|  |  |  |
| --- | --- | --- |
|  | Изображение выглядит как зарисовка, рисунок, символ, эмблема  Автоматически созданное описание |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| **Институт комплексной безопасности и специального приборостроения**  Кафедра КБ-14 «Технологии искусственного интеллекта и анализ данных»  Дисциплина «Введение в машинное обучение» | | |

**Отчёт**

«Практическая работа №1»

Выполнил:

Студент 3 курса

Группа БСБО-06-22

Шифр 22Б1240

Белявцева Екатерина Александровна

Москва, 2024

# Задание

## Часть 1

Построить прогностическую модель для набора данных в файле, проверить связь признаков, построить прогностические модели и модели тренда линейного и квадратичного. Оценить погрешность. (Можно использовать язык программирования)

## Часть 2

Разработать прогностическую модель для набора данных диабетических обследований diabetes.txt. Использовать логистическую регрессию и метод максимального правдоподобия. Коэффициенты логистической регрессии найти с помощью метода градиентного спуска, который необходимо запрограммировать вручную. Разбить выборку на обучающую и тестовую. Вычислить точность классификации. Применить отбор признаков на основе корреляции: выбрать наилучшее признаковое пространство, имеющее на два измерения меньше исходного. Построить новую модель и вычислить точность классификации. (Используя Pyhton или любой другой язык программирования)

# Ход работы

## Часть 1

Для выполнения первой части был использован язык программирования Python в приложении PyCharm. Для выполнения всех требований задания был написан единый код (см. Листинг 1), который выводит 4 графика: графический анализ исходных данных, связь признаков (реализована с помощью матрицы корреляции), прогностические модели и модели тренда линейного и квадратичного. Также после закрытия окна с графиками выводятся погрешности.

Листинг 1. Код для выполнения части 1

|  |
| --- |
| import pandas as pd import seaborn as sns import matplotlib.pyplot as plt from sklearn.linear\_model import LinearRegression from sklearn.preprocessing import PolynomialFeatures from sklearn.metrics import mean\_squared\_error from sklearn.metrics import mean\_absolute\_error  # Загрузка данных из Excel файла data = pd.read\_excel('C:/Users/neverElz/Downloads/practice\_1.xlsx')  # Изменение к данных для студенческого 22Б1240. data['x'] = data['x'] + 4 data['y'] = data['y'] + 1 data['z'] = data['z'] + (4 + 1) / 2  # Создание фигуры для нескольких диаграмм fig, schema = plt.subplots(2, 2, figsize=(14, 12))  # Диаграмма 1: линейные графики для x, y и z schema[0, 0].plot(data['Периоды времени'], data['x'], marker='.', color='r', label='x') schema[0, 0].plot(data['Периоды времени'], data['y'], marker='.', color='g', label='y') schema[0, 0].plot(data['Периоды времени'], data['z'], marker='.', color='b', label='z') # Настройка графика schema[0, 0].set\_title('Графический анализ исходных данных') schema[0, 0].set\_xlabel('Год') schema[0, 0].set\_ylabel('тыс. шт') schema[0, 0].set\_xticks(ticks=data['Периоды времени']) schema[0, 0].tick\_params(axis='x', rotation=45) schema[0, 0].legend() schema[0, 0].grid(True)  # Вычисление матрицы корреляции correlation\_matrix = data[['x', 'y', 'z']].corr() # Диаграмма 2: тепловая карта корреляции sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', ax=schema[0, 1]) schema[0, 1].set\_title('Матрица корреляции')  # Подготовка данных для линейной регрессии years = data[['Периоды времени']] X = data['x'] Y = data['y'] Z = data['z']  # Линейная регрессия для x, y и z prediction\_x = LinearRegression().fit(years, X).predict(years) prediction\_y = LinearRegression().fit(years, Y).predict(years) prediction\_z = LinearRegression().fit(years, Z).predict(years)  # Квадратичная модель тренда polynomial\_features = PolynomialFeatures(degree=2) poly\_x = polynomial\_features.fit\_transform(years) quadratic\_model\_x = LinearRegression().fit(poly\_x, X).predict(poly\_x) poly\_y = polynomial\_features.fit\_transform(years) quadratic\_model\_y = LinearRegression().fit(poly\_y, Y).predict(poly\_y) poly\_z = polynomial\_features.fit\_transform(years) quadratic\_model\_z = LinearRegression().fit(poly\_z, Z).predict(poly\_z)  # Линейно предсказанные данные для x, y и z schema[1, 0].plot(data['Периоды времени'], prediction\_x, marker='.', color='r', linewidth=2, label='Прогноз для x') schema[1, 0].plot(data['Периоды времени'], prediction\_y, marker='.', color='g', linewidth=2, label='Прогноз для y') schema[1, 0].plot(data['Периоды времени'], prediction\_z, marker='.', color='b', linewidth=2, label='Прогноз для z')  schema[1, 1].plot(data['Периоды времени'], quadratic\_model\_x, marker='.', color='r', label='Квадратичный тренд для x') schema[1, 1].plot(data['Периоды времени'], quadratic\_model\_y, marker='.', color='g', label='Квадратичный тренд для Y') schema[1, 1].plot(data['Периоды времени'], quadratic\_model\_z, marker='.', color='b', label='Квадратичный тренд для Z')  # Настройка графика schema[1, 0].set\_title('Модель линейного тренда') schema[1, 0].set\_xlabel('Периоды времени') schema[1, 0].set\_ylabel('Значение') schema[1, 0].set\_xticks(ticks=data['Периоды времени']) schema[1, 0].tick\_params(axis='x', rotation=45) schema[1, 0].legend() schema[1, 0].grid(True)  # Настройка графика schema[1, 1].set\_title('Модель квадратичного тренда') schema[1, 1].set\_xlabel('Периоды времени') schema[1, 1].set\_ylabel('Значение') schema[1, 1].set\_xticks(ticks=data['Периоды времени']) schema[1, 1].tick\_params(axis='x', rotation=45) schema[1, 1].legend() schema[1, 1].grid(True)  # Отображение графиков plt.tight\_layout() plt.show()   # Функция для вычисления ошибок MSE и MAE def calculate\_errors(real\_values, predicted\_values, name):  mse = mean\_squared\_error(real\_values, predicted\_values)  mae = mean\_absolute\_error(real\_values, predicted\_values)  print(f"{name}: MSE = {mse}, MAE = {mae}")   # Расчет ошибок для линейных моделей calculate\_errors(X, prediction\_x, 'Линейная модель x') calculate\_errors(Y, prediction\_y, 'Линейная модель y') calculate\_errors(Z, prediction\_z, 'Линейная модель z')  # Расчет ошибок для квадратичных моделей calculate\_errors(X, quadratic\_model\_x, 'Квадратичная модель x') calculate\_errors(Y, quadratic\_model\_y, 'Квадратичная модель y') calculate\_errors(Z, quadratic\_model\_z, 'Квадратичная модель z') |

Для корректной работы кода, на основании приведённой в задании таблицы «Задание на практическую работу 1 пустографка.xls», в отдельном файле Excel была построена таблица «practice\_1.xlsx» (см. Рисунок 1).

Изображение выглядит как текст, число, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1. Содержание файла «practice\_1.xlsx»

Далее будет представлен подробный анализ полученных графиков.

Первый график – графический анализ исходных данных (см. Рисунок 2). Каждая переменная представлена отдельной цветной линией (красный для x, зеленый для y, синий для z). График демонстрирует, как меняются значения переменных с течением времени.

Изображение выглядит как линия, График, диаграмма, текст

Автоматически созданное описание

Рисунок 2. Графический анализ исходных данных

На данном графике можно наблюдать следующие тенденции:

1. Показатель x сначала растет достаточно быстрыми темпами, достигая пика в районе 2019 года. После этого показатель немного снижается и практически стабилизируется. Это указывает на спад после активного роста в начале периода.
2. Показатель y вначале растет очень медленно, достигая максимальных значений в период 2017-2018 годов. Затем наблюдается резкое снижение, и к концу рассматриваемого периода значения остаются на уровне, значительно ниже максимума. Поведение показателя y характеризуется явным циклом подъема и падения.
3. Показатель z демонстрирует относительно равномерный рост на протяжении всего периода времени. Начиная с 2017-2018 года, заметен более выраженный рост, который продолжается вплоть до 2022 года. Это может свидетельствовать о положительном тренде и увеличении данного показателя с течением времени.

Следующий график – матрица корреляции (см. Рисунок 3) в виде тепловой карты, которая показывает степень корреляции между переменными x, y, и z. Значения корреляции отображены числовыми значениями (от -1 до 1), где 1 указывает на полную прямую зависимость, -1 — на полную обратную зависимость, а 0 — на отсутствие зависимости.

Матрица корреляции, отображённая на тепловой карте (см. Рисунок 3), подтверждает взаимосвязи между переменными. Высокий коэффициент корреляции между x и z может указывать на обратную связь между ними, а отсутствие корреляции между x и y подтверждает, что переменные изменяются независимо друг от друга.

Изображение выглядит как текст, снимок экрана, диаграмма, Прямоугольник

Автоматически созданное описание

Рисунок 3. Взаимосвязи между переменными. Матица корреляции

Следующий график – модель линейного тренда (см. Рисунок 4). На этом графике отображены предсказанные значения переменных x, y и z с использованием линейной регрессии.

Прогноз для x — показывает небольшое стабильное увеличение значения на протяжении всего периода. Прогноз для y — линия имеет слабо выраженный нисходящий тренд, показывает равномерное уменьшение значения на всём протяжении периода. Прогноз для z — демонстрирует сильный и резкий рост, относительно х и у.

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

Рисунок 4. Линейная регрессия

Последний график – модель квадратичного тренда (см. Рисунок 5). Этот график отображает прогнозы на основе квадратичной регрессии, которая позволяет учитывать нелинейные изменения данных.

Прогноз для x — начинается с роста, но затем модель показывает снижение после пика. Это указывает на то, что модель учитывает некоторое замедление и возможное падение значения x в будущем. Прогноз для y — сначала отображается небольшой рост, достигающий максимума немного раньше середины периода, после чего начинается спад. Судя по данному прогнозу, в будущем значение у будет уменьшаться. Прогноз для z —начинается с небольшого падения и после минимума резко сильно возрастает, что указывает на рост значения в будущем.

Изображение выглядит как линия, График, текст, диаграмма

Автоматически созданное описание

Рисунок 5. Квадратичная регрессия

После закрытия окна с графиками, в терминале выводятся погрешности (см. Рисунок 6). Для каждой переменной в каждой модели (линейной и квадратичной) рассчитываются ошибки MSE (среднеквадратичная ошибка) и MAE (средняя абсолютная ошибка). Эти метрики позволяют оценить, насколько сильно предсказанные значения отличаются от реальных данных.

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

Рисунок 6. Погрешности или ошибки предсказаний

## Часть 2

На основе текстового документа diabetes.txt была разработана и построена прогностическая модель. Для этого была использована логистическая регрессия. Коэффициенты логистической регрессии были найдены с помощью метода градиентного спуска с использованием MLE (методом максимального правдоподобия). Для реализации обучения выборка была разбита на обучающую и тестовую. Также была вычислена точность классификации. Код представлен ниже в листинге 2. При использовании данного кода точность (Accuracy) составляет 0.88.

Листинг 2. Прогностическая модель для набора данных диабетических обследований diabetes.txt

|  |
| --- |
| import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  # Загрузка данных с указанием кодировки  data = pd.read\_csv('diabetes.txt', delimiter='\t', encoding='windows-1251')  # Определение признаков и целевой переменной  X = data.drop(columns=['Диагноз']) # Признаки  y = data['Диагноз'] # Целевая переменная  # Масштабирование данных  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X) # Нормализация признаков  # Разделение данных на тренировочные и тестовые  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.01, random\_state=42)  # Добавление столбца единиц для theta\_0  X\_train = np.hstack((np.ones((X\_train.shape[0], 1)), X\_train)) # Обучающая выборка  X\_test = np.hstack((np.ones((X\_test.shape[0], 1)), X\_test)) # Тестовая выборка  # Гиперпараметры  learning\_rate = 0.01 # Шаг обучения  n\_iterations = 1000 # Количество итераций  # Инициализация параметров модели  theta = np.zeros(X\_train.shape[1]) # Вектор параметров  # Сигмоида  def sigmoid(z):  return 1 / (1 + np.exp(-z))  # Метод максимального правдоподобия (MLE) с использованием градиентного спуска  def fit\_logistic\_regression(X, y, theta, learning\_rate, n\_iterations):  for i in range(n\_iterations):  z = np.dot(X, theta) # Линейная комбинация признаков  y\_pred = sigmoid(z) # Предсказанные вероятности  # Вычисление градиентов на основе MLE  gradient = (1 / len(y)) \* np.dot(X.T, (y\_pred - y))  # Обновление параметров  theta -= learning\_rate \* gradient  return theta  # Обучение модели  theta = fit\_logistic\_regression(X\_train, y\_train, theta, learning\_rate, n\_iterations)  # Оценка точности  y\_test\_pred\_prob = sigmoid(np.dot(X\_test, theta)) # Предсказанные вероятности на тестовой выборке  y\_test\_pred = (y\_test\_pred\_prob >= 0.5).astype(int) # Преобразование вероятностей в классы  accuracy = np.mean(y\_test\_pred == y\_test)  print(f'Accuracy: {accuracy:.2f}') |

Далее был осуществлён отбор признаков на основе корреляции: были выбраны лучшие признаки и исключены 2 худших. После этого была построена новая модель и снова вычислена точность классификации. Ниже представлен обновлённый код с вышеописанными изменениями (см. Листинг 3.). При использовании данного кода точность (Accuracy) составляет 1.

Листинг 3. Обновлённый код и отбором признаков на основе корреляции

|  |
| --- |
| import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import StandardScaler  # Загрузка данных с указанием кодировки data = pd.read\_csv('diabetes.txt', delimiter='\t', encoding='windows-1251')  # Определение признаков и целевой переменной X = data.drop(columns=['Диагноз']) # Признаки y = data['Диагноз'] # Целевая переменная  # Вычисление корреляции correlation\_matrix = X.corrwith(y).abs() # Абсолютные значения корреляции correlation\_matrix = correlation\_matrix.sort\_values(ascending=False) # Сортировка по убыванию  # Выбор наилучшего признакового пространства: исключение 2 наименее коррелирующих признаков selected\_features = correlation\_matrix.index[:-2] # Все, кроме двух последних X\_selected = X[selected\_features] # Новая матрица признаков  # Масштабирование данных scaler = StandardScaler() X\_scaled = scaler.fit\_transform(X\_selected) # Нормализация признаков  # Разделение данных на тренировочные и тестовые X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.01, random\_state=42) # Добавление столбца единиц для theta\_0 X\_train = np.hstack((np.ones((X\_train.shape[0], 1)), X\_train)) # Обучающая выборка X\_test = np.hstack((np.ones((X\_test.shape[0], 1)), X\_test)) # Тестовая выборка  # Гиперпараметры learning\_rate = 0.01 # Шаг обучения n\_iterations = 1000 # Количество итераций  # Инициализация параметров модели theta = np.zeros(X\_train.shape[1]) # Вектор параметров   # Сигмоида def sigmoid(z):  return 1 / (1 + np.exp(-z))   # Метод максимального правдоподобия (MLE) с использованием градиентного спуска def fit\_logistic\_regression(X, y, theta, learning\_rate, n\_iterations):  for i in range(n\_iterations):  z = np.dot(X, theta) # Линейная комбинация признаков  y\_pred = sigmoid(z) # Предсказанные вероятности   # Вычисление градиентов на основе MLE  gradient = (1 / len(y)) \* np.dot(X.T, (y\_pred - y))   # Обновление параметров  theta -= learning\_rate \* gradient  return theta   # Обучение модели theta = fit\_logistic\_regression(X\_train, y\_train, theta, learning\_rate, n\_iterations)  # Оценка точности y\_test\_pred\_prob = sigmoid(np.dot(X\_test, theta)) # Предсказанные вероятности на тестовой выборке y\_test\_pred = (y\_test\_pred\_prob >= 0.5).astype(int) # Преобразование вероятностей в классы accuracy = np.mean(y\_test\_pred == y\_test) print(f'Accuracy: {accuracy:.2f}') |