ЛАБОРАТОРНА РОБОТА № 3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

Завдання 2.1. Створення регресора однієї змінної

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

# 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.show()
```

					Державний університет "Житомирська політехніка"	
Змн.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Дроботун Д. Я.			Звіт з практичної роботи	Літ. / Арк. 1 / Аркушів
Перевір.						
Керівник						
Н. контр.						Гр. ІПЗК-19-1
Зав. каф.						

```
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))

output_model_file = 'model.pkl'

with open(output_model_file, 'wb') as f:
   pickle.dump(regressor, f)

with open(output_model_file, 'rb') as f:
   regressor_model = pickle.load(f)

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))
```

Рис. 1.1 Код програми

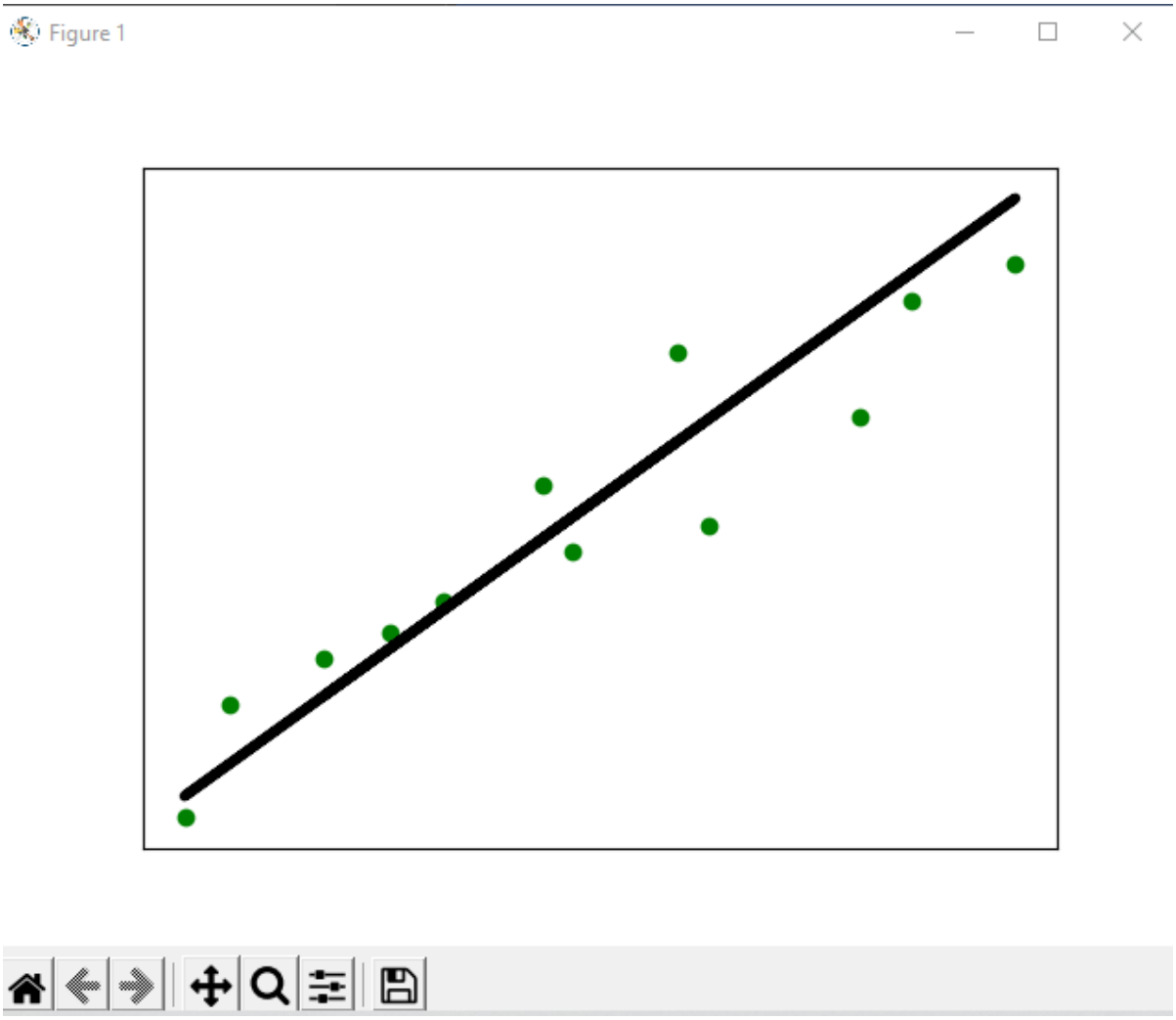

Рис. 1.2 Результат виконання програми

```
E:\ЖДУ\Python\Work_Drobotun_3\venv\Scripts\python.exe
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

Process finished with exit code 0
```

Рис. 1.3 Результат виконання програми

Завдання 2.2. Передбачення за допомогою регресії однієї змінної

За списком у журналі я **5** варіант

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

# Вхідний файл, який містить дані
input_file = 'data_regr_5.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.show()

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))

output_model_file = 'model.pkl'

with open(output_model_file, 'wb') as f:
   pickle.dump(regressor, f)

with open(output_model_file, 'rb') as f:
   regressor_model = pickle.load(f)

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))
```

Рис. 1.4 Код програми

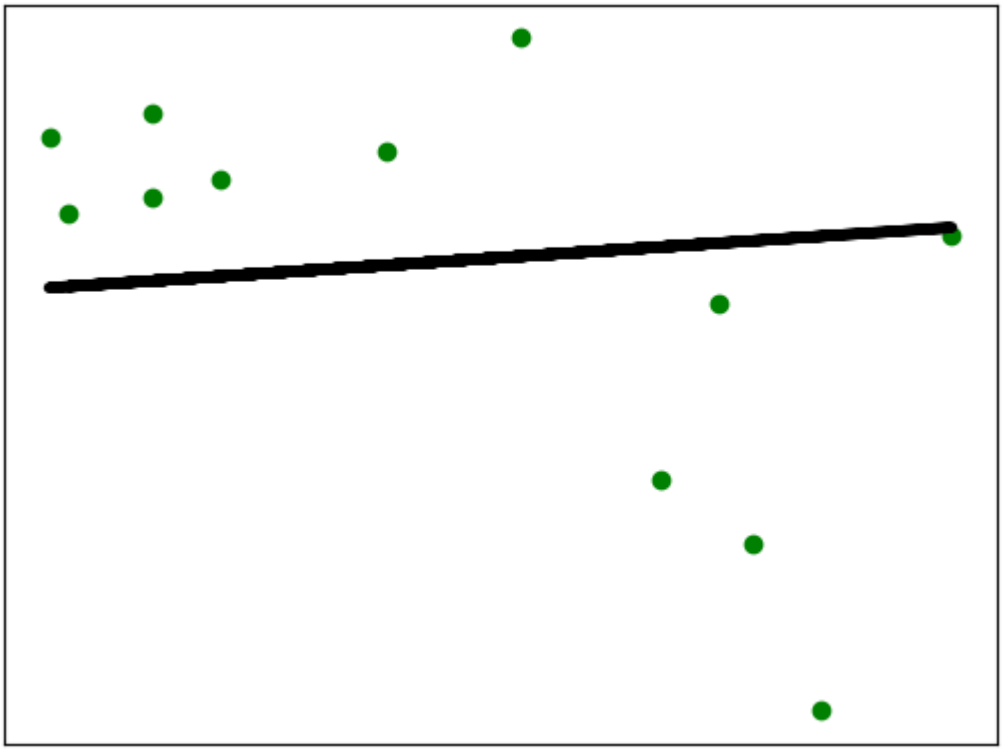

Рис. 1.5 Результат виконання програми

```
E:\ЖДУ\Python\Work_Drobotun_3\venv\Scripts\pyt
Linear regressor performance:
Mean absolute error = 3.31
Mean squared error = 16.98
Median absolute error = 2.66
Explain variance score = -0.14
R2 score = -0.15

New mean absolute error = 3.31
```

Рис. 1.6 Результат виконання програми

Завдання 2.3. Створення багатовимірного регресора

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

input_file = 'data_multivar_regr.txt'


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


num_training = int(0.8 * len(X))
num_test = len(X) - num_training


X_train, y_train = X[:num_training], y[:num_training]


X_test, y_test = X[num_training:], y[num_training:]


linear_regressor = linear_model.LinearRegression()


linear_regressor.fit(X_train, y_train)


y_test_pred = linear_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("Explained 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))
```

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

Рис. 1.7 Код програми

```
E:\ЖДУ\Python\Work_Drobotun_3\venv\Scrip
Linear Regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explained variance score = 0.86
R2 score = 0.86

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46581942]
```

Рис. 1.8 Результат виконання програми

Завдання 2.4. Регресія багатьох змінних

```
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

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)
```

```
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()
```

Рис. 1.9 Код програми

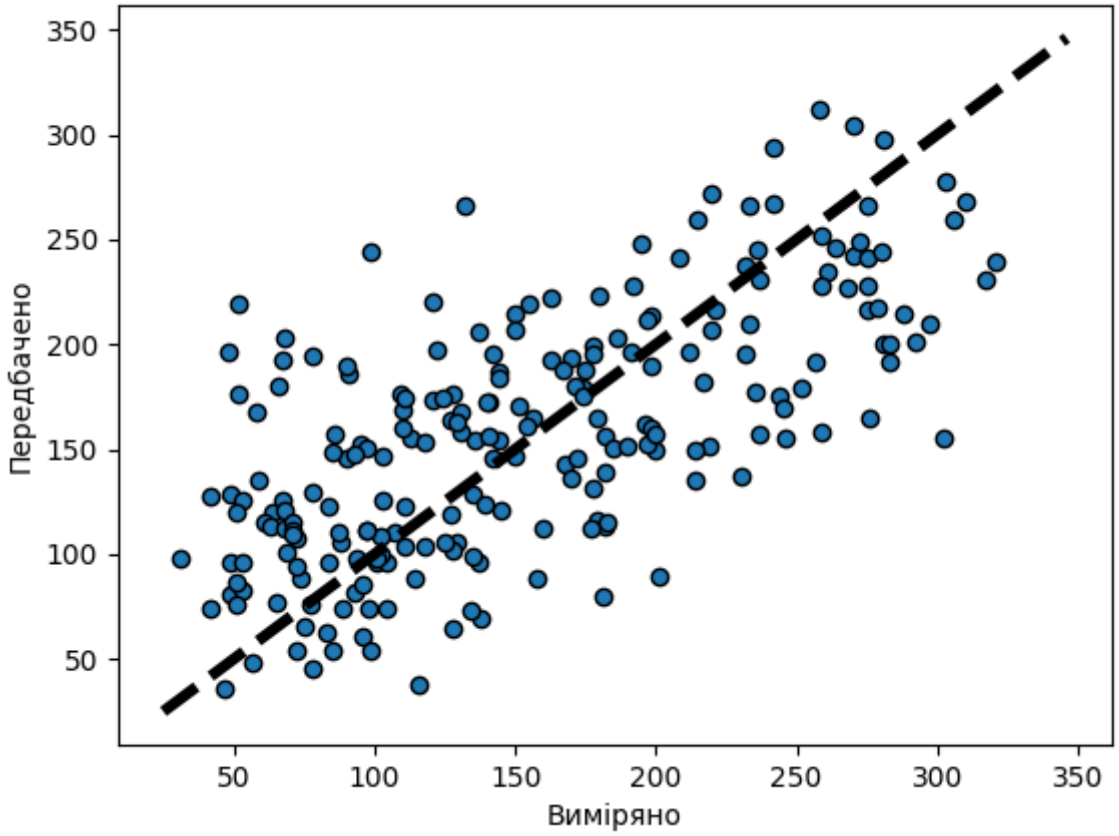

Рис. 2.1 Результат виконання програми

Завдання 2.5. Самостійна побудова регресії

За списком у журналі я **5** варіант

```
import numpy as np
from sklearn.linear_model import LinearRegression

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

reg = LinearRegression().fit(X, y)
print(reg.score(X, y))

print(reg.coef_)

print(reg.intercept_)
```

```
print(reg.predict)
```

Рис. 2.2 Код програми

```
E:\ЖДУ\Python\Work_Drobotun_3\venv\Scripts\python.exe E:/Ж
0.5873949421539033
[[1.00764301]]
[5.37564417]
<bound method LinearModel.predict of LinearRegression()>
```

Рис. 2.3 Результат виконання програми

Завдання 2.6. Побудова кривих навчання

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures

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

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")
   plt.show()
lin_reg=LinearRegression()
plot_learning_curves(lin_reg,X,y)
```

Рис. 2.4 Код програми

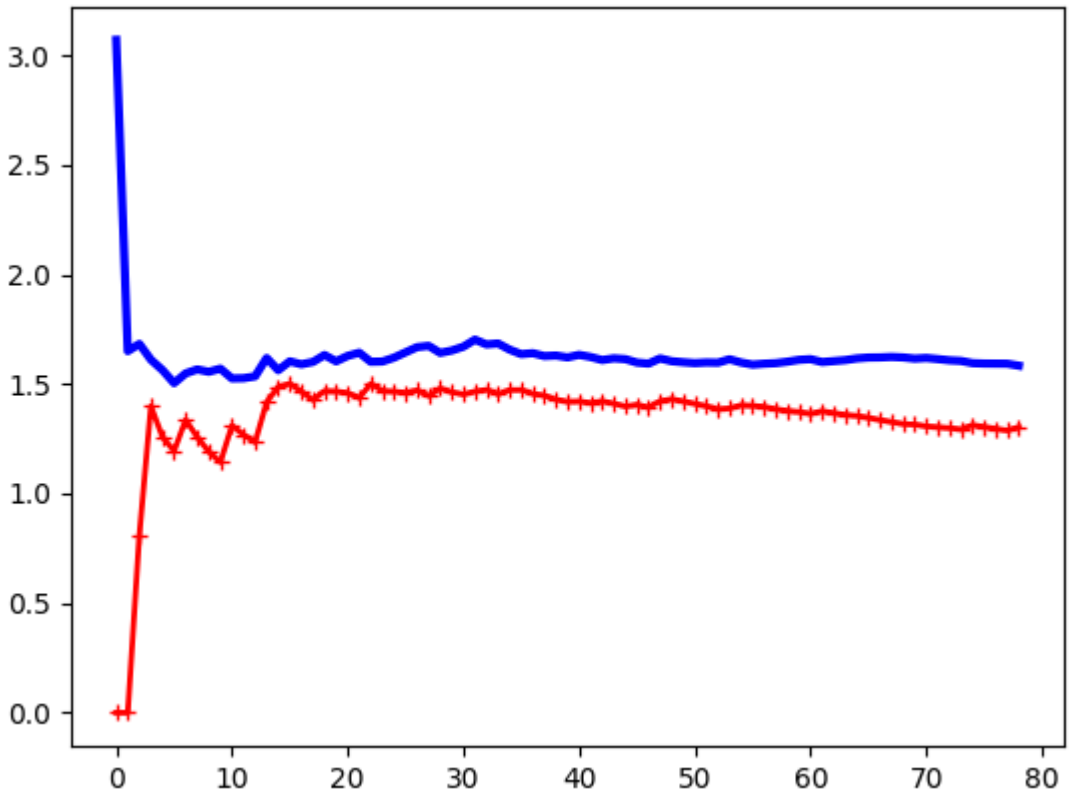

Рис. 2.5 Результат виконання програми

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures

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

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")
   plt.show()
lin_reg=LinearRegression()
plot_learning_curves(lin_reg,X,y)

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)
```

Рис. 2.6 Код програми

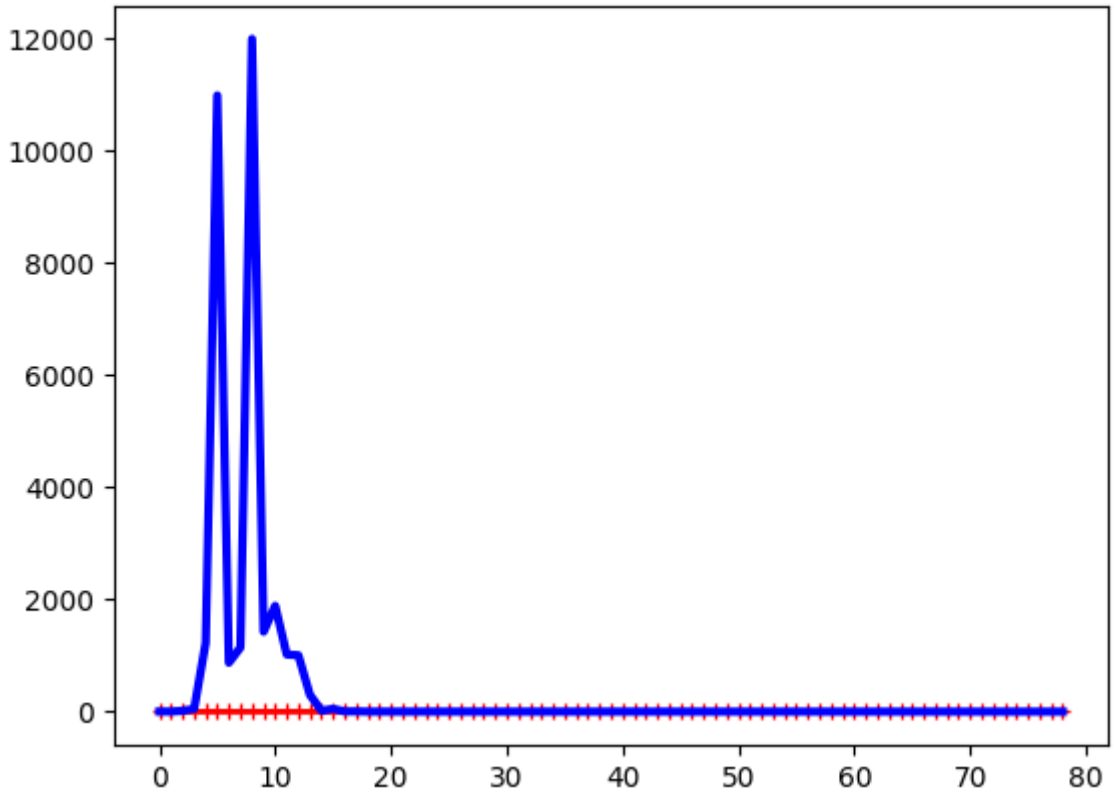

Рис. 2.7 Результат виконання програми

Висновок: я використовав спеціалізовані бібліотеки та мову програмування Python дослідивши методи регресії даних у машинному навчанні.

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 скориставшись класом PolynomialFeatures з Scikit-Learn, щоб перетворити наші навчальні дані, додавши як нові ознаки квадрат (поліном 2-го ступеня) кожної ознаки.