ЛАБОРАТОРНА РОБОТА № 4

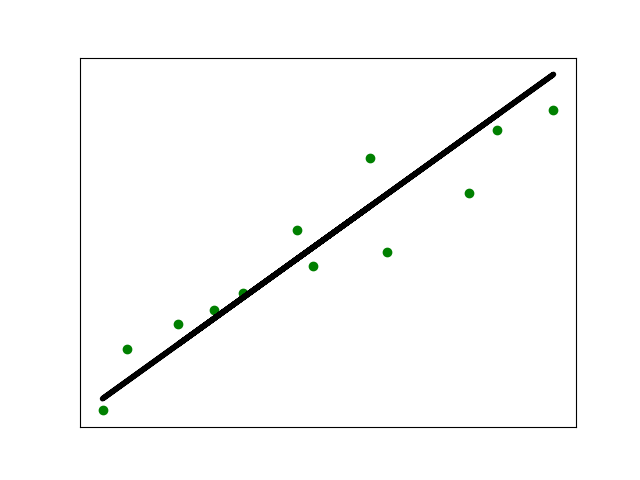
Дослідження методів регресії

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи регресії даних у машинному навчанні.

GitHub = <https://github.com/Ozzornin/AIS/tree/master>

Завдання 1.Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data\_singlevar\_regr.txt.

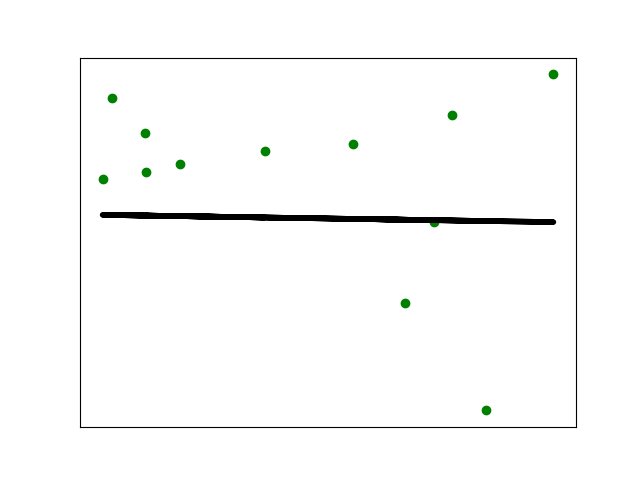
|  |
| --- |
| import pickle  import matplotlib.pyplot as plt  import numpy as np  import sklearn.metrics as sm  from sklearn import linear\_model  # Вхідний файл, який містить дані  input\_file = "data\_singlevar\_regr.txt"  # Завантаження даних  data = np.loadtxt(input\_file, delimiter=",")  X, y = data[:, :-1], data[:, -1]  # Розбивка даних на навчальний та тестовий набори  num\_training = int(0.8 \* len(X))  num\_test = len(X) - num\_training  # Тренувальні дані  X\_train, y\_train = X[:num\_training], y[:num\_training]  # Тестові дані  X\_test, y\_test = X[num\_training:], y[num\_training:]  # Створення об'єкта лінійного регресора  regressor = linear\_model.LinearRegression()  regressor.fit(X\_train, y\_train)  # Прогнозування результату  y\_test\_pred = regressor.predict(X\_test)  # Побудова графіка  plt.scatter(X\_test, y\_test, color="green")  plt.plot(X\_test, y\_test\_pred, color="black", linewidth=4)  plt.xticks(())  plt.yticks(())  plt.savefig("LR\_4\_task\_1.png")  print("Linear regressor performance:")  print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  print(      "Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2)  )  print(      "Explain variance score =",      round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2),  )  print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  # Файл для збереження моделі  output\_model\_file = "model.pkl"  # Збереження моделі  with open(output\_model\_file, "wb") as f:      pickle.dump(regressor, f)  # Завантаження моделі  y\_test\_pred\_new = regressor.predict(X\_test)  print(      "\nNew mean absolute error =",      round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2),  ) |
| Linear regressor performance:  Mean absolute error = 0.59  Mean squared error = 0.49  Median absolute error = 0.51  Explain variance score = 0.86  R2 score = 0.86  New mean absolute error = 0.59 |



Побудова регресійної моделі на основі однієї змінної. Модель показала високі результати з R² = 0.86, що вказує на хороше прогнозування залежності.

Завдання 2. Передбачення за допомогою регресії однієї змінної

|  |
| --- |
| import pickle  import matplotlib.pyplot as plt  import numpy as np  import sklearn.metrics as sm  from sklearn import linear\_model  # Вхідний файл, який містить дані  input\_file = "data\_regr\_3.txt"  # Завантаження даних  data = np.loadtxt(input\_file, delimiter=",")  X, y = data[:, :-1], data[:, -1]  # Розбивка даних на навчальний та тестовий набори  num\_training = int(0.8 \* len(X))  num\_test = len(X) - num\_training  # Тренувальні дані  X\_train, y\_train = X[:num\_training], y[:num\_training]  # Тестові дані  X\_test, y\_test = X[num\_training:], y[num\_training:]  # Створення об'єкта лінійного регресора  regressor = linear\_model.LinearRegression()  regressor.fit(X\_train, y\_train)  # Прогнозування результату  y\_test\_pred = regressor.predict(X\_test)  # Побудова графіка  plt.scatter(X\_test, y\_test, color="green")  plt.plot(X\_test, y\_test\_pred, color="black", linewidth=4)  plt.xticks(())  plt.yticks(())  plt.savefig("LR\_4\_task\_2.png")  print("Linear regressor performance:")  print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  print(      "Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2)  )  print(      "Explain variance score =",      round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2),  )  print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  # Файл для збереження моделі  output\_model\_file = "model.pkl"  # Збереження моделі  with open(output\_model\_file, "wb") as f:      pickle.dump(regressor, f)  # Завантаження моделі  y\_test\_pred\_new = regressor.predict(X\_test)  print(      "\nNew mean absolute error =",      round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2),  ) |
| Linear regressor performance:  Mean absolute error = 3.59  Mean squared error = 17.39  Median absolute error = 3.39  Explain variance score = 0.02  R2 score = -0.16  New mean absolute error = 3.59 |



Регресія на одній змінній показала нижчу точність (R² = -0.16). Модель не змогла ефективно передбачити дані.

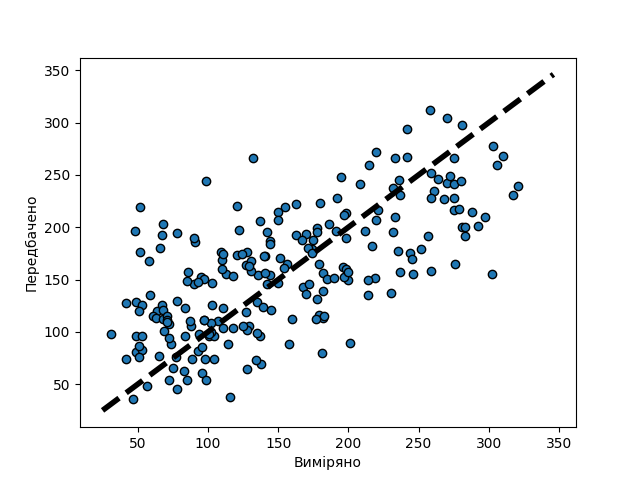
Завдання 3. Створення багатовимірного регресора

|  |
| --- |
| import numpy as np  import sklearn.metrics as sm  from sklearn import linear\_model  from sklearn.preprocessing import PolynomialFeatures  # Вхідний файл, який містить дані  input\_file = "data\_multivar\_regr.txt"  # Завантаження даних  data = np.loadtxt(input\_file, delimiter=",")  X, y = data[:, :-1], data[:, -1]  # Розбивка даних на навчальний та тестовий набори  num\_training = int(0.8 \* len(X))  num\_test = len(X) - num\_training  # Тренувальні дані  X\_train, y\_train = X[:num\_training], y[:num\_training]  # Тестові дані  X\_test, y\_test = X[num\_training:], y[num\_training:]  # Створення об'єкта лінійного регресора  regressor = linear\_model.LinearRegression()  regressor.fit(X\_train, y\_train)  # Прогнозування результату  y\_test\_pred = regressor.predict(X\_test)  print("Linear regressor performance:")  print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  print(      "Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2)  )  print(      "Explain variance score =",      round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2),  )  print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  # Завантаження моделі  y\_test\_pred\_new = regressor.predict(X\_test)  print(      "\nNew mean absolute error =",      round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2),  )  # Поліноміальна регресія  polynomial = PolynomialFeatures(degree=10)  X\_train\_transformed = polynomial.fit\_transform(X\_train)  datapoint = [[7.75, 6.35, 5.56]]  poly\_datapoint = polynomial.fit\_transform(datapoint)  poly\_linear\_model = linear\_model.LinearRegression()  poly\_linear\_model.fit(X\_train\_transformed, y\_train)  print("\nLinear regression:\n", regressor.predict(datapoint))  print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint)) |
| Linear regressor performance:  Mean absolute error = 3.58  Mean squared error = 20.31  Median absolute error = 2.99  Explain variance score = 0.86  R2 score = 0.86  New mean absolute error = 3.58  Linear regression:  [36.05286276]  Polynomial regression:  [41.45675521] |

Багатовимірний регресор. Лінійна регресія дала хороші результати (R² = 0.86), а поліноміальна регресія ще краще підійшла до даних.

Завдання 4. Регресія багатьох змінних

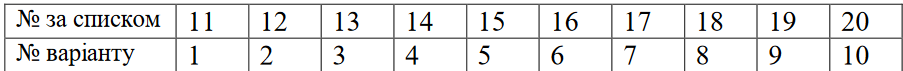
|  |
| --- |
| import matplotlib.pyplot as plt  from sklearn import datasets, linear\_model  from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score  from sklearn.model\_selection import train\_test\_split  diabetes = datasets.load\_diabetes()  X = diabetes.data  y = diabetes.target  Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.5, random\_state=0)  regr = linear\_model.LinearRegression()  regr.fit(Xtrain, ytrain)  ypred = regr.predict(Xtest)  print("Estimated coefficients for the linear regression ", regr.coef\_)  print(str(regr.intercept\_))  print("R2: ", r2\_score(ytrain, ypred))  print("Mean absolute error regression loss: ", mean\_absolute\_error(ytrain, ypred))  print("Mean squared error regression loss: ", mean\_squared\_error(ytrain, ypred))  fig, ax = plt.subplots()  ax.scatter(ytest, ypred, edgecolors=(0, 0, 0))  ax.plot([y.min(), y.max()], [y.min(), y.max()], "k--", lw=4)  ax.set\_xlabel("Виміряно")  ax.set\_ylabel("Передбачено")  plt.savefig("LR\_4\_task\_4.png") |
| Estimated coefficients for the linear regression [ -20.4047621 -265.88518066 564.65086437 325.56226865 -692.16120333  395.55720874 23.49659361 116.36402337 843.94613929 12.71856131]  154.3589285280134  R2: -0.49114093907045775  Mean absolute error regression loss: 81.37295843615576  Mean squared error regression loss: 9521.146930399502 |



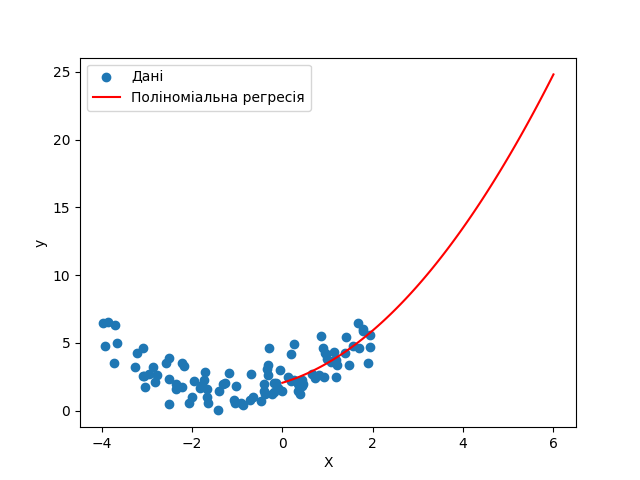
Багатозмінна регресія на основі набору даних про діабет. Модель мала низьку точність (R² = -0.49), що свідчить про проблеми з передбаченням.

Завдання 5. Самостійна побудова регресії

Мій варіант 13



|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.preprocessing import PolynomialFeatures  m = 100  X = 6 \* np.random.rand(m, 1) - 4  y = 0.5 \* X\*\*2 + X + 2 + np.random.randn(m, 1)  # Побудова поліноміальних ознак (квадратичних) для X  poly\_features = PolynomialFeatures(degree=2, include\_bias=False)  X\_poly = poly\_features.fit\_transform(X.reshape(-1, 1))  # Виведення значень ознак X[0] та X\_poly на екран  print("Значення ознак X[0]:", X[0])  print("Значення ознак після перетворення:", X\_poly[0])  # Підгонка лінійної моделі до розширених даних  lin\_reg = LinearRegression()  lin\_reg.fit(X\_poly, y)  # Виведення значень коефіцієнтів полінома  intercept = lin\_reg.intercept\_  coefficients = lin\_reg.coef\_  print("Значення intercept:", intercept)  print("Значення коефіцієнтів:", coefficients)  # Генерація значень для побудови графіку  X\_plot = np.linspace(0, 6, 100)  y\_plot = lin\_reg.predict(poly\_features.transform(X\_plot.reshape(-1, 1)))  # Побудова графіку  plt.scatter(X, y, label="Дані")  plt.plot(X\_plot, y\_plot, label="Поліноміальна регресія", color="red")  plt.xlabel("X")  plt.ylabel("у")  plt.legend()  plt.savefig("LR\_4\_task\_5.png") |
| Значення ознак X[0]: [-1.32432308]  Значення ознак після перетворення: [-1.32432308 1.75383163]  Значення intercept: [2.0579947]  Значення коефіцієнтів: [[0.99901533 0.4657998 ]] |

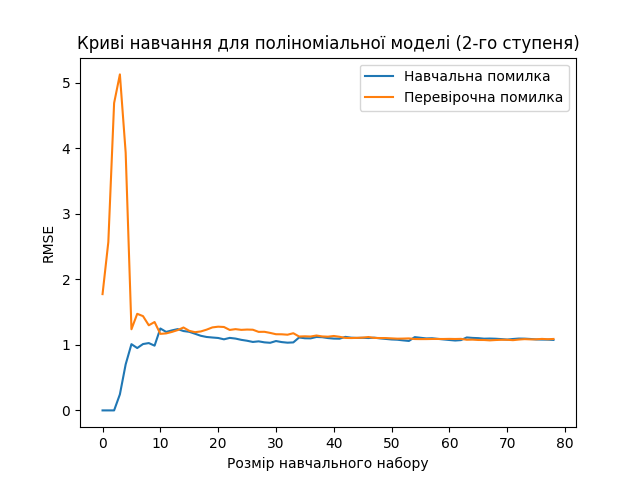


Рівняння отриманої поліноміальної моделі регресії має вигляд:

y=0.99901533 ⋅X2 +0.4657998⋅X+2.0579947

Завдання 6. Побудова кривих навчання

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import PolynomialFeatures  m = 100  X = 6 \* np.random.rand(m, 1) - 4  y = 0.5 \* X\*\*2 + X + 2 + np.random.randn(m, 1)  # Розділимо дані на навчальний та перевірочний набори  X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # Функція для побудови кривих навчання  def plot\_learning\_curves(model, X, y, X\_val, y\_val):      train\_errors, val\_errors = [], []      for m in range(1, len(X)):          model.fit(X[:m], y[:m])          y\_train\_predict = model.predict(X[:m])          y\_val\_predict = model.predict(X\_val)          train\_errors.append(mean\_squared\_error(y[:m], y\_train\_predict))          val\_errors.append(mean\_squared\_error(y\_val, y\_val\_predict))      plt.plot(np.sqrt(train\_errors), label="Навчальна помилка")      plt.plot(np.sqrt(val\_errors), label="Перевірочна помилка")      plt.legend()  poly\_features = PolynomialFeatures(degree=2, include\_bias=False)  X\_poly = poly\_features.fit\_transform(X\_train)  X\_poly\_val = poly\_features.transform(X\_val)  lin\_reg = LinearRegression()  plot\_learning\_curves(lin\_reg, X\_poly, y\_train, X\_poly\_val, y\_val)  plt.xlabel("Розмір навчального набору")  plt.ylabel("RMSE")  plt.title("Криві навчання для поліноміальної моделі (2-го ступеня)")  plt.savefig("LR\_4\_tasK\_6.png") |



Побудова кривих навчання для поліноміальної регресії. Криві показали, як змінюються навчальні і перевірочні помилки залежно від кількості даних, і допомогли проаналізувати якість моделі.

**Висновок:**

У цій лабораторній роботі було досліджено різні методи регресії для аналізу даних. Використання лінійної регресії на простих наборах даних показало високу точність моделі, однак із збільшенням кількості змінних точність прогнозування значно знижувалася. Поліноміальна регресія виявилася ефективнішою у випадках з нелінійними даними. Завдяки побудові кривих навчання було проаналізовано поведінку моделей залежно від кількості даних, що дало змогу оцінити ефективність навчання.