Государственное бюджетное общеобразовательное учреждение города Москвы «Школа № 1580»

Прогнозирование и оценка ценности Telegram NFT на основе его характеристик

Участники:

Ученик 10 «Т» класса ГБОУ Школа №1580 Лазаренко Алексей Андреевич

Ученик 10 «Т» класса ГБОУ Школа №1580 Тюриков Алексей Михайлович тб

Руководитель:
Педагог дополнительного образования
ФГАОУ ВО МГТУ им. Н.Э. Баумана
Узденов Таулан Тельманович

1. Введение

Текущие тенденции в цифровой сфере демонстрируют всё более глубокую интеграцию цифровых ресурсов и цифровых валют в повседневную жизнь современного общества.

Многие ведущие мировые державы активно формируют законодательные и технологические рамки, призванные регулировать этот стремительно развивающийся сектор.

Современные платформы и приложения, с которыми мы взаимодействуем ежедневно, также эволюционируют, внедряя новые технологии и концепции, соответствующие духу времени.

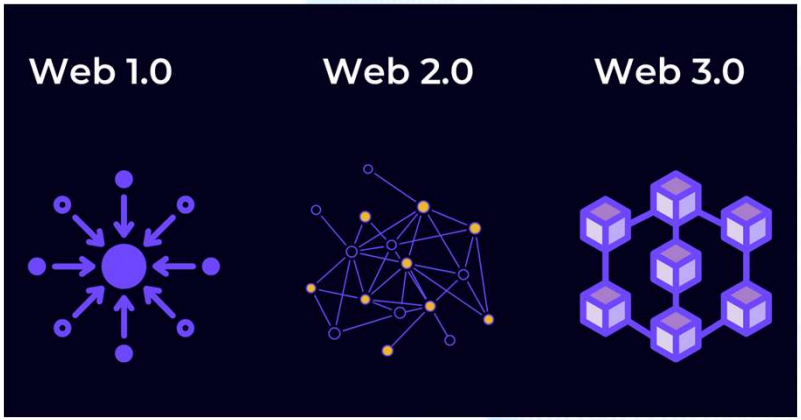
Цель: анализе устройства WEB-3, особенностей Telegram NFT и среды TON, в которой они существуют, а также в разработке модели машинного обучения для прогнозирования их стоимости и интерфейса для неё в виде бота в мессенджере Telegram.

Задачи:

1. Определение сущности и принципов технологии WEB-3, изучение уникальных особенностей блокчейна TON;
2. Рассмотрение отличительных признаков Telegram NFT подарков;
3. Создание отражающего специфику рынка датасета для дальнейшего обучения моделей.
4. Разработка, тестирование и оценка эффективности нескольких моделей машинного обучения.
5. Интеграция финальной модели машинного обучения в Telegram бота

2. WEB-3 и Telegram NFT

Концепция WEB-3 возникла как ответ на ограниченности предыдущих поколений интернета, прежде всего связанных с централизованным контролем.



Одна из центральных технологий WEB-3: блокчейн. Блокчейн представляет собой распределённую и децентрализованную технологию хранения данных, основанную на последовательной записи информации в виде связанных между собой блоков.

The Open Network (TON) — это децентрализованная блокчейн-платформа, созданная командой мессенджера Telegram

Некоторые преимущества TON над другими блокчейнами:

- Шардинг – блокчейн подстраивается под нагрузку и всегда эффективен
- Proof of Stake – энергоэффективный метод проведения транзакций
- Самовосстановление узлов – блокчейн быстрее восстанавливается после кибератак и сбоев
- Блокчейн поддерживает создание децентрализованных приложений
- Система доменных имён вместо сложных криптографических адресов



Важной инновацией Телеграма стало внедрение NFT-подарков в рамках мессенджера. Пользователи могут их покупать, продавать и передавать друг другу – это так же просто, как отправить сообщение.

У каждого подарка существует свой собственный URL формата http://t.me//НазваниеКоллекции-НомерПодарка, а также необходимые нам характеристики редкости: саплай, редкость модели и редкость фоноввых элементов.



3. Продукт

Используемые библиотеки:

- NumPy —для сложных математических функций
- Pandas — для работы с датасетами.
- Matplotlib.pyplot — для построения графиков.
- Seaborn — для визуализации специфических графиков.
- Scikit-learn (sklearn) — для моделей машинного обучения.
- Telebot — для работы с TG Ботом.

Разработка проходила в среде Google Collab: https://colab.research.google.com/drive/115U7DI-atmRTTx0gvZ9MJgQt-SDpNGrY

3.1 Датасет

Создание корректного, структурированного и репрезентативного набора данных является базовым и одним из наиболее критичных этапов любого проекта, связанного с машинным обучением. К сожалению, мы не нашли датасета для нашей темы в интернете и создали его самостоятельно.

Мы решили вручную создать датасет размером в тысячу строк, содержащий информацию о двухстах токенов пяти коллекций разных ценовых категорий – от стоящих несколько сот рублей Light Sword до многомиллионных Plush Pepe.

Получившийся датасет также был разделён на тренировочные датасеты в соотношении train/test 70 на 30 и 80 на 20.

После разработки датасета он был загружен в Google Collab:

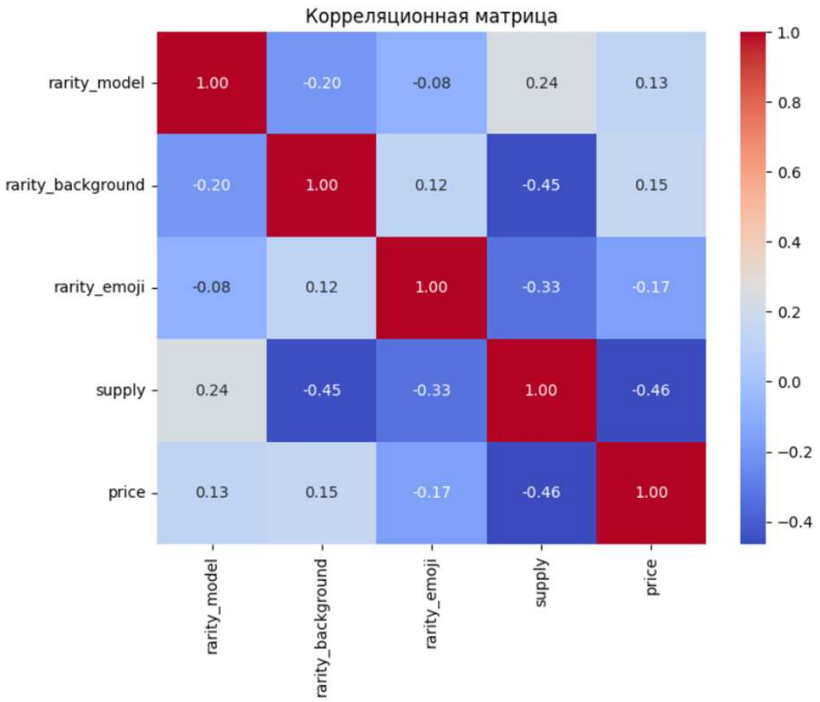
```
from google.colab import drive
drive.mount('/content/drive')
```

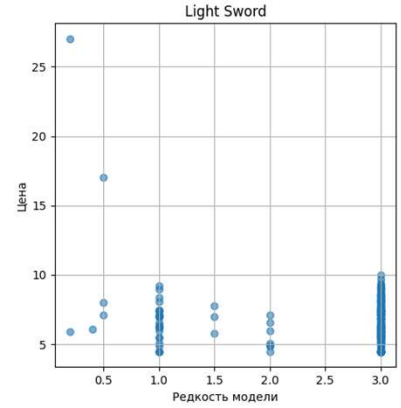
```
Mounted at /content/drive
```

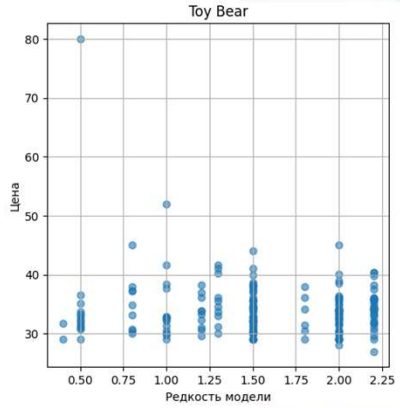
```
# В этом блоке происходит загрузка датасетов с Гугл Диска для дальнейшего использования

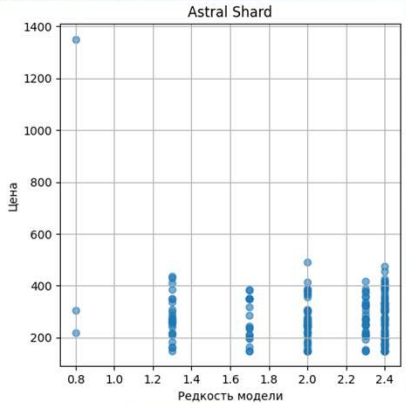
dataset_full = pd.read_csv("/content/drive/MyDrive/nft_dataset_final_.csv")
dataset20 = pd.read_csv("/content/drive/MyDrive/nft_dataset_20_.csv")
dataset30 = pd.read_csv("/content/drive/MyDrive/nft_dataset_30_.csv")
dataset70 = pd.read_csv("/content/drive/MyDrive/nft_dataset_70_.csv")
dataset80 = pd.read_csv("/content/drive/MyDrive/nft_dataset_80_.csv")
```

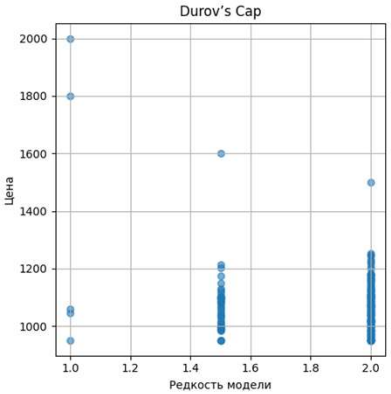
А затем была выведена статистика по датасету:

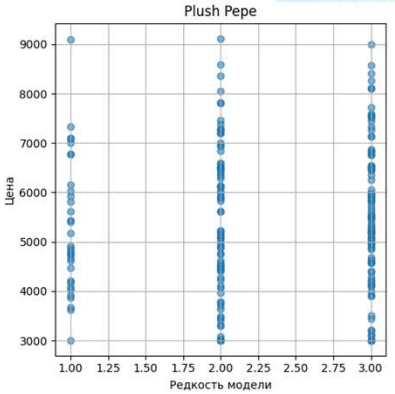












3.2 Алгоритмы машинного обучения.

На начальном этапе моделирования была реализована линейная регрессия, выступающая в роли базовой отправной точки для построения предсказательной системы.

```
featur = ['rarity_model', 'rarity_background', 'rarity_emoji', 'supply']

X_train = dataset70[featur]
y_train = dataset70['price']
X_test = dataset30[featur]
y_test = dataset30['price']

# Используем функцию LinearRegression() из библиотеки sklearn

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

Метрики модели:

MAE: 1272.873

RMSE: 1688.272

R^2: 0.450

Далее был создан график фактических/предсказанных значений:



Следующим этапом построения предсказательной модели явилась реализация регрессии Lasso() с дополнительными инструментами регуляризации – скалирование (для более «честной» обработки признаков) и логарифмирование целевой переменной (для снижения воздействия предполагаемых выбросов)

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

y_train_log = np.log1p(y_train)
y_test_log = np.log1p(y_test)

lasso = Lasso(alpha=0.1, random_state=42)
lasso.fit(X_train, y_train_log)

y_pred_log = lasso.predict(X_test)

y_pred = np.expm1(y_pred_log)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

Метрики новой модели:

MAE: 1251.323

RMSE: 2321.397

R^2: 0.054

Далее вновь был создан график фактических/предсказанных значений:



Заключительным и наиболее результативным этапом моделирования стало обучение ансамблевой модели Random Forest Regressor — алгоритма, функционирующего на основе множества независимых решающих деревьев.

```
featur = ['rarity_model', 'rarity_background', 'rarity_emoji', 'supply']

X_train = dataset70[featur]
y_train = dataset70['price']
X_test = dataset30[featur]
y_test = dataset30['price']

rf_model = RandomForestRegressor(
    n_estimators=200,      # количество деревьев
    max_depth=8,           # ограничение глубины против переобучения
    random_state=42
)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.3f}")
print(f"RMSE: {rmse:.3f}")
print(f"R^2: {r2:.3f}")
```

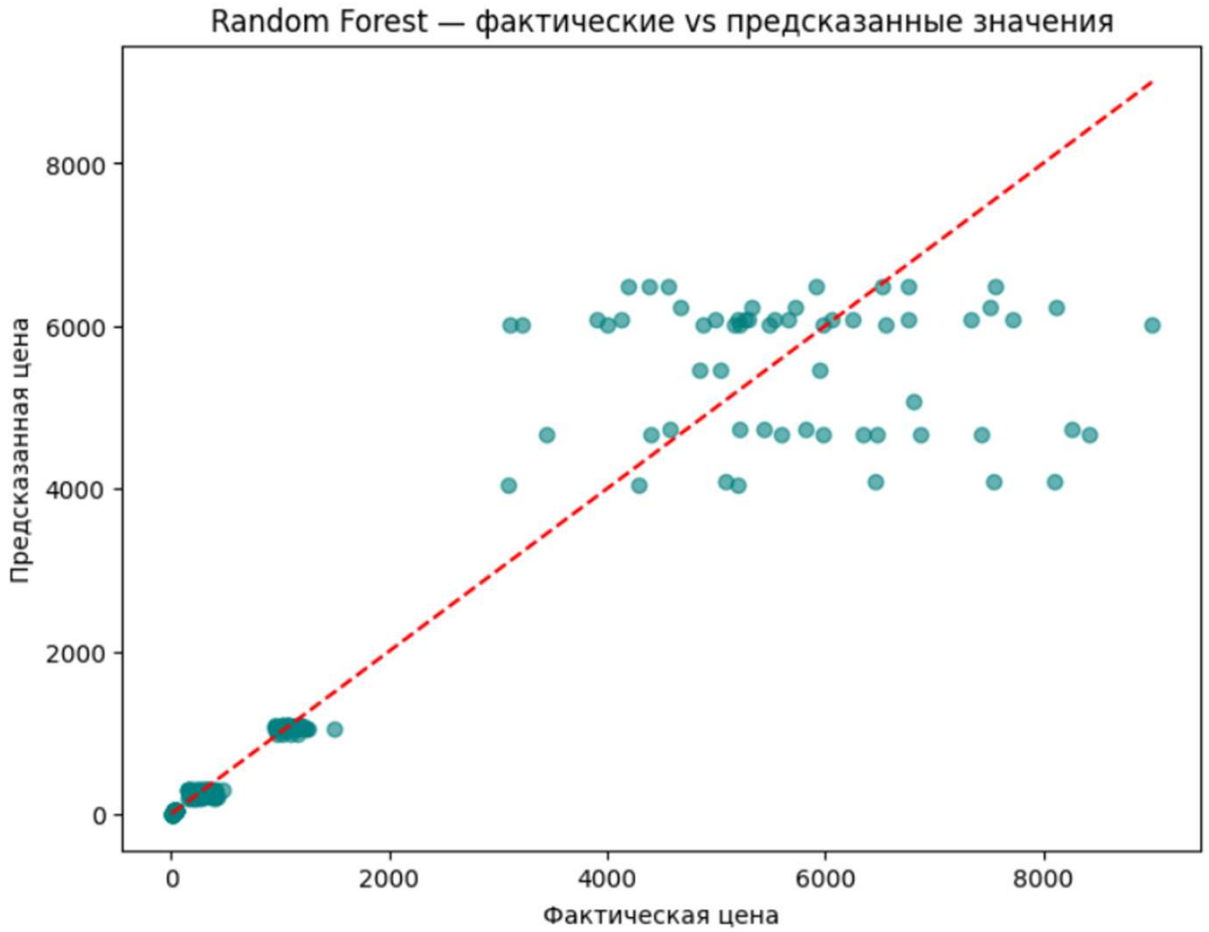
Метрики финальной модели:

MAE: 296.141

RMSE: 744.187

R^2: 0.893

Последний график фактических/предсказанных значений:



Пошаговое тестирование моделей показало следующую логическую динамику:

- LinearRegression обеспечила базовое представление о структуре данных и сформировала отправную точку для оценки качества;
- Lasso Regression при подобранных параметрах продемонстрировала существенное ухудшение метрик, что побудило нас отойти от линейной модели;
- Random Forest показал значительное улучшение по всем ключевым метрикам, что подтвердило корректность выбора ансамблевой стратегии для задачи прогнозирования цен NFT.

Таким образом, окончательный выбор модели — Random Forest — был сделан на основе сравнения и анализа поведения алгоритмов на реальных данных: ансамблевый метод обеспечил требуемую гибкость и высокую точность предсказаний.

3.3 Интерфейс: TG-бот

На заключительном этапе практической части проекта реализована интеграция обученной модели машинного обучения в пользовательский интерфейс в виде Telegram-бота. Такая реализация обеспечивает упрощение взаимодействия пользователя с системой прогнозирования и делает инструмент доступным и удобным для оперативного применения пользователями в реальном времени.

Архитектура бота логически разделена на шесть функциональных блоков: импорт необходимых библиотек и создание экземпляра интерфейса, настройка уникального идентификатора бота, объявление глобальных параметров — признаков NFT, используемых моделью для прогноза, и создание словарей для удобной навигации по коллекциям и возможным вариантам атрибутов.

Сначала через библиотеку telebot инициируется сам бот. Ему присваивается уникальный и собственный тег. Затем создаются главные переменные, в которые будут записаны параметры пользовательского NFT:

```
# t.me/NFT_price_predictor_bot
token = '8211214092:AAFbKd5Mv2e7t9aX6eMIEndY7GsBRyb0ZN4'
bot = telebot.TeleBot(token)
keyboard = types.ReplyKeyboardRemove()
```

```
bot_nft_name = ""
bot_nft_model = ""
bot_nft_rarity_model = 0
bot_nft_bg = ""
bot_nft_rarity_bg  = 0
bot_nft_emoji = ""
bot_nft_rarity_emoji = 0
bot_nft_supply = 0
```

Для удобства пользователя создаются словари, содержащие доступные модели коллекции, цвета фонов и варианты эмодзи:

```
names_dict = {"Light Sword": 131222, "Toy Bear": 57724, "Astral Shard": 6196, "Durov's Cap": 4774, "Plush Pepe": 2861}

models_dict = {"Light Sword": dataset_full[dataset_full["nft_name"] == "Light Sword"]["model"].unique(),
               "Toy Bear": dataset_full[dataset_full["nft_name"] == "Toy Bear"]["model"].unique(),
               "Astral Shard": dataset_full[dataset_full["nft_name"] == "Astral Shard"]["model"].unique(),
               "Durov's Cap": dataset_full[dataset_full["nft_name"] == "Durov's Cap"]["model"].unique(),
               "Plush Pepe": dataset_full[dataset_full["nft_name"] == "Plush Pepe"]["model"].unique()}

bgs_dict = {"Light Sword": dataset_full[dataset_full["nft_name"] == "Light Sword"]["background"].unique(),
            "Toy Bear": dataset_full[dataset_full["nft_name"] == "Toy Bear"]["background"].unique(),
            "Astral Shard": dataset_full[dataset_full["nft_name"] == "Astral Shard"]["background"].unique(),
            "Durov's Cap": dataset_full[dataset_full["nft_name"] == "Durov's Cap"]["background"].unique(),
            "Plush Pepe": dataset_full[dataset_full["nft_name"] == "Plush Pepe"]["background"].unique()}

emojis_dict = {"Light Sword": dataset_full[dataset_full["nft_name"] == "Light Sword"]["emoji"].unique(),
               "Toy Bear": dataset_full[dataset_full["nft_name"] == "Toy Bear"]["emoji"].unique(),
               "Astral Shard": dataset_full[dataset_full["nft_name"] == "Astral Shard"]["emoji"].unique(),
               "Durov's Cap": dataset_full[dataset_full["nft_name"] == "Durov's Cap"]["emoji"].unique(),
               "Plush Pepe": dataset_full[dataset_full["nft_name"] == "Plush Pepe"]["emoji"].unique()}
```

Затем прописывается логика взаимодействия бота с пользователем через сообщения:

```
@bot.message_handler(commands = ["start"])
def start(message):
    bot.send_message(message.from_user.id, "Здравствуйте! Вас приветствует бот для прогнозирования цены Telegram NFT
```

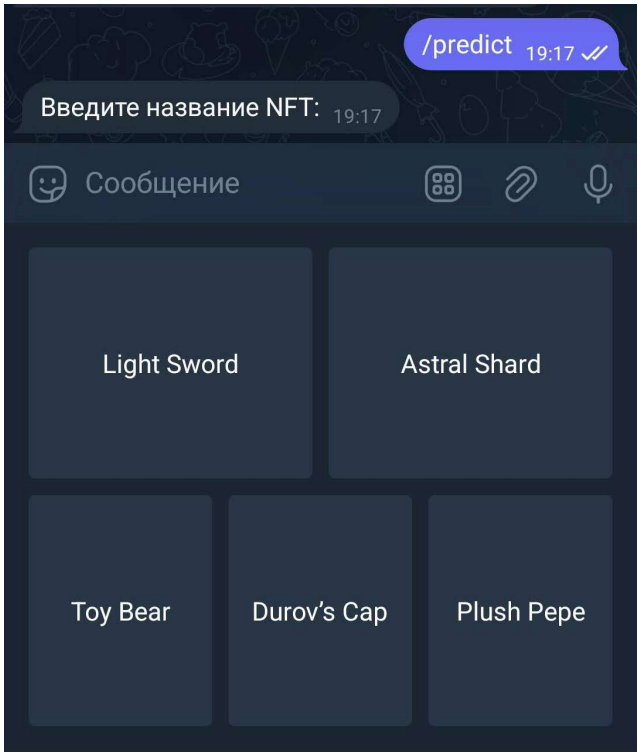
Вот, например, функция, получающая модель подарка:

```
def get_model(message):
    global bot_nft_model
    global bot_nft_rarity_model
    bot_nft_model = message.text
    if(message.text == "/models"):
        bot.send_message(message.from_user.id, str(models_dict[bot_nft_name]))
        bot.register_next_step_handler(message, get_model)
    elif(bot_nft_model in models_dict[bot_nft_name]):
        bot_nft_rarity_model = float(bot_nft_model.split()[-1][:-1])
        bot.send_message(message.from_user.id, "Введите фон NFT (пр. 'Pure Gold 1%' без кавычек) \n Список доступных фонов можно получить командой /bgs")
        bot.register_next_step_handler(message, get_bg)
    else:
        bot.send_message(message.from_user.id, "Модель не найдена. Попробуйте ещё раз")
        bot.send_message(message.from_user.id, "Введите модель NFT (пр. 'Jedi Luke 0.2%' без кавычек) \n Список доступных моделей можно получить командой /models")
        bot.register_next_step_handler(message, get_model)
```

Последний блок кода поддерживает постоянную работу бота, пока он запущен в среде выполнения:

```
while(True):
    try:
        bot.polling(none_stop = True, interval = 0)
    except Exception:
        print("reconnected...")
```

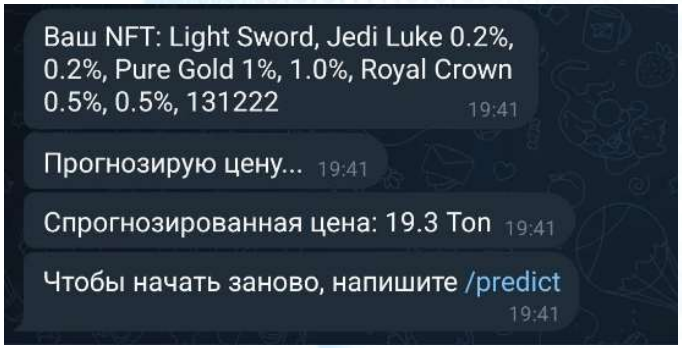
Таким образом, Telegram-бот выступает как практический интерфейс между пользователем и интеллектуальной системой прогнозирования, объединяя выводы машинного обучения с удобством взаимодействия, позволяя проводить моментальный анализ и оценку стоимости NFT без непосредственного обращения к исходному коду или работе с сырыми данными.



Шаг 1: выбрать коллекцию



Шаг 2: передать характеристики токена



Шаг 3: получить результат

4. Заключение

В ходе проведенного исследования была рассмотрена и практически реализована комплексная задача, совмещающая анализ современного феномена невзаимозаменяемых токенов (NFT) в рамках WEB-3 и применение методов машинного обучения для прогнозирования их стоимости.

Теоретический блок работы позволил выделить ключевые особенности экосистемы WEB-3, с акцентом на блокчейн TON, а также анализ специфики Telegram NFT, практическая часть проекта была сфокусирована на разработке и обучении регрессионной модели, способной предсказывать рыночную стоимость NFT, а также интеграции этой модели в бота мессенджера Telegram.

Перспективы дальнейшей работы связаны с расширением технических возможностей модели и объема NFT-коллекций Telegram, доступных для анализа, а также с применением более продвинутых моделей глубокого обучения для анализа изменений рыночной стоимости NFT-токенов в режиме реального времени.

СПАСИБО ЗА ВНИМАНИЕ!

