**Описание плана работы над проектом от команды «Мы от дяди Баира»**

Состав команды и зоны ответственности:

* Вамбуев Баир: координирование работы команды, создание проекта в репозитории, составление плана работы и подготовка к защите проекта.
* Ольга Редченко: исследование в выбранной предметной области, составление плана работы, подготовка к защите проекта.
* Карина Акчурина: предварительная обработка данных, разведовательный анализ данных.
* Михаил Кузьменков: анализ данных, подготовка данных для обучения моделей.
* Дмитрий Зайцев: построение моделей машинного обучения с использованием Optuna для подбора параметров (3 модели). Валидация качества модели.
* Елена Алмаева: построение моделей машинного обучения с использованием Optuna для подбора параметров (3 модели). Валидация качества модели.

**О проекте**

#### **Общая информация о заказчике:**

ООО "ПЭК" — это мультисервисный оператор логистических услуг, который занимается доставкой сборных грузов. Компания стремится превратить логистику в искусство, доступное каждому, и создает среду открытых доверительных отношений для развития и процветания людей и компаний.

#### **Целевая аудитория:**

* **B2B сегмент**: Основная аудитория, которая включает компании, занимающиеся перевозками на корпоративном уровне.
* **B2C сегмент**: Растущий сегмент, направленный на потребителей в сфере электронной коммерции, что требует более динамичных и адаптивных решений логистики.

#### **Задачи проекта:**

* **Цель**: Определение наиболее эффективного сегмента клиентов для таргетированной рекламы через email рассылки.
* **Ключевые задачи**:
  + Оценка ключевых метрик для оценки эффективности.
  + Составление детального клиентского портрета.
  + Выделение наиболее эффективного сегмента для таргетированных маркетинговых кампаний.

С учетом задачи предлагается следующий план решения задачи

**Создание проекта в репозитарии**

1. Организация структуры папок

* Создание папок: Распределите материалы проекта по папкам для четкой структуризации. Например, src для исходного кода, data для данных, docs для документации, и tests для тестов.
* Добавление файлов в репозиторий: Используйте команды Git для добавления и сохранения файлов в репозиторий: git add, git commit, и git push.

### Создание файла README.md

* **Содержание README.md**: Описание проекта, инструкции по установке, использованию и вкладу в проект. README должен быть информативным и кратким.
* **Форматирование**: Используйте Markdown для форматирования текста, добавления заголовков, списков, ссылок и изображений.

### Использование .gitignore

* **Конфигурация .gitignore**: Создайте файл .gitignore для исключения временных файлов, папок с данными и других несущественных для репозитория файлов. Это предотвратит их случайное добавление в репозиторий.

**Предварительная обработка данных**

1. Импорт данных:

* Использование библиотеки Pandas для загрузки данных из файлов behavior\_2023.csv  и feedback.csv. Это позволяет прочитать данные в формате DataFrame, что упрощает последующую обработку и анализ.
* Изучение данных behavior\_2023  и feedback для понимания структуры и типов данных: методы head(), info().

1. Очистка данных:

* Поиск и обработка пропущенных значений (методы isnull(), dropns()). Заполнение пропущенных значений медианными или средними значениями, если это целесообразно, или удаление строк/столбцов с большим количеством пропусков.
* Поиск и удаление дубликатов: методы duplicated(), drop\_dublicates().
* Удаление аномалий и выбросов, которые могут исказить результаты анализа. Это включает фильтрацию данных с использованием статистических методов, например, IQR (межквартильного размаха). Можно использовать метод метод describe().
* Проверка данных на согласованность.

### Обработка дат, добавление дополнительных полей с месяцами или кварталами для работы с сезонностью

* **Извлечение компонент даты**: из полной даты (например, 2024-05-15) можно извлечь год, месяц, день, день недели и так далее. Это помогает моделям машинного обучения улавливать временные тренды и сезонные изменения.
* **Добавление полей месяцев или кварталов**: в зависимости от анализа, можно добавить поля для месяца (например, январь, февраль) или квартала (Q1, Q2, Q3, Q4), что особенно полезно при работе с сезонными данными.
* **Кодирование временных переменных**: подобно категориальным переменным, временные данные часто кодируются для улучшения их обработки моделями машинного обучения, например, через one-hot encoding месяцев или бинаризацию дня недели.

### Генерация новых признаков на основе существующих данных

* **Взаимодействие признаков**: Можно создавать новые переменные путем комбинирования существующих. Например, произведение возраста на доход может дать новый признак для кредитного скоринга.
* **Агрегация данных**: Например, если у вас есть данные о транзакциях каждого клиента, можно агрегировать эти данные для создания признаков, таких как средний объем транзакции или общее количество транзакций за определенный период.
* **Применение статистических методов**: Вы можете использовать статистические меры (среднее, медиана, стандартное отклонение) на группах данных для создания новых признаков.

### Преобразование категориальных признаков в числовые

* **One-hot encoding**: метод преобразования категориальной переменной в набор бинарных переменных, где каждая категория представлена отдельной переменной со значениями 0 или 1.
* **Label encoding**: для порядковых переменных, где категории можно упорядочить в логической последовательности. Каждой категории присваивается уникальное числовое значение.
* **Binary encoding**: категории сначала преобразуются в двоичный код, а затем разбиваются на отдельные столбцы, что может быть эффективнее, чем one-hot encoding, особенно при большом количестве категорий.

1. Нормализация числовых переменных:

* Применение методов нормализации, таких как Min-Max Scaling или Standard Scaling, для улучшения сходимости алгоритмов машинного обучения. Нормализация помогает приблизить все числовые значения к одному масштабу, что важно для многих алгоритмов, чувствительных к различиям в масштабах признаков.

**Анализ данных:**

1. Визуализация распределений и основных статистических характеристик данных:

* Использование Matplotlib, Seaborn, Plotly.
* Примеры визуализаций:
  + Гистограммы и ящики с усами (boxplots) для анализа одномерных распределений данных по каждому числовому признаку, что помогает понять форму распределения, центральную тенденцию и наличие выбросов.
  + Диаграммы рассеяния (scatter plots) для исследования взаимосвязей между парами числовых переменных.
  + Тепловые карты (heatmaps) для визуализации статистических метрик, таких как среднее или медиана, по категориям.

### Поиск корреляции в данных. Этот этап анализа данных направлен на выявление взаимосвязей между переменными, что позволяет понять, какие признаки влияют друг на друга:

* **Тепловые карты (Heatmaps)**: визуальное представление матрицы корреляции, где различные уровни корреляций между переменными представлены в разных цветах. Тепловая карта позволяет быстро оценить, какие переменные сильно коррелируют или антикоррелируют между собой.
* **Матрица корреляции:** таблица, показывающая коэффициенты корреляции между всеми возможными парами переменных. Обычно используется коэффициент корреляции Пирсона для непрерывных переменных, который измеряет линейную связь между переменными.
* **Попарные графики корреляций (pairplot)**: c помощью библиотеки визуализации данных (например, seaborn в Python) можно создать сетку графиков для каждой пары переменных. Эти графики помогают визуально оценить, как переменные связаны друг с другом, включая возможные линейные или нелинейные взаимосвязи.

### Отбор признаков для обучения модели. Отбор признаков критически важен для повышения эффективности и точности моделей машинного обучения:

* **Удаление неинформативных признаков**: Некоторые переменные могут не нести значимой информации (например, уникальные идентификаторы или переменные с одним значением). К примеру вряд ли понадобятся id клиентов при обучении моделей.
* **Статистические методы**: Использование тестов, таких как ANOVA или хи-квадрат, для оценки значимости признаков.
* **Методы машинного обучения**: Использование алгоритмов, таких как Random Forest или Gradient Boosting, для оценки важности признаков на основе их вклада в точность предсказаний модели.
* **Итеративный отбор**: Последовательное добавление или удаление признаков и оценка влияния этих изменений на производительность модели.

### Определение количества кластеров и сегментация контрагентов. При использовании методов кластеризации, таких как K-means, важно определить оптимальное количество кластеров:

* **Метод локтя (Elbow method)**: через изменение суммы квадратов расстояний от точек до центров их кластеров в зависимости от количества кластеров, можно найти точку, в которой увеличение количества кластеров перестает значительно улучшать модель (точка "локтя").
* **Силуэтный анализ (Silhouette analysis)**: Оценка, насколько хорошо каждый объект был кластеризован. Высокий средний силуэтный коэффициент указывает на хорошее разделение кластеров.
* **Предварительно заданное количество кластеров**: предварительно можно сказать, что можно разделит пользователей на 3 кластера, эту информацию можно использовать при настройке алгоритма кластеризации.
* Сегментация контрагентов в качестве этапа в анализе данных позволяет определить различные группы контрагентов на основе их поведения и характеристик. Это помогает в дальнейшем таргетировании маркетинговых и коммуникационных стратегий. Для этого часто используется алгоритм K-means, один из самых популярных методов кластеризации.

1. Анализ характеристик каждого кластера. После формирования кластеров следующим шагом является анализ характеристик каждого кластера. Такой анализ помогает понять, какие поведенческие или демографические особенности объединяют контрагентов в каждом кластере.

* Сравнение кластеров: Сравните средние значения или медианы различных переменных внутри кластеров для выявления отличий и сходств.
* Визуализация: Используйте гистограммы, ящики с усами или тепловые карты для визуализации распределений признаков по кластерам.
* Интерпретация: Определите, какие признаки наиболее важны для каждого кластера, и как эти признаки могут быть использованы для улучшения бизнес-стратегий, направленных на конкретные группы контрагентов.

**Построение моделей машинного обучения с использованием Optuna для подбора параметров**

Построение моделей машинного обучения для формирования рекомендаций для отдела маркетинга включает не только выбор алгоритма, но и тщательную настройку его параметров для достижения максимальной эффективности. Optuna — это библиотека для автоматического подбора гиперпараметров, которая помогает оптимизировать модели машинного обучения. Рассмотрим использование различных алгоритмов вместе с Optuna:

1. Логистическая регрессия (Scikit-learn):

* Логистическая регрессия используется для моделирования вероятности наступления события путем логит-преобразования зависимости между независимыми переменными и зависимой переменной (бинарной).
* С помощью Optuna можно оптимизировать такие параметры, как коэффициент регуляризации `C` и тип регуляризации `penalty` (L1, L2, ElasticNet).

1. Дерево решений (Scikit-learn):

* Дерево решений — это модель предсказательной аналитики, которая используется для регрессии и классификации. ё
* При использовании Optuna для оптимизации параметров дерева решений, важными параметрами являются максимальная глубина дерева (max\_depth), минимальное количество объектов, необходимое для разделения внутреннего узла (min\_samples\_split), и минимальное количество объектов в листовом узле (min\_samples\_leaf). Эти параметры помогают контролировать сложность модели и предотвратить переобучение, улучшая обобщающую способность модели

1. Случайный лес (Scikit-learn):

* Случайный лес — это ансамблевая модель, состоящая из множества деревьев решений, что делает её отличным инструментом для анализа нелинейных зависимостей и важности признаков.
* Параметры, которые часто настраиваются с помощью Optuna, включают число деревьев (`n\_estimators`), максимальную глубину дерева (`max\_depth`) и минимальное число объектов в листьях (`min\_samples\_leaf`).

1. XGBoost:

* XGBoost использует градиентный бустинг для построения последовательности деревьев решений, каждое из которых стремится исправить ошибки предыдущих деревьев.
* Важные параметры для оптимизации с Optuna включают `learning\_rate`, `max\_depth`, `subsample` (доля выборки для обучения каждого дерева), и `colsample\_bytree` (доля признаков для обучения каждого дерева).

1. LightGBM:

* LightGBM — это градиентный бустинговый алгоритм, который оптимизирован для высокой скорости работы и эффективности с большими объемами данных.
* Настройка параметров с Optuna может включать `num\_leaves` (количество листьев в дереве), `max\_depth`, `min\_data\_in\_leaf` (минимальное количество данных в листе) и `feature\_fraction` (доля признаков для каждого дерева).

1. CatBoost:

* CatBoost эффективно работает с категориальными данными, автоматически обрабатывая их без предварительного кодирования.
* Параметры, которые можно оптимизировать через Optuna, включают `depth` (глубина деревьев), `learning\_rate` и `l2\_leaf\_reg` (коэффициент регуляризации L2).

Пример использования Optuna для настройки параметров модели XGBoost:

import optuna

import xgboost as xgb

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

data = pd.read\_csv('data.csv')

X = data.drop('target', axis=1)

y = data['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

def objective(trial):

param = {

'verbosity': 0,

'objective': 'binary:logistic',

'learning\_rate': trial.suggest\_float('learning\_rate', 0.01, 0.2),

'n\_estimators': trial.suggest\_int('n\_estimators', 100, 1000),

'max\_depth': trial.suggest\_int('max\_depth', 3, 9),

'subsample': trial.suggest\_float('subsample', 0.5, 1.0),

'colsample\_bytree': trial.suggest\_float('colsample\_bytree', 0.5, 1.0)

}

model = xgb.XGBClassifier(\*\*param)

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, preds)

return accuracy

study = optuna.create\_study(direction='maximize')

study.optimize(objective, n\_trials=100)

print('Best trial:', study.best\_trial.params)

**План по валидации качества модели**

Для оценки эффективности работы выбора наилучшего метода решения задачи кластеризации пользователей и выбора рекомендательной модели планируется протестировать следующие метрики:

1. Accuracy

* Описывает общую точность предсказания модели по всем классам (TP+TN)/(TP+TN+FP+FN).
* Подойдет только в случае сбалансированных классов.

1. Precision

* Precision = TP/(TP+FP).
* Показывает долю объектов, названных моделью положительными и при этом действительно являющимися положительными.
* Не зависит от соотношения классов, применимы в условиях несбалансированных выборок.

1. Recall

* Recall = TP / (TP + FN).
* Показывает какую долю объектов положительного класса из всех объектов положительного класса, которые нашел алгоритм.
* Не зависит от соотношения классов, применимы в условиях несбалансированных выборок.

1. F1-мера

* F1 = 2 \* (Precision \* Recall) / (Precision + Recall)
* Показывает одновременно насколько хорошо модель находит объекты положительного класса из всех объектов положительного класса и какая доля из тех, кого алгоритм назвал положительным классом, действительно являются положительным классом.

В случае бинарного классификатора эффективно использовать такие метрики оценки эффективности моделей, как ROC и AUC

1. ROC

* График, который иллюстрирует производительность классификационной модели при всех возможных порогах классификации. Ось X данного графика представляет собой FPR (ложноположительную частоту, а ось Y — TRP (истинноположительную частоту). TRP = TP/(TP+FN), FPR = FP / (TN + FP)

6. AUC

* Позволяет суммировать производительность модели одним числом, измеряя площадь под кривой ROC. AUC колеблется от 0 до 1, где более высокое значение AUC указывает на более высокую производительность модели.

Предварительно планируется ориентироваться на F1 при выборе оптимальной модели.