Следующее домашнее задание состоит из двух вариантов: базовый и продвинутый, студенту необходимо выбрать один вариант на выбор. Оба варианта оцениваются в формате «зачет/незачет». Выполненная работа должна быть представлена в формате .ipynb или ссылкой на ноутбук в Google Colaboratory. В начале ноутбука должен быть указан выбранный вариант. В ноутбуке под ячейкой должен быть результат выполнения, если ячейка должна что-то выводить.

Оформление:

1. В ноутбуке должны содержаться тематические разделы (заголовки первого или второго уровня) — например, по пунктам задания.
2. Код должен сопровождаться либо комментариями внутри ячеек, либо кратким текстом перед ячейкой. Пожалуйста, помните, что документировать код — это очень важно.

В качестве данных для работы используются результаты предыдущего домашнего задания (тема «Анализ данных»). **Проверьте, чтобы номера заказов были уникальными и не дублировались в разных строках.**

Требуемые колонки: store\_id, order\_price\*, profit, delivery\_distance, planned\_prep\_time, region\_id, items\_count, prepared\_on\_time. Если Выбрали продвинутый вариант, то добавить колонки day\_of\_the\_week, закодированные hour\_start, minutes\_start.

\***order\_price** – это сумма цен товаров, находящихся в каждом заказе.

Базовый вариант:

1. Загрузить предобработанные данные из предыдущего домашнего задания.
2. Разделить данные на признаки и целевую переменную. Целевая переменная — «prepared\_on\_time».
3. Нормализовать данные (с помощью минимаксной нормализации или стандартизации).
4. Разделить данные на обучающую и тестовую выборки в отношении 80% на 20%.
5. Подобрать оптимальные гиперпараметры для логистической регрессии. Также вывести оценку на кросс-валидации.
6. Заново обучить модель с подобранными гиперпараметрами. Вывести метрики: accuracy, precision, recall, f1-score.
7. Протестировать обученную модель на тестовой выборке. Вывести метрики: accuracy, precision, recall, f1-score.
8. Сделать выводы по полученным метрикам.

Продвинутый вариант:

1. Загрузить предобработанные данные из предыдущего домашнего задания. Вынести в отдельный датафрейм признак «order\_price» и «order\_id».
2. Также из исходного датафрейма вынести в отдельный датафрейм «order\_id» и «product\_name» (каждому product\_id соответствует product\_name из products.csv).
3. Создать новую колонку «products», в которой находятся названия всех продуктов в рамках одного заказа (order\_id). Колонку «product\_name» удалить, также удалить дубликаты записей. В полученном датафрейме должно остаться две колонки: order\_id и products.
4. Присоединить на основе order\_id признак «order\_price». В результирующем датафрейме должно остаться 3 колонки: ID заказа (order\_id), перечень всех его продуктов (products) и стоимость заказа (order\_price).
5. С помощью TF-IDF (TfidfVectorizer) создать числовое представление для товаров из колонки «products». Полученную матрицу перевести в датафрейм.
6. К полученному выше датафрейму присоединить «order\_price». Датафрейм должен содержать признаки после подсчета TF-IDF и цену заказа.
7. Понизить размерность полученного датафрейма (PCA, t-SNE, UMAP на выбор).
8. Определить начальное количество кластеров по сокращенному датафрейму. Интерпретировать результат.
9. Кластеризовать данные с учетом оптимального числа кластеров. Получить внутренние метрики.
10. Сделать выводы по результатам кластеризации.
11. Можно создать WordCloud (облако слов) для каждого кластера — визуальное отображение наиболее значимых слов в кластере.

Подсказка по созданию WordCloud:

from wordcloud import WordCloud

import matplotlib.pyplot as plt

import numpy as np

# Применение кластеризации

labels = kmeans.fit\_predict(X\_tfidf)

# Добавление меток кластеров в DataFrame

df['cluster'] = labels

# Проходим по каждому кластеру

for cluster in np.unique(labels):

# Извлекаем индексы документов, принадлежащих текущему кластеру

cluster\_docs\_indices = np.where(labels == cluster)

# Суммируем TF-IDF значения для всех слов в документах этого кластера

tfidf\_cluster = X\_tfidf[cluster\_docs\_indices].sum(axis=0)

# Преобразуем в словарь: ключи - слова, значения - веса

tfidf\_cluster\_dict = {words[i]: tfidf\_cluster[0, i] for i in range(tfidf\_cluster.shape[1])}

# Создаем WordCloud для текущего кластера

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate\_from\_frequencies(tfidf\_cluster\_dict)

# Отображаем WordCloud

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title(f'Cluster {cluster} Word Cloud')

plt.show()