МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБО6РОСТРОЕНИЯ»

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Поляк Марк Дмитриевич		
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Практическое задание №3

«ЛР3. Алгоритмы классификации»

по курсу:Основы машинного обучения

ВАРИАНТ №32

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	4233К	1.6.25	Ху Чунь
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Цель работы

В данной лабораторной работе необходимо реализовать два алгоритма классификации и затем использовать готовые реализации алгоритмов в библиотеке sklearn. Реализация SVM является необязательной и выполняется с целью получения дополнительных баллов.

Знакомство с алгоритмами классификации в машинном обучении.

Задачи

1. Определить номер варианта

Перейдите по ссылке из личного кабинета на Google Таблицу со списком студентов. Найдите свое ФИО в списке и запомните соответствующий порядковый номер (поле № п/п) в первом столбце. Заполните его в ячейке ниже и выполните ячейку. Если вы не можете найти себя в списке, обратитесь к своему преподавателю.

```
### BEGIN YOUR CODE

Student_ID = 32

### END YOUR CODE
```

Теперь выполните следующую ячейку. Она вычислит номер задания и выведет его.

```
algo_1_list = ['решающие деревья', 'логистическая регрессия', 'k ближайших соседей']
algo_1 = None if STUDENT_ID is None else algo_1_list[STUDENT_ID % len(algo_1_list)]

algo_2_list = ['SVM с линейным ядром', 'SVM с полиномиальным ядром степени 3', 'SVM с полиномиальным ядром степени 2', 'SVM с ядром RBF', 'SVM с сигмоидальным ядром']
algo_2 = None if STUDENT_ID is None else algo_2_list[STUDENT_ID % len(algo_2_list)]

datasets = [
    ('Stroke Prediction Dataset', 'https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset', 'https://www.kaggle.com/api/v1/datasets/download/fedesoriano/stroke-prediction-dataset'),
    ('Heart Failure Prediction Dataset', 'https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction', 'https://www.kaggle.com/api/v1/datasets/download/fedesoriano/heart-failure-prediction'),
    ('E-Commerce Shipping Data', 'https://www.kaggle.com/datasets/prachi13/customer-analytics', 'https://www.kaggle.com/api/v1/datasets/download/prachi13/customer-analytics'),
    ('Pulsar Classification For Class Prediction', 'https://www.kaggle.com/datasets/brsdincer/pulsar-classification-for-class-prediction', 'https://www.kaggle.com/api/v1/datasets/download/brsdincer/pulsar-classification-for-class-prediction'),
    ('Bank Customer Churn Prediction',
```

```
'https://www.kaggle.com/datasets/shubhammeshram579/bank-customer-churn-prediction',
'https://www.kaggle.com/api/v1/datasets/download/shubhammeshram579/bank-customer-churn-prediction'),
        ('Heart Disease Health Indicators Dataset',
'https://www.kaggle.com/datasets/alexteboul/heart-disease-health-indicators-dataset',
'https://www.kaggle.com/api/v1/datasets/download/alexteboul/heart-disease-health-indicators-dataset'),
        ('Body signal of smoking',
'https://www.kaggle.com/datasets/kukuroo3/body-signal-of-smoking',
'https://www.kaggle.com/api/v1/datasets/download/kukuroo3/body-signal-of-smoking'),
        ('Dataset Surgical binary classification',
'https://www.kaggle.com/datasets/omnamahshivai/surgical-dataset-binary-classification/data',
'https://www.kaggle.com/api/v1/datasets/download/omnamahshivai/surgical-dataset-binary-classification'),
        ('Campus Recruitment',
'https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement',
'https://www.kaggle.com/api/v1/datasets/download/benroshan/factors-affecting-campus-placement'),
        ('Online Payments Fraud Detection Dataset',
'https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset',
'https://www.kaggle.com/api/v1/datasets/download/rupakroy/online-payments-fraud-detection-dataset')
    ]
    DATASET_ID = None if STUDENT_ID is None else STUDENT_ID % len(datasets)

    if algo_1 is None or algo_2 is None or DATASET_ID is None:
        print("ОШИБКА! Не указан порядковый номер студента в списке группы.")
    else:
        print(f"Датасет: {datasets[DATASET_ID][0]}\nСсылка на описание датасета:
{datasets[DATASET_ID][1]}\nСсылка на скачивание датасета {datasets[DATASET_ID][2]}")
        print(f"Первый алгоритм: {algo_1}\nВторой алгоритм: {algo_2}")
```

Получение результатов прогона:

Датасет: E-Commerce Shipping Data

Ссылка на описание датасета:
https://www.kaggle.com/datasets/prachi13/customer-analytics

Ссылка на скачивание датасета
https://www.kaggle.com/api/v1/datasets/download/prachi13/customer-analytics

Первый алгоритм: k ближайших соседей

Второй алгоритм: SVM с полиномиальным ядром степени 2

Скачайте датасет с помощью команды !wget <dataset_url>, где <dataset_url> необходимо заменить на ссылку на датасет, появившуюся после выполнения предыдущей ячейки. При необходимости разархивируйте датасет, используя команды !unzip, !tar и др.

Примечание: в Jupyter-ноутбуке можно использовать любые команды командного интерпретатора bash. Для этого необходимо поставить в ячейке с кодом восклицательный знак !, после которого записать команду bash со всеми необходимыми аргументами. Результат выполнения этой команды bash будет возвращен в Jupyter и его можно использовать в коде на Python.

```
### BEGIN YOUR CODE
!wget https://www.kaggle.com/api/v1/datasets/download/prachi13/customer-analytics
!unzip customer-analytics
### END YOUR CODE
```

Получение результатов прогона:

```
--2025-06-03 08:46:52--  https://www.kaggle.com/api/v1/datasets/download/prachi13/customer-analytics
Resolving www.kaggle.com (www.kaggle.com)... 35.244.233.98
Connecting to www.kaggle.com (www.kaggle.com)|35.244.233.98|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://storage.googleapis.com:443/kaggle-data-sets/1176727/1970226/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceacc
--2025-06-03 08:46:53--  https://storage.googleapis.com/kaggle-data-sets/1176727/1970226/bundle/archive.zip?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.
Resolving storage.googleapis.com (storage.googleapis.com)... 142.250.141.207, 142.251.2.207, 74.125.137.207, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|142.250.141.207|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 123784 (121K) [application/zip]
Saving to: 'customer-analytics'

customer-analytics  100%[===================>] 120.88K  --.-KB/s    in 0.001s

2025-06-03 08:46:53 (118 MB/s) - 'customer-analytics' saved [123784/123784]

Archive:  customer-analytics
  inflating: Train.csv
```

2. Подготовить среду разработки

Добавьте импорт всех необходимых библиотек в ячейке ниже. Постарайтесь не импортировать библиотеки в других ячейках, чтобы избежать ошибок в коде.

Замечание: если при попытке импортировать библиотеку появляется ошибка ModuleNotFoundError, установите библиотеку при помощи команды !pip install LIBRARY_NAME.

```
### BEGIN YOUR CODE
import pandas as pd
import numpy as np
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score,
roc_curve,precision_score, f1_score
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, MinMaxScaler,OneHotEncoder
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn import svm
### END YOUR CODE
```

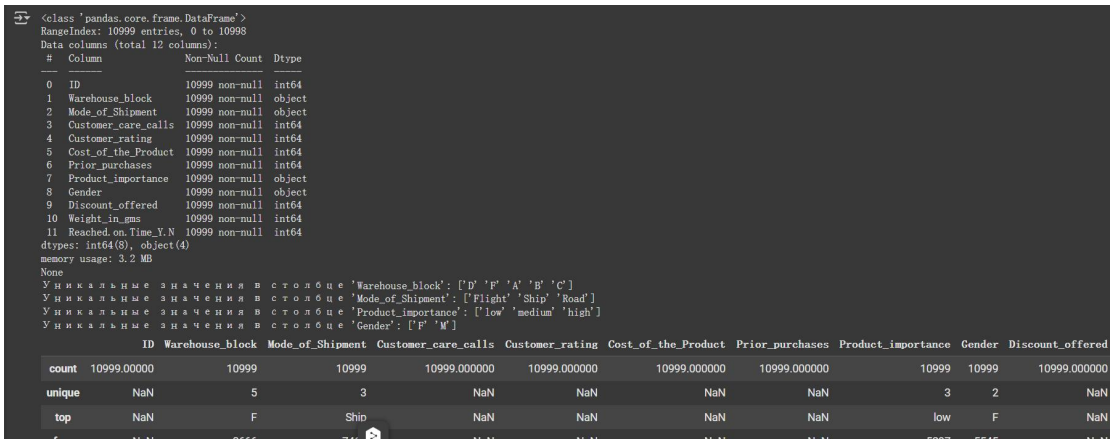
3. Посмотреть на общую картину (1 балл)

Ознакомьтесь с информацией о датасете по ссылке из задания и напишите один абзац текста с описанием решаемой задачи. В частности, ответьте в своем тексте на следующие вопросы:

- Каков размер датасета? (в Мб)
- Сколько в нем записей (объектов)?
- Сколько признаков (фич) у объектов в датасете?
- Есть ли категориальные данные? Какие?
- Есть ли пропущенные значения?
- Есть ли в датасете столбец с ответами (target)? Какой у него тип данных?
- Какую задачу может решать модель бинарной классификации, построенная на этом датасете? Какую величину она будет предсказывать?
- Приведите основные статистические данные о датасете, которые можно получить вызовом одной-двух функций в pandas. Какие выводы о датасете можно сделать?

В ячейке ниже напишите код, который выводит всю необходимую информацию, а в ячейке под ней (ее тип - Markdown, т.е. текст) опишите своими словами решаемую задачу и используемый набор данных.

```
### BEGIN YOUR CODE
df = pd.read_csv("Train.csv",)
display(df.info(verbose = True,memory_usage='deep'))

object_columns = df.select_dtypes(include=['object']).columns
for col in object_columns:
    unique_values = df[col].unique()
    print(f"Уникальные значения в столбце '{col}': {unique_values}")
display(df.describe(include='all'))
display(df.hist(bins = 100, figsize = [20,7]))

### END YOUR CODE
```

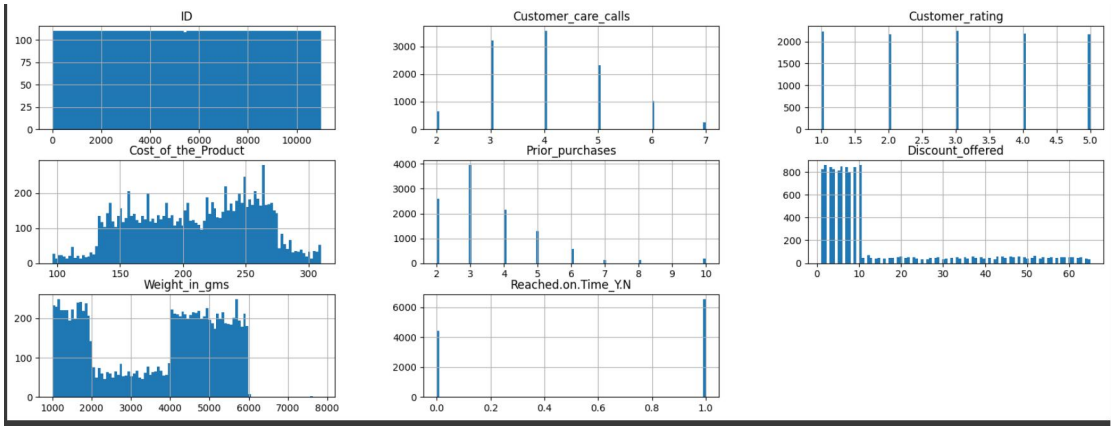


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   10999 non-null  int64
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64
 4   Customer_rating      10999 non-null  int64
 5   Cost_of_the_Product  10999 non-null  int64
 6   Prior_purchases      10999 non-null  int64
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64
 10  Weight_in_gms        10999 non-null  int64
 11  Reached.on.Time_Y.N  10999 non-null  int64
dtypes: int64(8), object(4)
memory usage: 3.2 MB
None
Уникальные значения в столбце 'Warehouse_block': ['D' 'F' 'A' 'B' 'C']
Уникальные значения в столбце 'Mode_of_Shipment': ['Flight' 'Ship' 'Road']
Уникальные значения в столбце 'Product_importance': ['low' 'medium' 'high']
Уникальные значения в столбце 'Gender': ['F' 'M']
```

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
count	10999.00000	10999	10999	10999.000000	10999.000000	10999.000000	10999.000000	10999	10999	10999.000000
unique	NaN	5	3	NaN	NaN	NaN	NaN	3	2	NaN
top	NaN	F	Ship	NaN	NaN	NaN	NaN	low	F	NaN

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Product_importance	Gender	Discount_offered
count	10999.00000	10999	10999	10999.000000	10999.000000	10999.000000	10999.000000	10999	10999	10999.000000
unique	NaN	5	3	NaN	NaN	NaN	NaN	3	2	NaN
top	NaN	F	Ship	NaN	NaN	NaN	NaN	low	F	NaN
freq	NaN	3666	7462	NaN	NaN	NaN	NaN	5297	5545	NaN
mean	5500.00000	NaN	NaN	4.054459	2.990545	210.196836	3.567597	NaN	NaN	13.373216
std	3175.28214	NaN	NaN	1.141490	1.413603	48.063272	1.522860	NaN	NaN	16.205527
min	1.00000	NaN	NaN	2.000000	1.000000	96.000000	2.000000	NaN	NaN	1.000000
25%	2750.50000	NaN	NaN	3.000000	2.000000	169.000000	3.000000	NaN	NaN	4.000000
50%	5500.00000	NaN	NaN	4.000000	3.000000	214.000000	3.000000	NaN	NaN	7.000000
75%	8249.50000	NaN	NaN	5.000000	4.000000	251.000000	4.000000	NaN	NaN	10.000000
max	10999.00000	NaN	NaN	7.000000	5.000000	310.000000	10.000000	NaN	NaN	65.000000

```
array([[<Axes: title={'center': 'ID'}>,
        <Axes: title={'center': 'Customer_care_calls'}>,
        <Axes: title={'center': 'Customer_rating'}>],
       [<Axes: title={'center': 'Cost_of_the_Product'}>,
        <Axes: title={'center': 'Prior_purchases'}>,
        <Axes: title={'center': 'Discount_offered'}>],
       [<Axes: title={'center': 'Weight_in_gms'}>,
        <Axes: title={'center': 'Reached.on.Time_Y.N'}>, <Axes: >]],
      dtype=object)
```



Данный датасет, весит 3.2 Мб и содержит 10999 объектов с 10 признаками и целевым столбцом "Reached.on.Time_Y.N".

Из признаков есть 4 категориальных:

- "Warehouse_block" — (A,B,C,D,F)
- "Mode_of_Shipment" — (Flight, Ship, Road)
- "Product_importance" — (low, medium, high)
- "Gender" — (F, M)

Пропущенные значения отсусвуют.

Модель построенная на этом датасете может решать задачу предсказания доставки в срок товара в зависимости от свойств покупателя, характеристик товара и других данных.

По статистическим данным видно что присутсвует признак "ID" который не несёт никакой информации и является просто порядковым номером заказа. Также можно заметить что данные расположены либо разреженно, либо неравномерно с выбросами. Одно значение целевой переменной в полтора раза больше другого.

4. Подготовка данных

При необходимости выполните преобразование признаков, фильтрацию, предобработку.

Разбейте выборку на обучающую и тестовую. Не забудьте отделить признаки от меток классов.

```
### BEGIN YOUR CODE
df = df.astype(float,errors="ignore")
df = df.drop("ID",axis =1)

X= df.drop('Reached.on.Time_Y.N',axis = 1)
y = df['Reached.on.Time_Y.N']

categorical_features = [
    'Warehouse_block',
    'Mode_of_Shipment',
    'Product_importance',
    'Gender'
]

numerical_features = [
    'Customer_care_calls',
    'Customer_rating',
    'Cost_of_the_Product',
    'Prior_purchases',
    'Discount_offered',
    'Weight_in_gms'
]



# Строим трансформер с ColumnTransformer
preprocessor = ColumnTransformer([
    ("num", StandardScaler(), numerical_features),    # Пропускаем числовые признаки без изменений
    ("cat", OneHotEncoder(drop='first', handle_unknown='ignore'), categorical_features) # Кодируем категориальные признаки
])
X_clean = preprocessor.fit_transform(X)

y_clean = y.copy()
X_train, X_test, y_train, y_test = train_test_split(X_clean, y_clean, test_size=0.2, random_state=42)
### END YOUR CODE
```

5. Реализовать первый алгоритм (3 балла)

Замените название класса на одно из следующих, в зависимости от названия вашего алгоритма: KNNClassifier, LogisticRegressionClassifier, DecisionTreeClassifier.

Определите параметры, которые необходимо подавать на вход алгоритма, и задайте их

в качестве аргументов функции __init__. В остальных методах входные данные определены, их менять не нужно.

Примечание 1: в Python параметр self - специальный параметр, который передается первым аргументом в методе класса и представляет собой ссылку на экземпляр класса.

Примечание 2: в Python инструкция pass является заглушкой. Удалите заглушки при выполнении задания.

Внимание: нельзя использовать библиотечный код для реализации алгоритма классификации, напишите свой!

```
class KNNClassifier:
    def __init__(self, k): # input_value - переменная, введенная для демонстрации синтаксиса Python. Удалите ее.
        ### BEGIN YOUR CODE
        self.k = k
        self.X_train = None
        self.y_train = None
        ### END YOUR CODE

    def fit(self, X, y): # функция, которая обучает классификатор
        '''
        param X: features from train set
        param y: labels from train set
        '''
        ### BEGIN YOUR CODE
        self.X_train = np.array(X)

        self.y_train = np.array(y)

        ### END YOUR CODE

    def predict(self, X): # функция, которая делает предсказания на основе входных данных
        '''
        param X: input features
        '''
        ### BEGIN YOUR CODE
        X = np.array(X)
        predictions = []
        for x in X:
            distances = self._compute_distances(x)
            nearest_indices = np.argsort(distances)[:self.k]
            nearest_neighbors = self.y_train[nearest_indices]
            unique_classes, counts = np.unique(nearest_neighbors, return_counts=True)
            majority_class = unique_classes[np.argmax(counts)]
```

```
                predictions.append(majority_class)

        return np.array(predictions)
    def predict_proba(self, X):
        X = np.array(X)
        probabilities = []
        for x in X:
            distances = self._compute_distances(x)
            nearest_indices = np.argsort(distances)[:self.k]
            nearest_neighbors = self.y_train[nearest_indices]
            unique_classes = np.unique(self.y_train)
            class_probs = []
            for c in unique_classes:
                # Вероятность = доля соседей, принадлежащих классу c
                prob = np.sum(nearest_neighbors == c) / self.k
                class_probs.append(prob)
            probabilities.append(class_probs)
        return np.array(probabilities)
    def _compute_distances(self,x):
        return np.sqrt(np.sum((self.X_train - x) ** 2, axis=1))

        ### END YOUR CODE
```

Используйте написанный выше код для обучения классификатора на обучающей выборке. Сделайте предсказания на тестовой выборке. Рассчитайте метрику Accuracy и сохраните ее значение в переменной algo_1_accuracy.

```
### BEGIN YOUR CODE

knn = KNNClassifier(7)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(accuracy_score(y_test,y_pred))

algo_1_accuracy = accuracy_score(y_test,y_pred)

### END YOUR CODE
```

6. Познакомиться с реализацией алгоритма SVM в библиотеке scikit-learn (1 балл)

Используя документацию к библиотеке scikit-learn, найдите реализацию SVM. Подставьте параметры в соответствии с описанием второго алгоритма, заданного по

варианту. Обучите классификатор при помощи средств библиотеки scikit-learn. Сделайте предсказания на тестовой выборке. Рассчитайте метрику Accuracy и сохраните ее значение в переменной algo_2_accuracy.

```
### BEGIN YOUR CODE
svm_model = svm.SVC(kernel="poly",degree = 2,probability=True)
svm_model.fit(X_train,y_train)
y_pred = svm_model.predict(X_test)
print(accuracy_score(y_test,y_pred))
algo_2_accuracy = accuracy_score(y_test,y_pred)

### END YOUR CODE
```

7. *БОНУСНОЕ ЗАДАНИЕ. Самостоятельно реализовать алгоритм SVM (3 балла)

Эта задача не является обязательной к выполнению.

Определите параметры, которые необходимо подавать на вход алгоритма, и задайте их в качестве аргументов функции __init__. В остальных методах входные данные определены, их менять не нужно.

```
class SVM:
    def __init__(self, C=1.0, degree=2, max_iter=100, tol=0.01):
        """
        Инициализация SVM классификатора с полиномиальным ядром

        Параметры:
        C : float, параметр регуляризации (по умолчанию 1.0)
        degree : int, степень полиномиального ядра (по умолчанию 2)
        max_iter : int, максимальное количество итераций (по умолчанию 100)
        tol : float, допуск сходимости (по умолчанию 0.01)
        """
        ### BEGIN YOUR CODE
        self.C = C
        self.degree = degree
        self.max_iter = max_iter
        self.tol = tol
        self.b = 0
        self.alphas = None
        self.X_train = None
        self.y_train = None
        self.X_sv = None
        self.y_sv = None
        self.alphas_sv = None
```

```
        ### END YOUR CODE

    def polynomial_kernel(self, x1, x2):
        """
        Полиномиальное ядро степени d
        Формула: (x1·x2 + 1)^degree
        """
        return (np.dot(x1, x2) + 1) ** self.degree

    def fit(self, X, y):
        """
        Обучение SVM с использованием SMO алгоритма
        Оптимизация: используется подвыборка 1000 образцов для ускорения
        """
        ### BEGIN YOUR CODE
        # 1. Обработка различных форматов данных
        if hasattr(X, 'toarray'):    # для разреженных матриц
            X = X.toarray()
        if hasattr(y, 'values'):    # для pandas Series
            y = y.values

        # 2. Выбор подвыборки (1000 случайных образцов)
        n_samples = X.shape[0]
        sample_size = min(1000, n_samples)    # не более 1000 образцов
        indices = np.random.choice(n_samples, sample_size, replace=False)
        X = X[indices]
        y = y[indices]

        # 3. Преобразование меток классов (0 → -1, 1 → 1)
        y = np.where(y == 0, -1, 1)

        # 4. Инициализация параметров
        n_samples, n_features = X.shape
        self.X_train = X
        self.y_train = y
        self.alphas = np.zeros(n_samples)
        self.b = 0

        # 5. Упрощенный SMO алгоритм (ограниченный max_iter)
        for iter_num in range(self.max_iter):
            num_changed_alphas = 0
            for i in range(n_samples):
                # 5.1 Вычисление ошибки для i-го образца
                Ei = self._decision_function(X[i]) - y[i]
```

```
                    # 5.2 Проверка условий KKT для нарушения
                    if ((y[i] * Ei < -self.tol and self.alphas[i] < self.C) or
                        (y[i] * Ei > self.tol and self.alphas[i] > 0)):

                        # 5.3 Выбор случайного j ≠ i
                        j = np.random.choice([k for k in range(n_samples) if k != i])
                        Ej = self._decision_function(X[j]) - y[j]

                        # 5.4 Вычисление eta (проверка на положительность)
                        eta = 2 * self.polynomial_kernel(X[i], X[j]) - \
                              self.polynomial_kernel(X[i], X[i]) - \
                              self.polynomial_kernel(X[j], X[j])
                        if eta >= 0:
                            continue

                        # 5.5 Сохранение старых значений alpha
                        alpha_i_old = self.alphas[i]
                        alpha_j_old = self.alphas[j]

                        # 5.6 Вычисление границ L и H
                        if y[i] != y[j]:
                            L = max(0, alpha_j_old - alpha_i_old)
                            H = min(self.C, self.C + alpha_j_old - alpha_i_old)
                        else:
                            L = max(0, alpha_i_old + alpha_j_old - self.C)
                            H = min(self.C, alpha_i_old + alpha_j_old)

                        # 5.7 Обновление alpha_j и alpha_i
                        self.alphas[j] = alpha_j_old - y[j] * (Ei - Ej) / eta
                        self.alphas[j] = max(L, min(self.alphas[j], H))
                        self.alphas[i] = alpha_i_old + y[i] * y[j] * (alpha_j_old -
self.alphas[j])

                        # 5.8 Обновление смещения b
                        b1 = self.b - Ei -
y[i]*(self.alphas[i]-alpha_i_old)*self.polynomial_kernel(X[i], X[i]) \
                                    -
y[j]*(self.alphas[j]-alpha_j_old)*self.polynomial_kernel(X[i], X[j])
                        b2 = self.b - Ej -
y[i]*(self.alphas[i]-alpha_i_old)*self.polynomial_kernel(X[i], X[j]) \
                                    -
y[j]*(self.alphas[j]-alpha_j_old)*self.polynomial_kernel(X[j], X[j])
```

```
                        if 0 < self.alphas[i] < self.C:
                            self.b = b1
                        elif 0 < self.alphas[j] < self.C:
                            self.b = b2
                        else:
                            self.b = (b1 + b2) / 2

                        num_changed_alphas += 1

                    # 5.9 Проверка сходимости
                    if num_changed_alphas == 0 and iter_num > 10:
                        break

            # 6. Сохранение опорных векторов
            sv_indices = self.alphas > 1e-5
            self.X_sv = X[sv_indices]
            self.y_sv = y[sv_indices]
            self.alphas_sv = self.alphas[sv_indices]
            ### END YOUR CODE

        def _decision_function(self, x):
            """
            Внутренняя функция для вычисления решения (используется во время
обучения)
            """
            if self.alphas is None:
                return 0

            result = 0
            for i in range(len(self.alphas)):
                if self.alphas[i] > 0:    # Только для опорных векторов
                    result += self.alphas[i] * self.y_train[i] *
self.polynomial_kernel(self.X_train[i], x)
            return result + self.b

        def decision_function(self, x):
            """
            Решающая функция после обучения
            Возвращает значение f(x) = sign(Σ alpha_i * y_i * K(x_i, x) + b)
            """
            if self.alphas_sv is None or len(self.alphas_sv) == 0:
                return 0

            result = 0
```

```
            for i in range(len(self.alphas_sv)):
                result += self.alphas_sv[i] * self.y_sv[i] * self.polynomial_kernel(self.X_sv[i],
x)
            return result + self.b

        def predict(self, X):
            """
            Предсказание меток классов (0 или 1)
            Для каждого образца вычисляет decision_function и возвращает 1 если >=0,
иначе 0
            """
            ### BEGIN YOUR CODE
            # Преобразование входных данных
            if hasattr(X, 'toarray'):    # для разреженных матриц
                X = X.toarray()

            # Проверка, обучена ли модель
            if self.alphas_sv is None or len(self.alphas_sv) == 0:
                return np.zeros(X.shape[0])

            # Предсказание для каждого образца
            predictions = np.zeros(X.shape[0])
            for i in range(X.shape[0]):
                fx = self.decision_function(X[i])
                predictions[i] = 1 if fx >= 0 else 0

            return predictions
            ### END YOUR CODE
```

Используйте написанный выше код для обучения классификатора на обучающей выборке. Сделайте предсказания на тестовой выборке. Рассчитайте метрику Accuracy и сохраните ее значение в переменной algo_2_own_accuracy.

```
### BEGIN YOUR CODE

# Инициализация SVM с полиномиальным ядром степени 2
# Оптимизация: уменьшено количество итераций (max_iter=50) для ускорения
own_svm = SVM(C=1.0, degree=2, max_iter=50)

# Обучение модели на полном наборе данных (внутри fit будет использована
подвыборка)
own_svm.fit(X_train, y_train)

# Предсказание на тестовой выборке
y_pred_own = own_svm.predict(X_test)
```

```
# Расчет точности
algo_2_own_accuracy = accuracy_score(y_test, y_pred_own)
print(f"Точность самостоятельно реализованного SVM: {algo_2_own_accuracy:.4f}")

### END YOUR CODE
```

```
Точность самостоятельно реализованного SVM: 0.6232
```

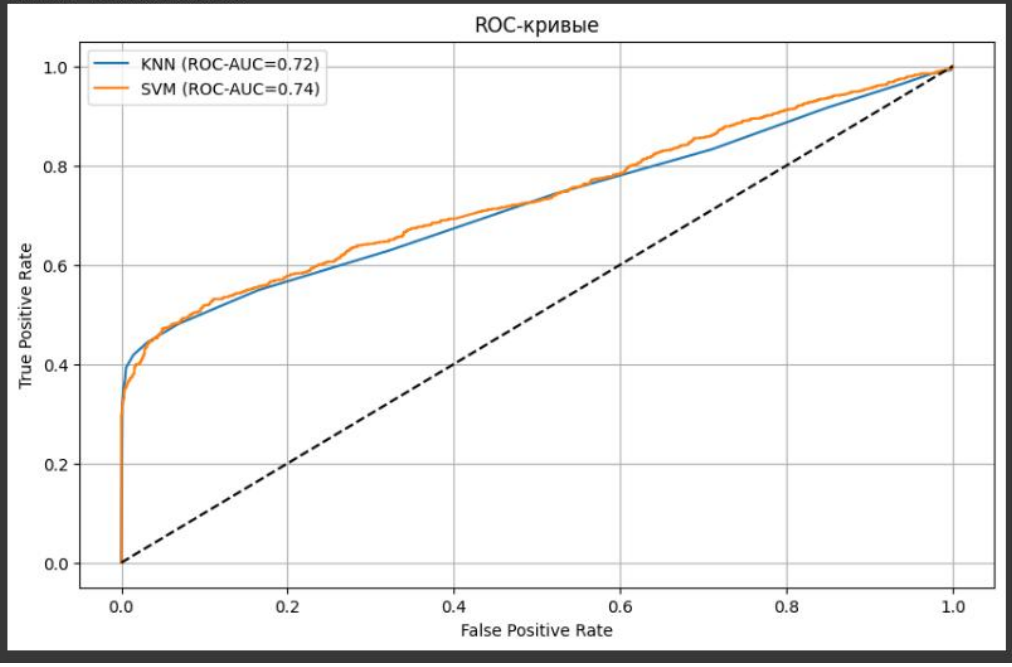
8. Сравнить разные методы (1 балл)

Рассчитайте для каждого алгоритма и его реализации значения метрик Precision, Recall, F1, ROC-AUC, а также постройте ROC-кривую. Сравните результаты работы алгоритмов, сделайте выводы.

```
### BEGIN YOUR CODE
knn = KNNClassifier(17)
svm_model = svm.SVC(kernel="poly", degree=2, probability=True)

knn.fit(X_train, y_train)
svm_model.fit(X_train, y_train)

knn_predict =    knn.predict(X_test)
knn_probs = knn.predict_proba(X_test)

svm_predict =    svm_model.predict(X_test)
svm_probs = svm_model.predict_proba(X_test)

precision_knn = precision_score(y_test,knn_predict)
recall_knn = recall_score(y_test, knn_predict)
f1_knn = f1_score(y_test, knn_predict)
roc_auc_knn = roc_auc_score(y_test,knn_probs[:,1])

print("KNN:")
print(f"Precision: {precision_knn}")
print(f"Recall: {recall_knn}")
print(f"F1: {f1_knn}")
print(f"ROC-AUC: {roc_auc_knn}\n")

# SVM
precision_svm = precision_score(y_test, svm_predict)
recall_svm = recall_score(y_test, svm_predict)
f1_svm = f1_score(y_test, svm_predict)
roc_auc_svm = roc_auc_score(y_test, svm_probs[:,1])

print("SVM:")
print(f"Precision: {precision_svm}")
```

```
print(f"Recall: {recall_svm}")
print(f"F1: {f1_svm}")
print(f"ROC-AUC: {roc_auc_svm}")

# Построение ROC-кривых
fpr_knn, tpr_knn, _ = roc_curve(y_test, knn_probs[:,1])
fpr_svm, tpr_svm, _ = roc_curve(y_test, svm_probs[:,1])

# Визуализация ROC-кривых
plt.figure(figsize=(10, 6))
plt.plot(fpr_knn, tpr_knn, label=f'KNN (ROC-AUC={roc_auc_knn:.2f})')
plt.plot(fpr_svm, tpr_svm, label=f'SVM (ROC-AUC={roc_auc_svm:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  # Линия случайного классификатора
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.title('ROC-кривые')
plt.grid()
plt.show()

# Вывод результатов метрик

### END YOUR CODE
```

```
KNN:
Precision: 0.7396396396396396
Recall: 0.6291187739463602
F1: 0.6799171842650104
ROC-AUC: 0.7233010124360538

SVM:
Precision: 0.783433133732535
Recall: 0.6015325670498084
F1: 0.6805374945817079
ROC-AUC: 0.7368586656392474
```



У SVM на 0.045 выше Precision, а у kNN на 0.02 выше Recall. Кривые ROC схожие, по ROC-AUC лучшее значение показывает SVM, обгоняя KNN на 0.015. Но в целом оба метода показывают очень близкие результаты.

9. Опишите полученные результаты (1 балл)

Напишите краткие выводы объемом в один абзац, ориентированные на нетехническую аудиторию (например, на вашего менеджера или начальника). Сосредоточьтесь на следующих вопросах:

- Как вы можете эффектно и эффективно представить ваше решение для проблемы предсказания значения целевой переменной из вашего датасета?
- Что вы узнали о проблеме на данный момент?
- Как можно улучшить ваше решение?

Для прогнозирования доставки груза в срок с помощью данного датасета я использовал такие методы как написанный мной "k ближайших соседей" и библиотечный "Метод опорных векторов с полиномиальным ядром степени 2". На данный момент оба метода показывают схожие результаты. Однако они далеки от идеала и их возможно улучшить, попробовав подобрать более подходящее ядро для SVM и/или попробовать удалить неинформативные признаки.

Выход

В данной лабораторной работе я реализовал алгоритм k-ближайших соседей и самостоятельно разработал метод опорных векторов с полиномиальным ядром степени 2, что позволило глубоко понять принципы классификации. Работа охватила полный цикл машинного обучения: от загрузки данных, разведочного анализа и предобработки (стандартизация и one-hot кодирование) до реализации моделей и оценки их эффективности (точность, precision, recall, F1-score и ROC-кривые). Самостоятельная реализация SVM особенно укрепила понимание опорных векторов, ядерных методов и оптимизации SMO. Сравнение производительности алгоритмов на датасете e-commerce наглядно продемонстрировало практические аспекты выбора моделей.