеграції з такими фреймворками, як Django, Flask та FastAPI.

Серед переваг Redis варто відмітити [8].

1. Надійність та відмовостійкість: Реплікація та кластеризація забезпечує відмовостійкість Redis.

2. Швидкість роботи: Зберігання задач у оперативній пам’яті забезпечує високу швидкість роботи.

3. Підтримка асинхронних задач: Redis підтримує асинхронне виконання задач.

4. Масштабованість: Redis підтримує кластери та шардування.

5. Можливість інтеграції з різними фреймворками: Redis підтримує інтеграцію з такими фреймворками, як Django, Flask та FastAPI. Також, Redis часто використовується разом з Celery.

Основні кроки програмної реалізації. Серед основних кроків програмної реалізації для виявлення шахрайських транзакцій варто відмітити такі, як інтеграція моделі у веб-сервіс, а також реалізація основних компонентів системи. Основні компоненти програмної системи - мікро-сервіс для реєстрації та авторизації, мікро-сервіс для особистого профілю користувача, мікро-сервіс для перевірки транзакцій за допомогою форми, мікро-сервіс для роботи з API ключами, мікро-сервіс для перевірки транзакцій за допомогою API, мікро-сервіс для керування системою правил перевірки транзакцій, а також мікро-сервіс для керування адміністративною панеллю.

Мікро-сервіс для реєстрації та авторизації містить модель для зберігання користувачів, CBV для реєстрації у системі, CBV для авторизації у системі. Крім того, даний мікро-сервіс містить ModelForm для реєстрації, а також ModelForm для авторизації.

Мікро-сервіс для особистого профілю користувача містить CBV для зміни паролю користувача, CBV для редагування облікового запису та CBV для видалення облікового запису. Крім того, даний мікро-сервіс містить PasswordChangeForm для зміни паролю, а також ModelForm для редагування облікового запису.

Мікро-сервіс для перевірки транзакцій за допомогою форми містить модель для зберігання результатів перевірок, CBV для перевірки транзакцій за допомогою форми та CBV для формування звіту за результатами перевірки. Крім того, даний мікро-сервіс містить ModelForm для перевірки транзакцій за допомогою форми.

Мікро-сервіс для роботи з API ключами містить модель для зберігання API ключів, CBV для генерації ключів, CBV для видалення API ключів, ModelForm для генерації API ключів.

Мікро-сервіс для перевірки транзакцій за допомогою API містить ендпоінт для перевірки однієї транзакції, а також окремий ендпоінт для перевірки набору транзакції.

Мікро-сервіс для керування системою правил перевірки транзакцій містить модель для зберігання правил перевірки транзакцій, CBV для створення правил перевірки транзакцій, CBV для редагування правил перевірки транзакцій, CBV для видалення правил перевірки транзакцій, CBV для перегляду правил перевірки транзакцій. Крім того, даний мікро-сервіс містить ModelForm для створення та редагування правил перевірки транзакцій.

Мікро-сервіс для керування адміністративною панеллю містить CBV для перегляду списку всіх користувачів, CBV для створення користувача з можливістю вибору типу користувача, CBV для редагування користувача з можливістю вибору типу користувача, CBV для перегляду даних користувача, CBV для перегляду списку всіх перевірок, CBV для перегляду перевірки, CBV для перегляду списку всіх API ключів, CBV для видалення API ключів. Крім того, даний мікро-сервіс містить ModelForm для створення та редагування користувача з можливістю вибору типу користувача.

Для інтеграції моделі у програмну систему оптимальним рішенням є створення окремого мікро-сервісу. Подібний підхід надає можливість версіонування - оновлення моделі без порушення роботи основного коду програмної системи. Функціями даного мікро-сервісу є завантаження моделі, можливість оновлення моделі, взаємодія з ендпоінтами API, а також передобробка даних.

Висновки. Було описано етапи програмної реалізації програмної системи для виявлення шахрайських транзакцій. Описано головні метрики, на які необхідно звертати увагу при аналізі аналогічних програмних систем, а саме: відсоток виявлених шахрайських транзакцій, відсоток помилкових спрацьовувань, здатність моделі адаптуватися для успішної протидії новим шахрайським схемам, наявність можливості налаштування системи правил, а також швидкість обробки транзакції. Крім того, описано проектування архітектури програмної системи, порівняння алгоритмів машинного навчання та вибір найбільш ефективного або ефективних, аргументовано важливість метрик Recall та Precision, описано основні кроки навчання та тестування моделі, описано вибір стеку технологій для програмної реалізації, переваги та особливості обраних для програмної реалізації технологій, а також основні кроки програмної реалізації системи. Також, було проведено експеримент з ансамблюванням алгоритмів машинного навчання.

СПИСОК ЛІТЕРАТУРИ

1. “Classification: Accuracy, recall, precision, and related metrics – Machine Learning”. – URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall (дата звернення: 18.09.2025).

2. “Thresholds and the confusion matrix”. – URL: https://developers.google.com/machine-learning/crash-course/classification/thresholding#confusion_matrix (дата звернення: 18.08.2025).

3. “scikit-learn Machine Learning in Python”. – URL: https://scikit-learn.org/stable/ (дата звернення: 18.09.2025).

4. “Django documentation”. – URL: https://docs.djangoproject.com/en/5.2/ (дата звернення: 18.09.2025).

5. “Quickstart – Django REST framework”. – URL: Режим доступу: https://www.django-rest-framework.org/tutorial/quickstart/ (дата звернення: 18.08.2025).

6. “PostgreSQL Documentation”. – URL: https://www.postgresql.org/docs/ (дата звернення: 18.08.2025).

7. “Celery – Distributed Task Queue”. – URL: https://docs.celeryq.dev/en/stable/ (дата звернення: 18.08.2025).

8. “Redis Docs”. – URL: https://redis.io/docs/latest/ (дата звернення: 18.08.2025).

9. Матеріали П'ятнадцятої Міжнародної наукової конференції студентів та молодих вчених «Сучасні інформаційні технології – 2025». *Одеса : Наука і техніка*. 2025.

10. “LightGBM’s documentation!”. – URL: https://lightgbm.readthedocs.io/en/stable/ (дата звернення: 25.08.2025).

11. “XGBoost Documentation”. – URL: https://xgboost.readthedocs.io/en/stable/ (дата звернення: 25.09.2025).

12. “imbalanced-learn documentation”. – URL: https://imbalanced-learn.org/stable/ (дата звернення: 25.09.2025).

DOI: https://doi.org/10.15276/ict.02.2025.49
UDC 004.8:004.75:004.056.5

Web service for detecting fraudulent transactions using artificial intelligence technologies

Mykhaylo A. Shylin[1)]
Student of the Department of Software Engineering
ORCID: https://orcid.org/0009-0003-6527-9356; 10387444@stud.op.edu.ua
Oleksandr S. Prygozhev[1)]
PhD, Associate Professor of the Department of Software Engineering
ORCID: https://orcid.org/0000-0001-8532-8897; o.s.prygozhev@op.edu.ua
[1)] Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine

ABSTRACT

These theses describe the stages of software implementation of a software system for detecting fraudulent transactions using machine learning and artificial intelligence technologies. The description of the stages of software implementation of this system includes: analysis of similar software systems, design of the software system architecture, comparison of machine learning algorithms and selection of the most effective or effective, training and testing of the model, selection of a technology stack for software implementation, as well as the main steps of software implementation. The stage "analysis of similar software systems" contains a detailed description of the metrics, characteristics and features that should be paid attention to when implementing a similar system. This stage helps to identify the strengths and weaknesses of similar systems. The stage "design of the software system architecture" contains a description of the design of the software system architecture with a justification of the effectiveness of this architecture. The stage "comparison of machine learning algorithms and selection of the most effective or effective" contains a description of existing metrics of the effectiveness of machine learning models, as well as the most important metrics of effectiveness specifically for those models that are designed to detect fraudulent transactions. The stage "model training and testing" contains a description of the preparation of the dataset for model training, the design of the machine learning model, the training of the model, and its testing. The stage "technology stack selection for software implementation" contains a description of the selection of a technology stack for software implementation of a software system for detecting fraudulent transactions with a justification for why this particular technology stack was chosen. The stage "basic steps of software implementation" contains a description of the basic steps of software implementation of such a software system. Design and software implementation of a software system for detecting fraudulent transactions is relevant today, since the problem of transaction fraud and the adaptation of fraudsters to existing software systems designed to combat transaction fraud is relevant. The purpose of writing these theses is to describe the stages of software implementation of a system for detecting fraudulent transactions using machine learning and artificial intelligence technologies, which would compare the strengths of existing similar systems and minimize their shortcomings. Also, the theses contain a description of the preparatory actions without which the development of a software system for detecting fraudulent transactions using machine learning and artificial intelligence technologies is impossible.

Keywords: Software system; transactions; design; implementation; training; testing; model; detection; countermeasure