

## ЛАБОРАТОРНА РОБОТА № 3

### ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

#### 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Теоретичні відомості подані на лекціях. Також доцільно вивчити матеріал поданий в літературі:

Джоши Пратик. Искусственный интеллект с примерами на Python. : Пер. с англ. - СПб. : ООО "Диалектика", 2019. - 448 с. - Парал. тит. англ. ISBN 978-5-907114-41-8 (рус.)

Можна використовувати Google Colab або Jupiter Notebook.

Регресія - це процес оцінки того, як співвідносяться між собою вхідні та вихідні змінні. Слід зазначити, що вихідні змінні можуть мати значення з безперервного ряду дійсних чисел. Отже, існує безліч результуючих можливостей. Це різко контрастує з процесом класифікації, у якому кількість вихідних класів фіксовано.

У регресії передбачається, що вихідні змінні залежить від вхідних, і завдання полягає у з'ясуванні співвідношення між ними. Звідси вхідні змінні називаються незалежними змінними (або предикторами), а вихідні – залежними (або критеріальними змінними). При цьому не потрібно, щоб вхідні змінні були незалежними один від одного. Існує безліч ситуацій, коли між вхідними змінними існує кореляція.

Регресійний аналіз дозволяє з'ясувати, як змінюється значення вихідний змінної, коли змінюємо лише частина вхідних змінних, залишаючи інші вхідні змінні фіксованими. У разі лінійної регресії передбачається, що вхідні та вихідні змінні пов'язані між собою лінійною залежністю. Це накладає обмеження на нашу процедуру моделювання, але прискорює її та робить більш ефективною.

Іноді лінійної регресії виявляється недостатньо для пояснення співвідношень між вхідними та вихідними змінними. У подібних випадках ми використовуємо поліноміальну регресію, в якій вхідні та вихідні змінні пов'язані між собою поліноміальною залежністю.

З обчислювальної точки зору такий підхід складніший, але забезпечує більш високу точність. Вибір виду регресії для виявлення зазначених відношень визначається видом конкретної задачі. Регресію часто використовують для прогнозування цін, економічних показників та інше.

## 2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

### Завдання 2.1. Створення регресора однієї змінної

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

#### *РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ*

Створіть новий файл Python та імпортуйте такі пакети.

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
```

Завантажуємо вхідні дані.

```
# Вхідний файл, який містить дані
input_file = 'data_singlevar_regr.txt'
```

У цьому текстовому файлі використовується кома, тому для завантаження даних можна скористатися наступним викликом функції.

```
# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
```

Розіб'ємо дані на навчальний та тестовий набори.

```
# 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:]
```

Створимо об'єкт лінійного регресора та навчимо його, використовуючи тренувальні дані.

```
# Створення об'єкта лінійного регресора
regressor = linear_model.LinearRegression()
```

```
regressor.fit(X_train, y_train)
```

Спрогнозуємо результат для тестового набору даних, використовуючи модель, що навчається.

```
# Прогнозування результату
y_test_pred = regressor.predict(X_test)
```

Побудуємо вихідний графік.

```
# 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.show()
```

Обчислимо метричні параметри регресора, порівнюючи справжні значення з передбаченими.

```
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))
```

Створивши модель, ми можемо зберегти її у файлі для подальшого використання. Python надає відмінний модуль, який дозволяє легко це зробити.

```
# Файл для збереження моделі
output_model_file = 'model.pkl'
```

```
# Збереження моделі
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
```

Завантажимо модель з файлу на диску та побудуємо прогноз.

```
# Завантаження моделі
y_test_pred_new = regressor_model.predict(X_test)
print("\nNew mean absolute error =",
round(sm.mean_absolute_error(y_test, y_test_pred_new), 2))
```

**Збережіть код робочої програми під назвою LR\_3\_task\_1.py**  
**Код програми, графік функції та результати оцінки якості занесіть у звіт.**  
**Зробіть висновок**

## Завдання 2.2. Передбачення за допомогою регресії однієї змінної

Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі (таблиця 2.1).

Таблиця 2.1

№ за списком	1	2	3	4	5	6	7	8	9	10
№ варіанту	1	2	3	4	5	1	2	3	4	5

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	1	2	3	4	5

№ за списком	21	22	23	24	25	26	27	28	29	30
№ варіанту	1	2	3	4	5	1	2	3	4	5

Варіант 1 файл: data\_regr\_1.txt  
 Варіант 2 файл: data\_regr\_2.txt  
 Варіант 3 файл: data\_regr\_3.txt  
 Варіант 4 файл: data\_regr\_4.txt  
 Варіант 5 файл: data\_regr\_5.txt

## РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Зробити по аналогії з пунктом 2.1.

**Збережіть код робочої програми під назвою LR\_3\_task\_2.py**  
**Код програми, графік функції та результати оцінки якості занесіть у звіт.**  
**Зробіть висновок**

## Завдання 2.3. Створення багатовимірного регресора

Використовувати файл вхідних даних: data\_multivar\_regr.txt, побудувати регресійну модель на основі багатьох змінних.

## РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Створіть новий файл Python та імпортуйте такі пакети.

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
```

Відкрийте файл, який містить дані: data\_multivar\_regr.txt.

У цьому текстовому файлі в якості роздільника використовується кома.

Завантажте дані .

Розбийте дані на навчальний та тестовий набори (як в завданні 2.1).

Створіть та навчіть модель **лінійного регресора** (як в завданні 2.1).

Спрогнозуйте результат для тестового набору даних.

Виведіть на екран метрики якості лінійної регресії.

Linear Regressor performance: Mean absolute error, Mean squared error, Median absolute error, Explained variance score, R2 score.

Створіть **поліноміальний регресор** ступеня 10 та навчіть його на тренувальних даних.

```
# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
```

Візьміть деяку вибірку точку даних і спрогнозуйте для неї результат. Перший крок полягає в тому, щоб перетворити її на поліном.

```
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.fit_transform(datapoint)
```

Неважко зазначити, що ця точка дуже близька до точки даних [7.66, 6.29, 5.66], зазначеної в рядку 11 нашого файлу даних. Тому вдало створений регресор повинен передбачити результат, близький до 41.35. Створіть об'єкт лінійного регресора і виконайте підгонку до полінома. Побудуйте прогноз з використанням як лінійного, так і поліноміального регресора, щоб побачити різницю.

```
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)

print("\nLinear regression:\n",
      linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n",
      poly_linear_model.predict(poly_datapoint))
```

Оцініть та порівняйте отримані характеристики.

Зверніть увагу, що порівняно з лінійним регресором поліноміальний регресор забезпечує отримання результату, ближчого до значення 41.35. Тобто дає кращі результати.

**Збережіть код робочої програми під назвою *LR\_3\_task\_3.py*. Код програми та результати оцінки якості занесіть у звіт. Зробіть висновок**

## Завдання 2.4. Регресія багатьох змінних

Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в `sklearn.datasets`.

Набір даних містить 10 вихідних змінних — вік, стать, індекс маси тіла, середній артеріальний тиск і шість вимірювань сироватки крові, отриманих у 442 пацієнтів із цукровим діабетом, а також реакцію, що цікавить, — кількісний показник прогресування захворювання через 1 рік після вихідного рівня. Отже, існує 442 екземпляри з 10 атрибутами. Колонка 11 є кількісною мірою прогресування захворювання через 1 рік після вихідного рівня. Кожен з цих 10 атрибутів був відцентрований по середньому та масштабований за часом стандартного відхилення `n_samples` (тобто сума квадратів кожного стовпця складає 1). Оригінальні дані можна завантажити з: <https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html>.

Використайте всі функції набору даних про діабет, щоб побудувати двовимірний графік лінійної регресії. Побудуйте графік залежності між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням (крапками) та пряму лінію, по цьому графіку, що покаже, як лінійна регресія намагається провести пряму лінію, яка мінімізує залишкову суму квадратів між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням. Також розрахуйте коефіцієнт кореляції  $R^2$ , середню абсолютну помилку (MAE) і середньоквадратичну помилку (MSE).

## РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Вам знадобляться:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
```

Завантажте дані

```
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
```

Поділіть їх на навчальну та тестову вибірки з параметрами `test_size = 0.5`, `random_state = 0`

```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size
= 0.5, random_state = 0)
```

Створіть модель лінійної регресії та натренуйте її.

```
regr = linear_model.LinearRegression()
```

```
regr.fit(Xtrain, ytrain)
```

Зробіть прогноз по тестовій вибірці

```
ypred = regr.predict(Xtest)
```

Розрахуйте, підпишіть та виведіть на екран коефіцієнти регресії та показники

```
regr.coef_
regr.intercept_
r2_score
mean_absolute_error
mean_squared_error
```

Побудуйте графіки

```
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.show()
```

**Збережіть код робочої програми під назвою `LR_3_task_4.py`**

**Код програми, графіки та результати оцінки якості занесіть у звіт.**

**Зробіть висновок**

## Завдання 2.5. Самостійна побудова регресії

Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

Таблиця 2.2

№ за списком	1	2	3	4	5	6	7	8	9	10
№ варіанту	1	2	3	4	5	6	7	8	9	10

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	6	7	8	9	10

№ за списком	21	22	23	24	25	26	27	28	29	30
№ варіанту	1	2	3	4	5	6	7	8	9	10

### Варіант 1

```
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)
```

### Варіант 2

```
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.6 * X ** 2 + X + 2 + np.random.randn(m, 1)
```

### Варіант 3

```
m = 100
X = 6 * np.random.rand(m, 1) - 4
y = 0.5 * X ** 2 + X + 2 + np.random.randn(m, 1)
```

### Варіант 4

```
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)
```

### Варіант 5

```
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)
```

### Варіант 6

```
m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, m)
```

### Варіант 7

```
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

### Варіант 8



```
m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

### Варіант 9

```
m = 100
X = np.linspace(-3, 3, m)
y = 3 + np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

### Варіант 10

```
m = 100
X = np.linspace(-3, 3, m)
y = 4 + np.sin(X) + np.random.uniform(-0.6, 0.6, m)
```

## РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

Побудуйте вашу модель, та виведіть ваші дані на графік залежності  $y=f(X)$ .

Що, якщо ваші дані насправді складніші за звичайну пряму лінію? Дивно, але ви насправді можете застосовувати лінійну модель для припасування до нелінійних даних. Простий спосіб передбачає додавання ступенів кожної ознаки у вигляді нових ознак і наступне навчання лінійної моделі на такому розширеному наборі ознак. Цей прийом називається *поліноміальною регресією (polynomial regression)*.

Якщо вхідні данні розподілені нелінійно, то, безумовно, пряму лінію ніколи не буде підігнано під такі дані належним чином. Тому скористайтеся класом `PolynomialFeatures` з `Scikit-Learn`, щоб перетворити наші навчальні дані, додавши як нові ознаки квадрат (поліном 2-го ступеня) кожної ознаки (у нашому випадку є тільки одна ознака):

```
PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
```

Виведіть значення коефіцієнтів полінома `X[0]` та `X_poly` на екран.

Тепер `X_poly` містить первинну ознаку `X` плюс її квадрат.

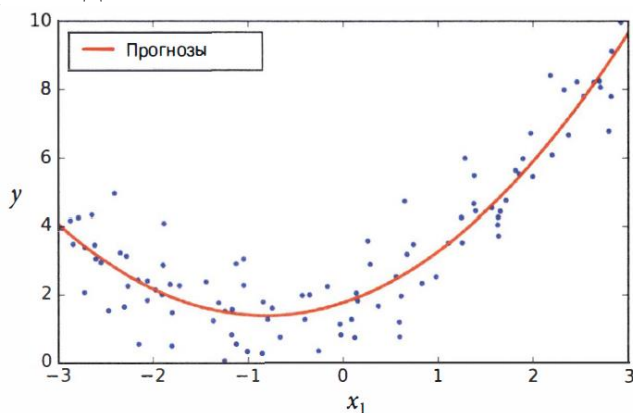
Далі підгоніть модель `LinearRegression` до таких розширених навчальних даних.

```
lin_reg = LinearRegression( )
lin_reg.fit(X_poly, y)
lin_reg.intercept, lin_reg.coef
```

Зверніть увагу, що за наявності множини ознак поліноміальна регресія здатна знайти зв'язок між ознаками (те, що проста лінійна регресійна модель робити неспроможна). Це стає можливим завдяки тому факту, що клас `PolynomialFeatures` також додає всі комбінації ознак до заданого ступеня.

Наприклад, якщо є дві ознаки  $a$  і  $b$ , тоді `PolynomialFeatures` з `degree=3` додав би як ознаки  $a^2$ ,  $a^3$ ,  $b^2$  і  $b^3$ , а й комбінації  $ab$ ,  $a^2b$  і  $ab^2$ .

Виведіть графік вашої моделі крапками, а регресію лінією. Наприклад:



**Ваш графік занесіть у звіт.**

**Запишіть модель вашого варіанта у вигляді математичного рівняння (наприклад  $y = 0.5x_1^2 + 1.0x_1 + 2.0 + \text{гауссов шум}$ ) та запишіть отриману вами модель регресії з передбаченими коефіцієнтами (наприклад  $y = 0.56x_1^2 + 0.93x_1 + 1.78$ ).**

Отримані вами коефіцієнти повинні бути близьким до модельних. І це буде означати що модель навчена правильно.

**Збережіть код робочої програми під назвою `LR_3_task_5.py`  
Код програми та результати регресії занесіть у звіт.  
Зробіть висновок**

## Завдання 2.6. Побудова кривих навчання

Побудуйте криві навчання для ваших даних у попередньому завданні.

### РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ

У разі виконання поліноміальної регресії високого ступеня, ймовірно, ви краще підганяєте навчальні дані, ніж за допомогою лінійної регресії. Наприклад, на рис. 4 демонструється застосування поліноміальної моделі 300-го ступеня до попередніх навчальних даних, а результат порівнюється з чистою лінійною моделлю та квадратичною моделлю (поліноміальною 2-го ступеня).

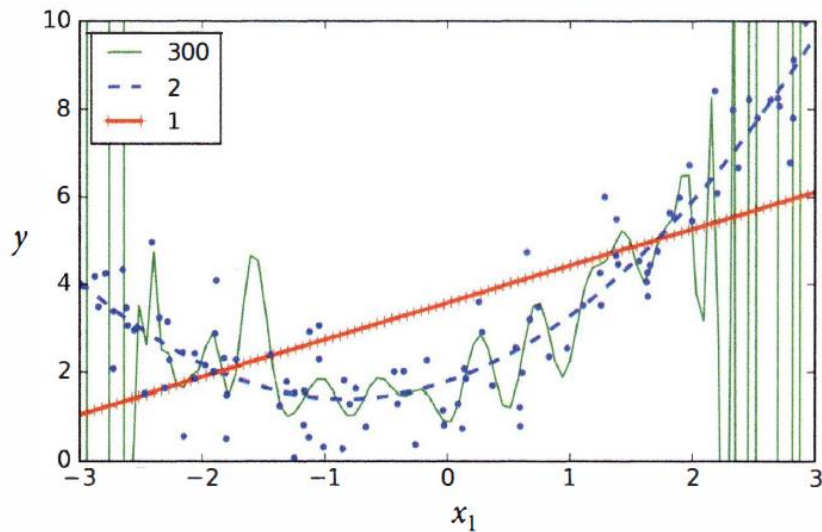


Рис.4

Зверніть увагу, як поліноміальна модель 300-го ступеня коливається, щоб якомога більше наблизитися до навчальних зразків. Зрозуміло, така поліноміальна регресійна модель високого ступеня викликає сильне перенавчання навчальними даними, тоді як лінійна модель - недонавчання ними. У даному випадку добре узагальнюватиметься квадратична модель. Це має сенс, оскільки дані генерувалися з використанням квадратного рівняння, але з урахуванням того, що зазвичай ви *не знатимете функцію*, що використовується для генерації даних, як приймати рішення про те, наскільки складною має бути модель? Як з'ясувати, що модель перенавчається або недонавчається на даних?

У попередніх лабораторних роботах за допомогою перехресної перевірки оцінювалася продуктивність узагальнення моделі. Якщо згідно з метриками перехресної перевірки модель добре виконується на навчальних даних, але погано узагальнюється, то модель перенавчена. Якщо модель погано виконується в обох випадках, тоді вона недонавчена. Так виглядає один із способів сказати, що модель надто проста чи надмірно складна.

Інший спосіб передбачає перегляд кривих навчання (learning curve): вони є графіки продуктивності моделі на навчальному наборі і перевірочному наборі як функції від розміру навчального набору (або ітерації навчання). Щоб отримати такі графіки, потрібно просто навчити модель кілька разів на підмножині різних розмірів, взятих з навчального набору.

Визначте функцію, яка будує криві навчання моделі для встановлених навчальних даних:

```

from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val =
        train_test_split(X, y, test_size=0.2)
    train_errors, val_errors = [], []
    for m in range(1, len(X_train)):
        model.fit(X_train[:m], y_train[:m])
        y_train_predict = model.predict(X_train[:m])
        y_val_predict = model.predict(X_val)
        train_errors.append(mean_squared_error(y_train_predict,
                                                y_train[:m]))

        val_errors.append(mean_squared_error(y_val_predict, y_val))
    plt.plot(np.sqrt(train_errors), "r--", linewidth=2, label="train")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")

```

Побудуйте криві навчання для звичайної лінійної регресійної моделі із попереднього завдання (ваш варіант):

```

lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)

```

Ви повинні отримати щось подібне

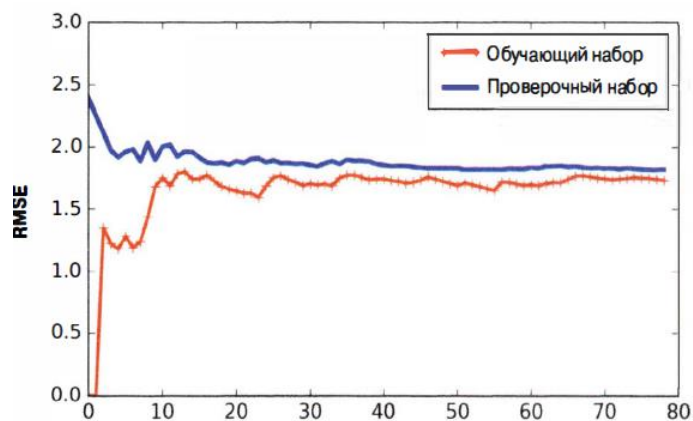


Рис. 5. Криві навчання для лінійної моделі

**Ваш графік занесіть у звіт.**

Тут потрібні деякі пояснення. Насамперед зверніть увагу на продуктивність моделі у разі використання навчальних даних: коли в навчальному наборі є тільки один або два зразки, модель може бути повною мірою підігнана до них, що пояснює початок кривої з нульової помилки. Але в міру додавання зразків у навчальний набір ідеальна підгонка моделі до навчальних даних стає неможливою, як через те, що дані зашумлені, так і тому, що вони зовсім відрізняються від лінійних. Отже, помилка на навчальних даних рухається вгору, поки не стабілізується, коли додавання

нових зразків у навчальний набір не робить середню помилку набагато краще чи гірше. Тепер перейдемо до продуктивності моделі на перевірочних даних. Коли модель навчалася на незначній кількості зразків, вона нездатна узагальнюватися належним чином, а тому помилка перевірки спочатку досить велика. Потім у міру того, як модель бачить все більше навчальних зразків вона навчається, а помилка перевірки відповідно повільно знижується. Однак пряма лінія знову не в змозі добре змоделювати дані, тому помилка стабілізується поблизу іншої кривої.

Такі криві навчання типові для недонавченої моделі. Обидві криві стабілізуються; вони розташовані близько одна до одної і знаходяться досить високо.

Якщо ваша модель недонавчена на навчальних даних, тоді додавання додаткових навчальних зразків не допоможе. Вам потрібно вибрати складнішу модель або знайти найкращі ознаки.

Тепер побудуйте криві навчання поліноміальної моделі 10-го ступеня на тих самих ваших даних:

```
from sklearn.pipeline import Pipeline
polynomial_regression = Pipeline([
    ("poly_features",
     PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)
```

Ви повинні отримати щось подібне до рис. 6.

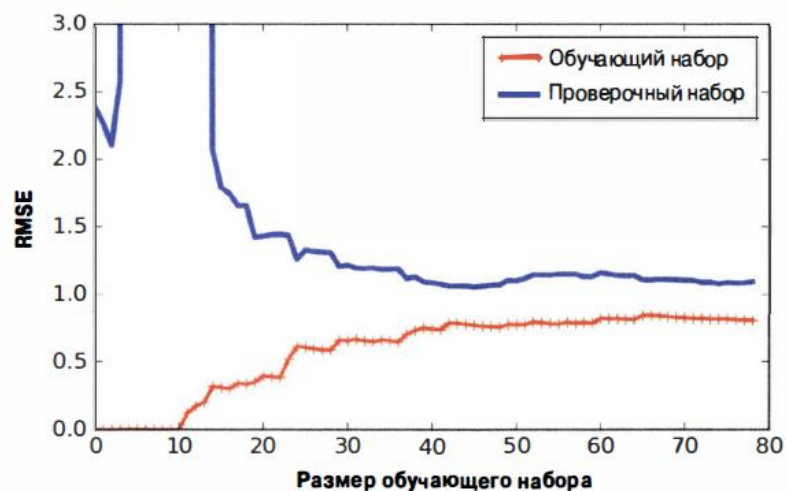


Рис.6. Криві навчання для поліноміальної моделі

**Ваш графік занесіть у звіт.**

Криві навчання виглядають трохи краще за попередні, але є дві дуже важливі відмінності.

- Помилка на навчальних даних набагато нижча, ніж у випадку лінійної регресійної моделі.

- Між кривими є проміжок. Це означає, що модель виконується значно краще на навчальних даних, ніж на перевірочних даних, демонструючи ознаку перенавчання. Тим не менш, якщо ви застосуєте набагато більший навчальний набір, дві криві продовжать зближення.

Один із способів поліпшення перенавченої моделі полягає у наданні їй додаткових навчальних даних доти, доки помилка перевірки не досягне помилки навчання.

#### *Компроміс між зміщенням та дисперсією*

Важливим теоретичним результатом статистики та машинного навчання є той факт, що помилка узагальнення моделі може бути виражена у вигляді суми трьох різних помилок.

**Зміщення.** Ця частина помилки узагальнення пов'язана з невірними припущеннями, такими як припущення того, що дані є лінійними, коли вони насправді квадратичні. Модель з високим зсувом, швидше за все, недонавчиться на навчальних даних.

**Дисперсія.** Ця частина пояснюється надмірною чутливістю моделі до невеликих змін у навчальних даних. Модель з багатьма ступенями свободи (така як поліноміальна модель високого ступеня), ймовірно, матиме високу дисперсію і тому перевчитися навчальними даними.

**Непереборна похибка.** Ця частина з'являється внаслідок шуму самих даних. Єдиний спосіб скоротити непереборну похибку в помилці передбачає очищення даних (наприклад, упорядкування джерел даних, таких як несправні датчики, або виявлення та усунення викидів).

Зростання складності моделі зазвичай збільшує її дисперсію та зменшує зміщення. І навпаки, скорочення складності моделі збільшує її зміщення та зменшує дисперсію. Ось чому це називається компромісом.

Тепер побудуйте криві навчання поліноміальної моделі 2-го ступеня на тих самих ваших даних.

***Ваш отриманий графік занесіть у звіт.***

***Код програми та результати занесіть у звіт.***

***Програмний код збережіть під назвою LR\_3\_task\_6.py***

***Коди комітити на GitHub. У кожному звіті повинно бути посилання на GitHub.***

***Назвіть бланк звіту СШ-ЛР-3-NNN-XXXXX.doc***

*де NNN – позначення групи*

*XXXXX – позначення прізвища студента.*

*Переконвертуйте файл звіту в СШІ-ЛР-3-NNN-XXXXX.pdf*