МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
«МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ»

Лабораторна робота № 4

«Лінійна регресія»

Виконав:	Гоша Давід	**Перевірив**:	
Група	ІПЗ-33	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		
2022			

Варіант 4

Тема (завдання) для дослідження – Проста лінійна регресія.

Математичний опис моделі лінійної регресії

Широко розповсюдженою задачею обробки даних є представлення їх сукупності деякою функцією $y(x)$. Задача регресії найчастіше всього полягає у отриманні параметрів цієї функції такими, щоб функція наближала «хмарку» вихідних точок (заданих векторами VX та XY) із найменшою середньоквадратичною похибкою. У цьому випадку говорять про регресію методом найменших квадратів.

Найчастіше використовується лінійна регресія, в якій функція описує відрізок прямої та має вид: До лінійної регресії можна звести багато вид:

$$y(x) = a + bx$$

До лінійної регресії можна звести багато видів нелінійної регресії при залежностях виду $y(x)$.

Математично постановка задачі регресії зводиться до наступного. Нехай є набір точно визначених значень і відповідних їм неточних значень x_i. Припустимо, що існує деяка залежність $(x_i, a_0, a_1, \ \dots \ a_k)$, яка може розглядатись як наближення до залежності $y(x)$, чиї точки представлені як $y_i(x_i)$. Таким чином, ми маємо право записати:

$$= f(x_i, a_0, a_1, \qquad \dots \ a_k) + \xi_i$$

Тут ξ_i незалежні випадкові величини із нормальним законом розподілення, які визначають похибку завдання уі. Зазвичай їх вважають наслідком помилок експерименту. Задача регресії полягає у тому, щоб знайти параметри $a_0, a_1,$ $\dots \ a_k$ такими, при яких представлення $y(x)$ нашою функцією $f(x)$ мало найменшу середньоквадратичну похибку. Для цього треба мінімізувати функцію:

$$\Phi(a_0, a_1 \dots \ a_k) = \sum (f(x_i, a_0, a_1 \dots \ a_k) - y_i)^2$$

Наприклад, для найширше розповсюдженої лінійної регресії, коли $f(x) = a_0 + a_1 x$ (часто замінюють = a та = b), треба мінімізувати наступний вираз:

$$\Phi(a_0, a_1 \dots \ a_k) = \sum (a_0 - a_1 x_i - y_i)^2$$

Якщо прирівняти до нуля, то для лінійної регресії можна знайти її параметри та у явній формі:

$$a_0 = \frac{\sum_{i=1}^{n} y_i \ \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Аналогічним чином можна отримати вираз і для інших видів регресії: поліноміальної, експоненціальної, логарифмічної та ін. З огляду на складність відповідних виразів вони не наводяться - деякі з формул можна знайти у літературі. Багато задач нелінійної регресії можна звести до розглянутої вище лінійної, використовуючи аналогічні перестановки. Але Numpi для багатьох видів регресії задає потрібні формули явно, що робить перетворення даних непотрібними.

Комп'ютерна програма мовою Python без використання спеціалізованої бібліотеки

```
def np_linereg():
     #линия регрессии
    slope, intercept, r_value, p_value, std_err = stats.linregress(x,y)

    line = slope*x+intercept

    #создание кроссплота
    fig = plt.figure(figsize=(10,7))
    ax = plt.subplot(111)

    RSS = np.sum(np.square(line - y))
    MSE = np.mean(np.square(line - y))
    RMSE = np.sqrt(np.mean(np.square(line - y)))

    print(' residual sum of squares is : '+ str(RSS))
    print(' mean squared error is : '+ str(MSE))
    print(' root mean squared error is : '+ str(RMSE))

    plt.scatter(x,y, s=50, c=numb)
    plt.plot(x, line, 'r', label='y={:.2f}x+{:.2f}'.format(slope,intercept))
    plt.plot([], [], ' ', label='R_sq = '+'{:.2f}'.format(r_value**2))
    plt.plot([], [], ' ', label='RSS = '+'{:.2f}'.format(RSS))
    plt.plot([], [], ' ', label='MSE = '+'{:.2f}'.format(MSE))
    plt.plot([], [], ' ', label='RMSE = '+'{:.2f}'.format(RMSE))

    plt.grid(True)
    plt.legend(fontsize=12)
    plt.colorbar()
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.show()
```

Комп'ютерна програма мовою Python з використанням спеціалізованої бібліотеки

```
def scipi_mth(x,y):
    # Create linear regression object
    regr = LinearRegression()
    x = x.reshape(-1, 1)
    y = y.reshape(-1, 1)
    # Train the model using the training sets
    regr.fit(x, y)

    # Make predictions using the testing set
    diabetes_y_pred = regr.predict(x)

    # The coefficients
    print("Coefficients: \n", regr.coef_)
    # The mean squared error
    print("Mean squared error: %.2f" % mean_squared_error(y, diabetes_y_pred))
    # The coefficient of determination: 1 is perfect prediction
    print("Coefficient of determination: %.2f" % r2_score(y, diabetes_y_pred))

    plt.figure(figsize=(10,7))

    # Plot outputs
    plt.scatter(x, y, color="black", s =50)

    plt.plot(x, diabetes_y_pred, color="blue", linewidth=3,
label='y={:.2f}x+{:.2f}'.format(float(regr.coef_),float(regr.intercept_)))

    plt.plot([], [], ' ', label='RSS = '+'{:.2f}'.format(np.sum((diabetes_y_pred - y) ** 2)))
    plt.plot([], [], ' ', label='MSE = '+'{:.2f}'.format(mean_squared_error(y, diabetes_y_pred)))
    plt.plot([], [], ' ', label='RMSE = '+'{:.2f}'.format(np.sqrt(mean_squared_error(y, diabetes_y_pred))))

    plt.grid(True)
    plt.legend(fontsize=12)
    plt.colorbar()

    plt.xlabel('X')
    plt.ylabel('Y')
    plt.xticks(())
    plt.yticks(())

    plt.show()
```

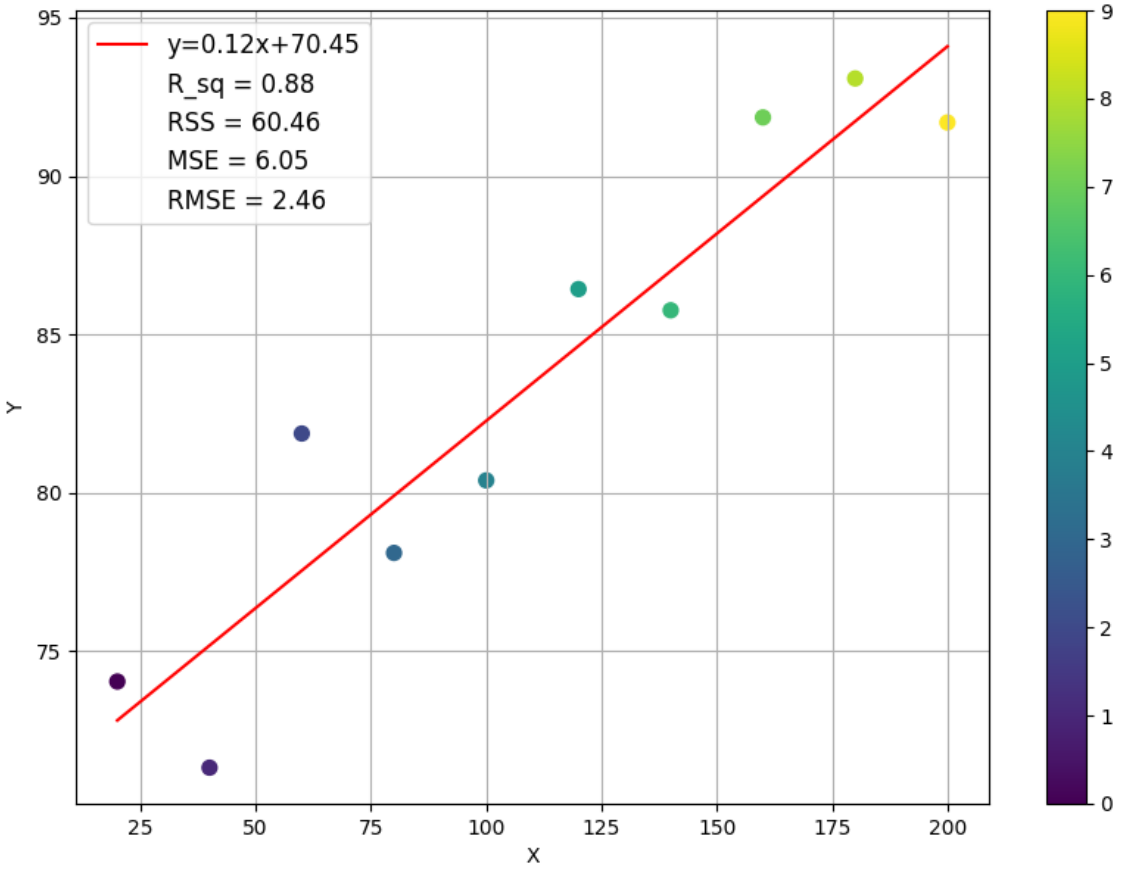
Аналіз результатів

Лінійні моделі – це клас моделей, які широко використовуються на практиці та широко вивчаються протягом останніх кількох десятиліть, коріння яких сягає понад сто років. Лінійні моделі роблять прогноз, використовуючи лінійну функцію вхідних характеристик. Іншими словами, його основна мета — знайти

лінійну функцію, яка виражає зв'язок між залежною (y) і незалежною (s) змінними. Xp представляє p-й предиктор, β_p кількісно визначає зв'язок між цією змінною та відповіддю. Ми інтерпретуємо β_p як середній вплив на Y збільшення Xp на одиницю, утримуючи всі інші предиктори фіксованими. Крім того, β_p можна назвати вивченими коефіцієнтами.

Немає прямої лінії, яка проходить через усі точки даних. Отже, мета тут полягає в тому, щоб підібрати найкращу відповідність прямої лінії, яка намагатиметься мінімізувати похибку між очікуваним і фактичним значенням. Лінійна регресія має дві основні мети. По-перше, оцінка значень залежної змінної через змінні, визначені для впливу на залежну змінну. Другий – визначити, які з незалежних змінних, які, як вважають, впливають на залежну змінну, або як і яким чином це впливає на залежну змінну. Функція втрат для регресії, пропорційна квадрату втрат, які ми зазнаємо, віддаляючись від справжнього значення. Під час навчання моделі ми дбаємо не лише про мінімізацію втрати вибірки, ми дбаємо про мінімізацію втрати всього нашого набору даних.

Почнемо з аналізу результату для побудованої лінії регресії для вбудованоъ функції numpy.



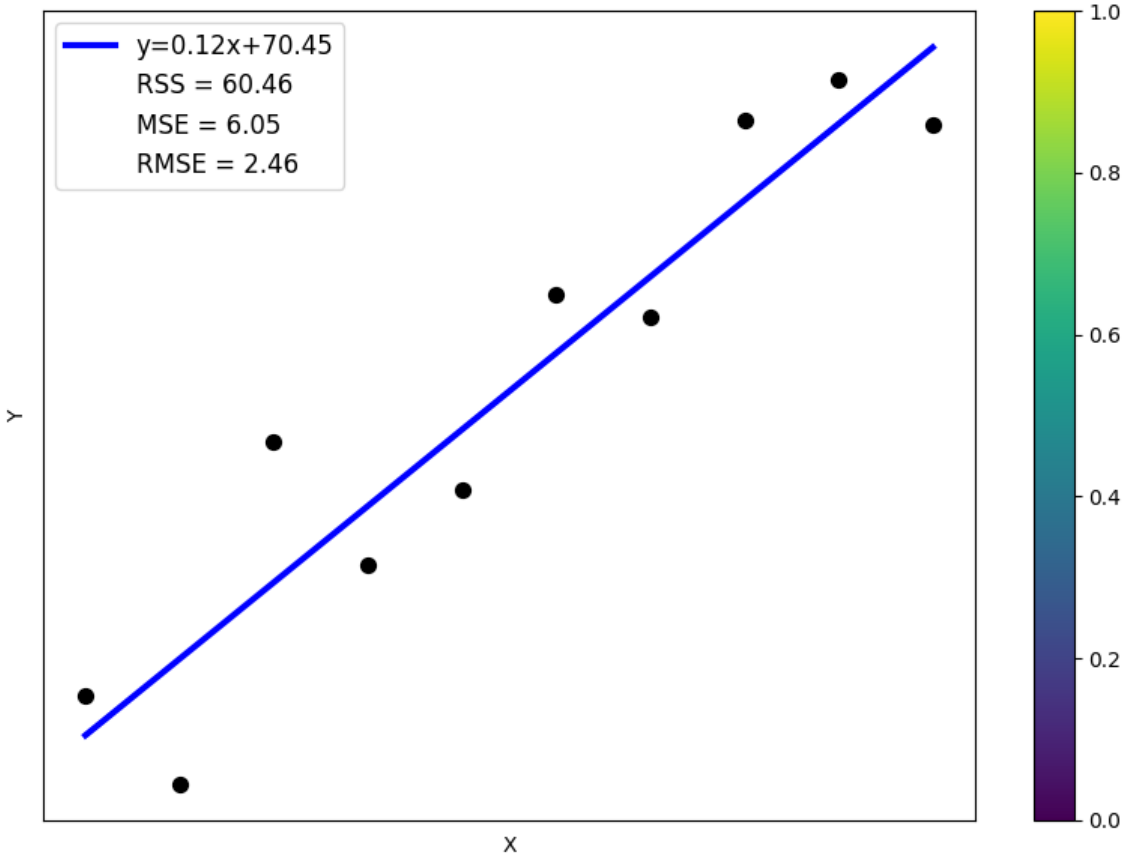
Residual sum of squares (RSS): 60.45656727272744
Mean squared error (MSE): 6.045656727272744

Root mean squared error (RMSE) : 2.4587917210029695

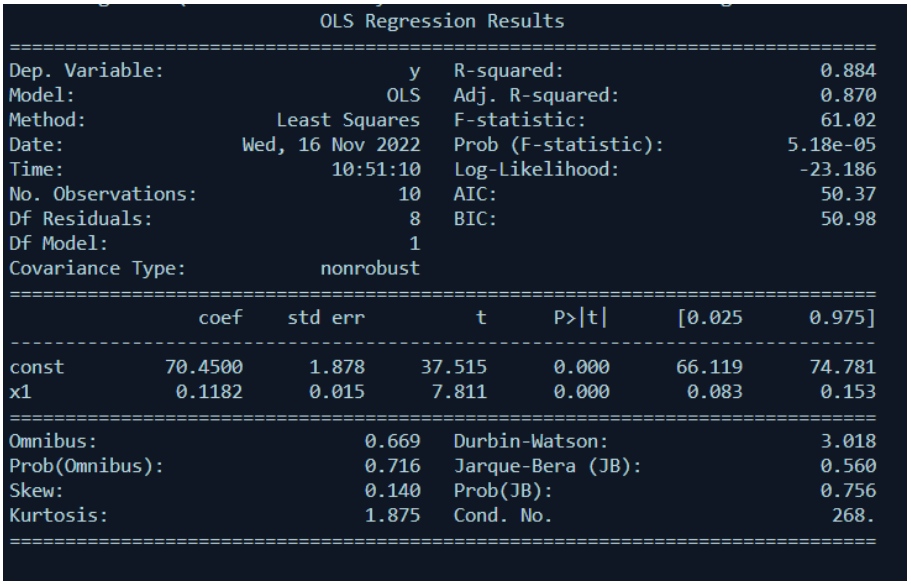
	20	40	60	80	100	120	140	160	180	200
Y	74.04	71.32	81.87	78.1	80.39	86.43	85.76	91.85	93.08	91.69
$Y_{predict}$	72.814	75.178	77.542	79.906	82.270	84.635	86.999	89.363	91.727	94.09

Далі проаналізуємо результати з використанням бібліотеки sklearn.

У наведеному нижче прикладі використовується лише перша функція набору даних діабету, щоб проілюструвати точки даних на двовимірному графіку. На графіку можна побачити пряму лінію, яка показує, як лінійна регресія намагається провести пряму лінію, яка найкраще мінімізує залишкову суму квадратів між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням. Також розраховуються коефіцієнти, залишкова сума квадратів і коефіцієнт детермінації.



Для перевірки результатів скористаємося іншим інстурментом. Функція OLS() модуля statsmodels.api використовується для виконання регресії OLS. Він повертає об’єкт OLS. Потім для цього об’єкта викликається метод fit() для підгонки лінії регресії до даних.



OLS Regression Results

Dep. Variable:	y	R-squared:	0.884
Model:	OLS	Adj. R-squared:	0.870
Method:	Least Squares	F-statistic:	61.02
Date:	Wed, 16 Nov 2022	Prob (F-statistic):	5.18e-05
Time:	10:51:10	Log-Likelihood:	-23.186
No. Observations:	10	AIC:	50.37
Df Residuals:	8	BIC:	50.98
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	70.4500	1.878	37.515	0.000	66.119	74.781
x1	0.1182	0.015	7.811	0.000	0.083	0.153

Omnibus:	0.669	Durbin-Watson:	3.018
Prob(Omnibus):	0.716	Jarque-Bera (JB):	0.560
Skew:	0.140	Prob(JB):	0.756
Kurtosis:	1.875	Cond. No.	268.

Висновок

У цій лабораторній, перш за все, зроблені теоретичні пояснення лінійної регресії. У лабораторній не було цілі роботи над максимальної роботи над кодом регресії. Також було показано, як інтерпретувати результати за допомогою бібліотеки «Statsmodels» для більш глибокого аналізу.

У першій частині реалізовано регресійну модель за допомогою бібліотеки numpy. Було побудовано діаграму та розраховані метрики аналізу похибок а саме :

- Залишкова сума квадратів
- Середня квадратична помилка
- Root mean squared error

Саме після цього ми провели таке саме моделювання за допомогою спеціалізованої бібліотеки та отримали аналогічні, але в деякому сенсі більш точні результати. Саме з цього можна зробити висновки що побудована регресійна модель є правильною.