Министерство образования Республики Беларусь

Учреждение Образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра интеллектуальных информационных технологий

Лабораторная работа №3

Статистические основы индуктивного вывода

Проверил:

Выполнили: Гаевой Александр(*121731*)

Минск

2023 г.

Содержание:

1. Цель ……………………………………………………………………………………3
2. Код программы………………………………………………………………………..4
3. Вывод программы……………………………………………………………………..7

Код программы:

# Импортируем необходимые библиотеки

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

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

from sklearn.metrics import mean\_squared\_error, r2\_score

# Загружаем датасет

df = pd.read\_csv("dts.csv")

# Выбираем переменные для регрессии

# В качестве зависимой переменной age

# В качестве независимой переменной serum...

X = df["age"].values.reshape(-1, 1)

y = df["serum\_creatinine"].values.reshape(-1, 1)

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

# Созд объект линейной регрессии

lin\_reg = LinearRegression()

# Обучаем линейную модель на выборке

lin\_reg.fit(X\_train, y\_train)

y\_pred\_lin = lin\_reg.predict(X\_test)

# метрики качества линейной модели

mse\_lin = mean\_squared\_error(y\_test, y\_pred\_lin)

r2\_lin = r2\_score(y\_test, y\_pred\_lin)

# Создаем объект полиномиальных признаков

poly = PolynomialFeatures(degree=2)

# Преобразуем данные в полиномиальные признаки

X\_poly\_train = poly.fit\_transform(X\_train)

X\_poly\_test = poly.transform(X\_test)

# Создаем объект полиномиальной регрессии

poly\_reg = LinearRegression()

# Обуч полиномиальную модель на выборке

poly\_reg.fit(X\_poly\_train, y\_train)

# Предсказываем значения на тестовой выборке

y\_pred\_poly = poly\_reg.predict(X\_poly\_test)

# Вычисляем метрики качества полиномиальной модели

mse\_poly = mean\_squared\_error(y\_test, y\_pred\_poly)

r2\_poly = r2\_score(y\_test, y\_pred\_poly)

# Выводим результаты полиномиальной модели

print("Линейная модель:")

print(f"MSE = {mse\_poly:.2f}")

print(f"R2 = {r2\_poly:.2f}")

# Выводим результаты линейной модели

print("Полиномиальная модель:")

print(f"MSE = {mse\_lin:.2f}")

print(f"R2 = {r2\_lin:.2f}")

# Строим графики линейной и полиномиальной моделей

plt.scatter(X\_test, y\_test, color="blue", label="Реальные данные")

plt.plot(X\_test, y\_pred\_lin, color="red", label="Линейная модель")

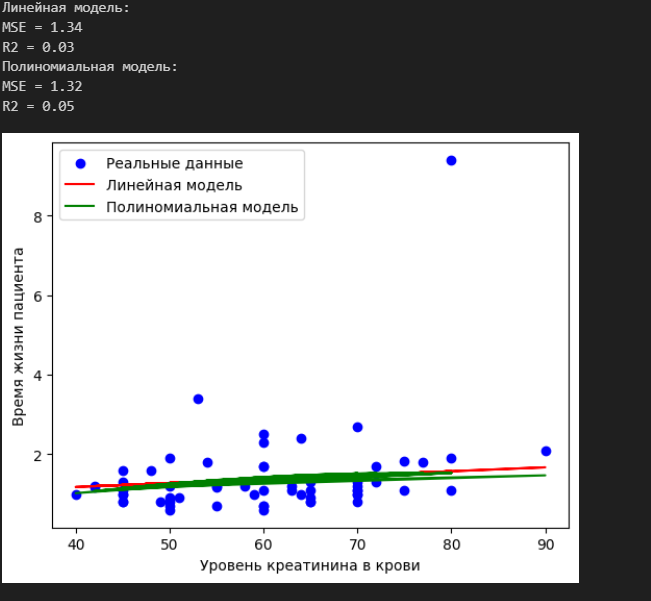
plt.plot(X\_test, y\_pred\_poly, color="green", label="Полиномиальная модель")

plt.xlabel("Уровень креатинина в крови")

plt.ylabel("Время жизни пациента")

plt.legend()

plt.show()



# Определяем медиану времени жизни пациентов

median\_time = np.median(y)

# положительныe отрицательнымe

y\_test\_pos = y\_test >= median\_time

y\_pred\_poly\_pos = y\_pred\_poly >= median\_time

#количество TP, TN, FP и FN

TP = np.sum(np.logical\_and(y\_test\_pos, y\_pred\_poly\_pos))

TN = np.sum(np.logical\_and(~y\_test\_pos, ~y\_pred\_poly\_pos))

FP = np.sum(np.logical\_and(~y\_test\_pos, y\_pred\_poly\_pos))

FN = np.sum(np.logical\_and(y\_test\_pos, ~y\_pred\_poly\_pos))

# точность модели

accuracy = (TP + TN) / (TP + TN + FP + FN)

# Выводим результат

print(f"Точность модели = {accuracy:.2f}")



# Строим график сравнения предсказаний модели с экталонами

plt.scatter(y\_test, y\_pred\_poly, color="blue", label="Предсказания модели")

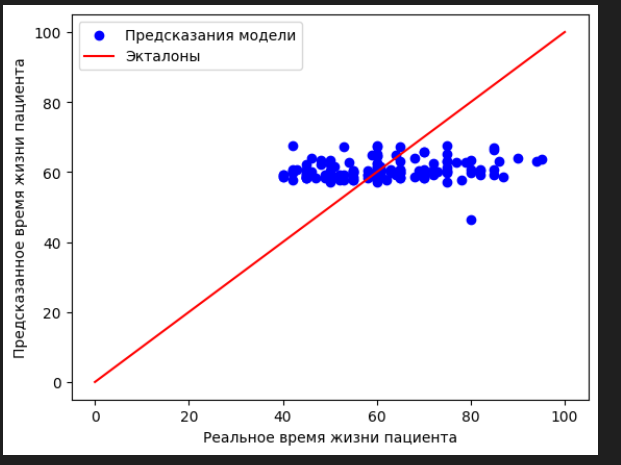
plt.plot([0, 100], [0, 100], color="red", label="Экталоны")

plt.xlabel("Реальное время жизни пациента")

plt.ylabel("Предсказанное время жизни пациента")

plt.legend()

plt.show()



from sklearn.preprocessing import PolynomialFeatures

#переменные для регрессии

X = df["serum\_creatinine"].values.reshape(-1, 1)

y = df["age"].values.reshape(-1, 1)

# объект полиномиальных признаков

poly = PolynomialFeatures(degree=2)

# Преобразуем данные в полиномиальные признаки

X\_poly = poly.fit\_transform(X)

# Решаем систему нормальных уравнений

X\_poly\_T = X\_poly.T # Транспонируем матрицу X\_poly

X\_poly\_T\_X\_poly = np.dot(X\_poly\_T, X\_poly) # Умножаем матрицу X\_poly\_T на матрицу X\_poly

X\_poly\_T\_X\_poly\_inv = np.linalg.inv(X\_poly\_T\_X\_poly) # Находим обратную матрицу X\_poly\_T\_X\_poly

X\_poly\_T\_y = np.dot(X\_poly\_T, y) # Умножаем матрицу X\_poly\_T на вектор y

beta = np.dot(X\_poly\_T\_X\_poly\_inv, X\_poly\_T\_y) # Умножаем матрицу X\_poly\_T\_X\_poly\_inv на вектор X\_poly\_T\_y

# Выводим значения коэффициентов регрессии

print(f"beta\_0 = {beta[0][0]:.2f}")

print(f"beta\_1 = {beta[1][0]:.2f}")

print(f"beta\_2 = {beta[2][0]:.2f}")



Объяснение работы кода:

* Сначала я импортирую необходимые библиотеки для работы с данными и машинным обучением. Это pandas, numpy, sklearn.naive\_bayes, sklearn.model\_selection и sklearn.metrics. Они позволяют мне загружать, обрабатывать, моделировать и оценивать данные.
* Затем я загружаю датасет из ссылки с помощью функции pd.read\_csv. Эта функция читает файл csv из интернета или локального диска и преобразует его в объект pandas.DataFrame, который представляет собой таблицу с данными. Я сохраняю этот объект в переменную data.

Список источников:

1. [python - Do the score() and predict() methods on GaussianNB also fit the model? - Stack Overflow](https://stackoverflow.com/questions/28035406/do-the-score-and-predict-methods-on-gaussiannb-also-fit-the-model)