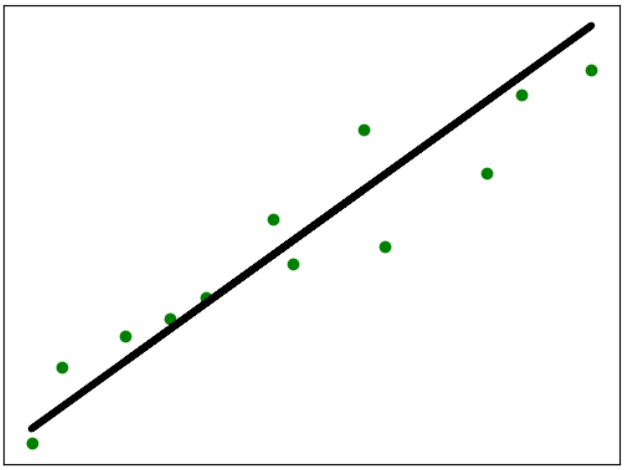
Системи штучного інтелекту.
Лабораторна робота 4.Федорович Дмитро ІПЗ-21-3

Завдання №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()
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
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
y_test_pred_new = regressor_model.predict(X_test)
print("New 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
```



Графік зображує незалежні змінні x та залежну змінну y. Лінійна модель регресії показує нормальні показники, проте є відхилення.

Завдання №2. Передбачення за допомогою регресії однієї змінної

№ за списком – 22

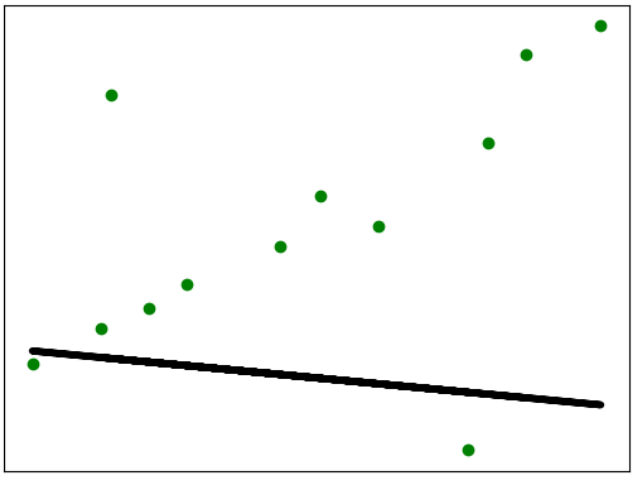
№ варіанту – 2

Варіант 2 файл: data_regr_2.txt

```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_2.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))
# Файл для збереження моделі
output_model_file = 'model_task2.pkl'
# Збереження моделі
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
# Завантаження моделі
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
y_test_pred_new = regressor_model.predict(X_test)
print("New mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

```
Linear regressor performance:
Mean absolute error = 2.42
Mean squared error = 9.02
Median absolute error = 2.14
Explain variance score = -0.15
R2 score = -1.61
New mean absolute error = 2.42
```



Лінійна регресія показує непогані показники - середнє абсолютне відхилення, тобто середня різниця між прогнозованими й фактичними даними є 2.42. А середня квадратична різниця - 9.09. Оцінка r2 -1.61 свідчить що модель не може пояснити відхилення між частиною показників.

Завдання №3. Створення багатовимірного регресора

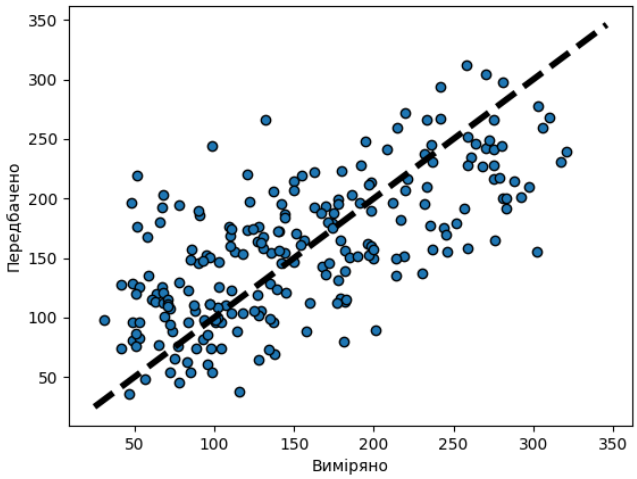
```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
# Вхідний файл, який містить дані
input_file = 'data_multivar_regr.txt'
# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -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:]
# 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)
# Створення об'єкта полілінійного регресора
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
print("\nLinear regression:\n",
linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n",
poly_linear_model.predict(poly_datapoint))
# Прогнозування результату
y_test_pred = linear_regressor.predict(X_test)
print("\nLinear 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))
```

```
Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46174702]

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
```

В даному прикладі поліноміальна регресія дає ближчий до реальності результат ніж лінійна регресія (41.46 - 41.35), що показує її вищу ефективність в складних залежностях.

Завдання №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)
print('Coefficients: \n', regr.coef_)
print('Intercept: \n', regr.intercept_)
print('Coefficient of determination: %.2f' % r2_score(ytest, ypred))
print('Mean absolute error: %.2f' % mean_absolute_error(ytest, ypred))
print('Mean squared error: %.2f' % mean_squared_error(ytest, 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.show()
```




```
Coefficients:
 [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Intercept:
 154.3589285280134
Coefficient of determination: 0.44
Mean absolute error: 44.80
Mean squared error: 3075.33
```

По графіку можна розуміти, що зі збільшенням фактичного рівня цукру в крові, збільшується і спрогнозований. Лінія показує місце з мінімальною різницею між спостережуваними й передбаченими показниками. Модель в 44% випадків може пояснити дисперсію між даними. Середня абсолютна похибка моделі 44.8, а середня квадратична похибка 3075.

Завдання №5. Самостійна побудова регресії

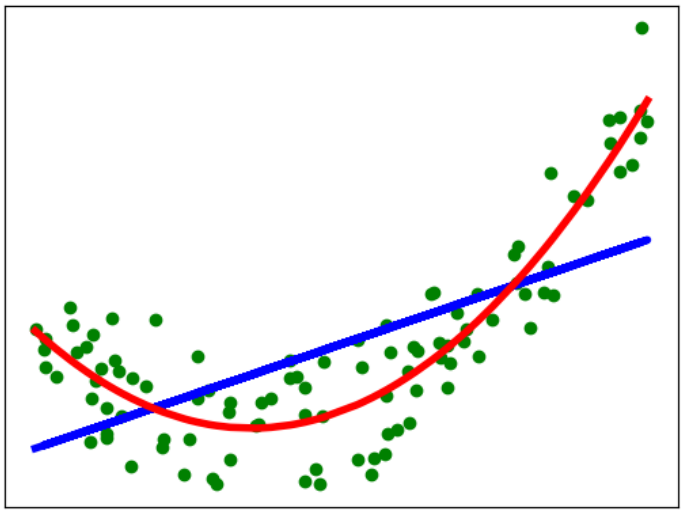
№ за списком – 22

№ варіанту – 2

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
np.random.seed(0)
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.6 * X ** 2 + X + 2 + np.random.randn(m, 1)
# Побудова графіка
plt.scatter(X, y, color='green', label='data')
# Лінійна регресія
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X, y)
y_linear_pred = linear_regression.predict(X)
# Поліноміальна регресія
polynomial_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = polynomial_features.fit_transform(X)
linear_regression_poly = linear_model.LinearRegression()
linear_regression_poly.fit(X_poly, y)
y_poly_pred = linear_regression_poly.predict(X_poly)
# Побудова графіка
plt.plot(X, y_linear_pred, color='blue', label='linear', linewidth=4)
sort_indices = np.argsort(X[:, 0])
X_sorted = X[sort_indices]
y_poly_pred_sorted = y_poly_pred[sort_indices]
plt.plot(X_sorted, y_poly_pred_sorted, color='red', label='polynomial',
linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()
print("Linear regressor performance:")
print("Mean absolute error =",
round(sm.mean_absolute_error(y, y_linear_pred), 2))
print("Mean squared error =",
round(sm.mean_squared_error(y, y_linear_pred), 2))
print("Median absolute error =",
round(sm.median_absolute_error(y, y_linear_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y,
y_linear_pred), 2))
print("R2 score =", round(sm.r2_score(y, y_linear_pred), 2))
print("\nPolinomial regressor performance:")
print("Mean absolute error =",
round(sm.mean_absolute_error(y, y_poly_pred), 2))
print("Mean squared error =",
round(sm.mean_squared_error(y, y_poly_pred), 2))
print("Median absolute error =",
round(sm.median_absolute_error(y, y_poly_pred), 2))
print("Explain variance score =", round(sm.explained_variance_score(y,
y_poly_pred), 2))
print("R2 score =", round(sm.r2_score(y, y_poly_pred), 2))
```

```
print('\n', linear_regression_poly.coef_,
linear_regression_poly.intercept_)
```

```
Linear regressor performance:
Mean absolute error = 1.46
Mean squared error = 3.24
Median absolute error = 1.22
Explain variance score = 0.41
R2 score = 0.41

Polinomial regressor performance:
Mean absolute error = 0.83
Mean squared error = 0.97
Median absolute error = 0.7
Explain variance score = 0.82
R2 score = 0.82

 [[0.97906552 0.54978823]] [2.34050076]
```

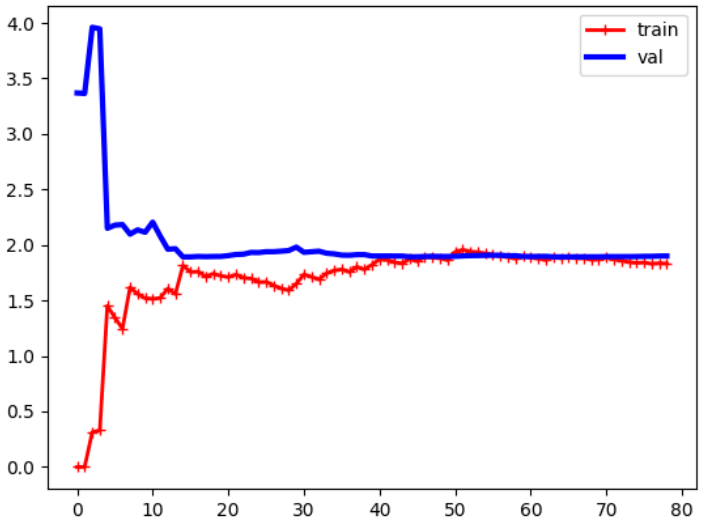


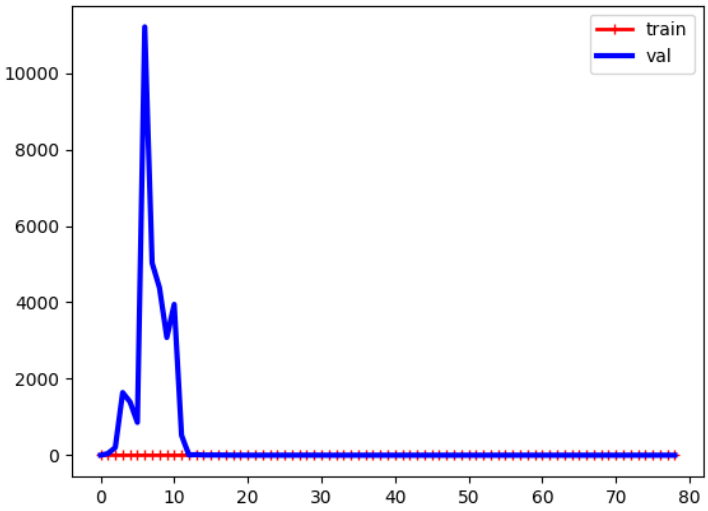
$y = 0.58208917 * x^2 + 1.03175502 * x + 2.1067843$

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.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures


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

```
# Функція для побудови кривих навчання
def plot_learning_curves(model, X, y):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)
    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.legend()
    plt.show()


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

poly_reg = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression())])
plot_learning_curves(poly_reg, X, y)
```




train
val
10000
8000
6000
4000
2000
0
0
10
20
30
40
50
60
70
80