ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

Лукашевич Влада ІПЗ-21-1

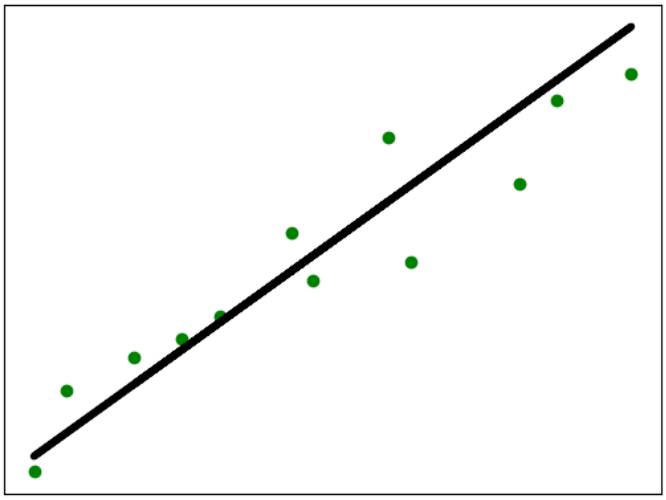
https://github.com/vladalukashevych/artificial-intelligence-systems

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
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
```



Створення лінійного регресора з послідуючим збереженням і завантаженням моделі відбулось успішно. Показник mean absolute error не змінив значення при використанні збереженої моделі.

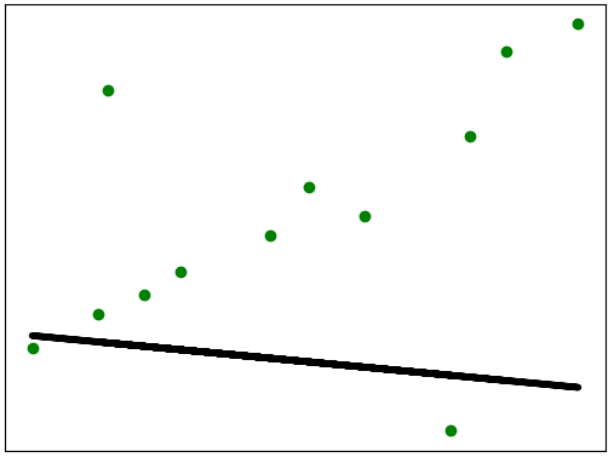
Завдання №2. Передбачення за допомогою регресії однієї змінної.

12 варіант за списком групи, тож використовуватиму файл даних data_regr_2.txt

```
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
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 = 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
```



Побудова регресійна модель на основі однієї змінної, дані використані з файлу за варіантом. Після збереження і завантаження моделі показник mean absolute error залишився незмінним при тестуванні.

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:]

# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = linear_regressor.predict(X_test)

print("Linear regressor performance based on single variable:")
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))


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

y_test_pred_mult = linear_regressor.predict(X_test)
```

```
Linear regressor performance based on single variable:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explain variance score = 0.86
R2 score = 0.86

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46633863]
```

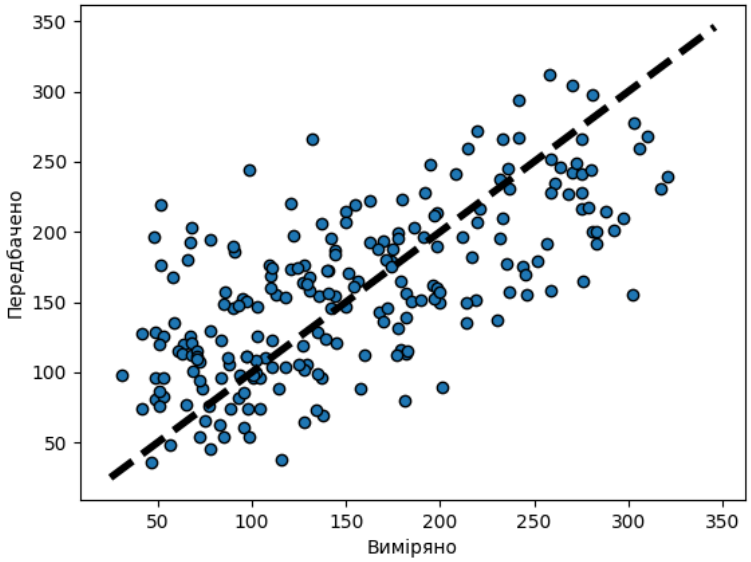
Були побудовані лінійна регресійна модель та поліноміальна модель 10 ступеня на основі багатьох змінних. Вибрано точку зі значенням наближеним до тієї, що міститься в початковому наборі даних. Порівнявши результати передбачень двох регресій зі значенням з початкового набору, побачимо, що поліноміальна регресія видала результат ближчий за значенням, тобто ближчий до вірного.

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
import sklearn.metrics as sm

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

```
y = diabetes.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5,
random_state = 0)
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
y_pred = regr.predict(X_test)



print("Regression coefficient =",
regr.coef_)
print("Regression interception =",
round(regr.intercept_, 2))
print("R2 score =",
round(sm.r2_score(y_test, y_pred), 2))
print("Mean absolute error =",
round(sm.mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error =",
round(sm.mean_squared_error(y_test, y_pred), 2))


fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, 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()
```

```
Regression coefficient = [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Regression interception = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```



Розроблено лінійний регресор, використовуючи набір даних по діабету з sklearn.datasets. Побудовано графік залежності між спостережуваними

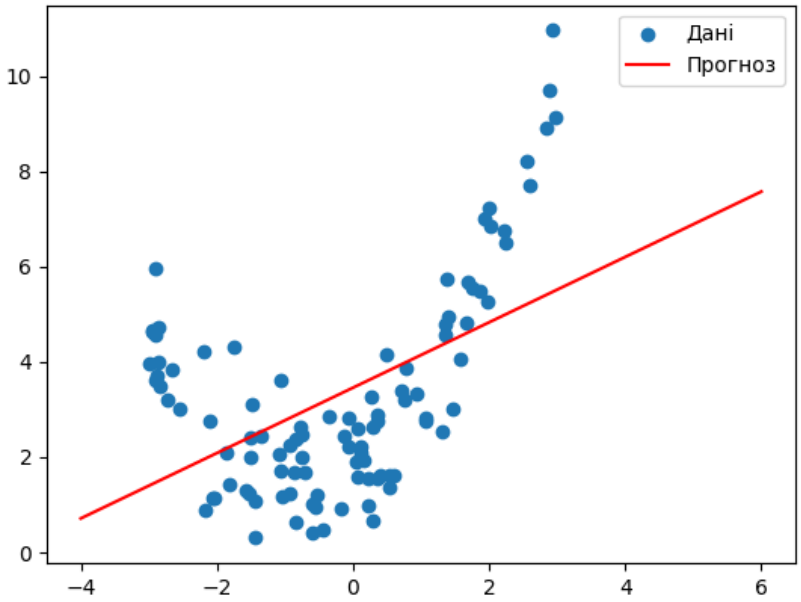
відповідями в наборі даних і відповідями, передбаченими лінійним наближенням (крапками), та пряму лінію, так як лінійна регресія мінімізує залишкову суму квадратів між початковими даними і передбачуваними.

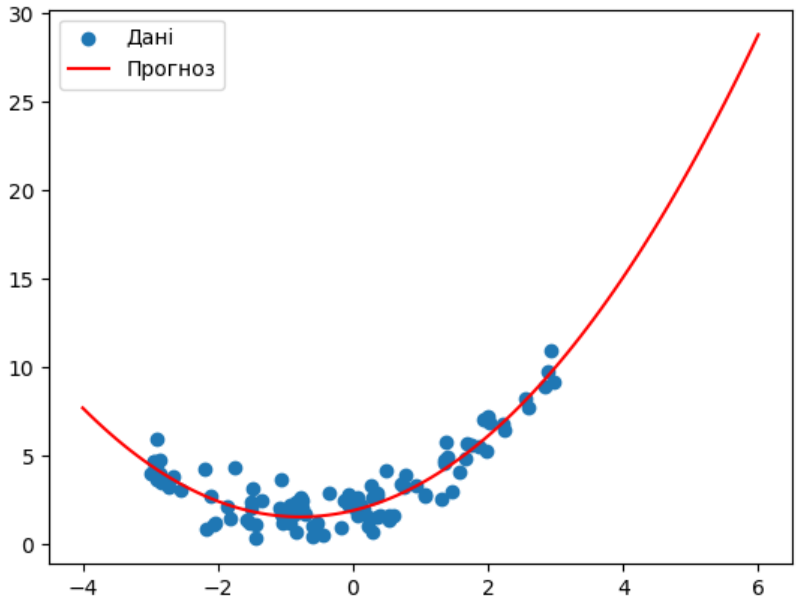
R2 = 0.44 означає, що відповідний відсоток варіацій залежної змінної можуть бути пояснені змінними моделі. Значення MAE вказує на середнє відхилення прогнозу на 44.8 одиниць від реальних даних. Показник MSE = 3075.33 є досить високим, що свідчить про наявність значних помилок у прогнозах.

Завдання №5. Самостійна побудова регресії.

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
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)

reg = linear_model.LinearRegression()
reg.fit(X, y)

X_plot = np.linspace(-4, 6, 100)
y_plot = reg.predict(X_plot.reshape(-1, 1))
plt.scatter(X, y, label="Дані")
plt.plot(X_plot, y_plot, label="Прогноз", color="red")
plt.legend()
plt.show()


poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
reg = linear_model.LinearRegression()
reg.fit(X_poly, y)

print("Features X[0]:", X[0])
print("Features after transformation:", X_poly[0])

print("Regression coefficient =", reg.coef_)
print("Regression interception =", reg.intercept_)

X_plot = np.linspace(-4, 6, 100)
y_plot = reg.predict(poly_features.transform(X_plot.reshape(-1, 1)))
plt.scatter(X, y, label="Дані")
plt.plot(X_plot, y_plot, label="Прогноз", color="red")
plt.legend()
plt.show()
```

```
Features X[0]: [-2.49794043]
Features after transformation: [-2.49794043  6.2397064 ]
Regression coefficient = [[1.03175502 0.58208917]]
Regression interception = [2.1067843]
```



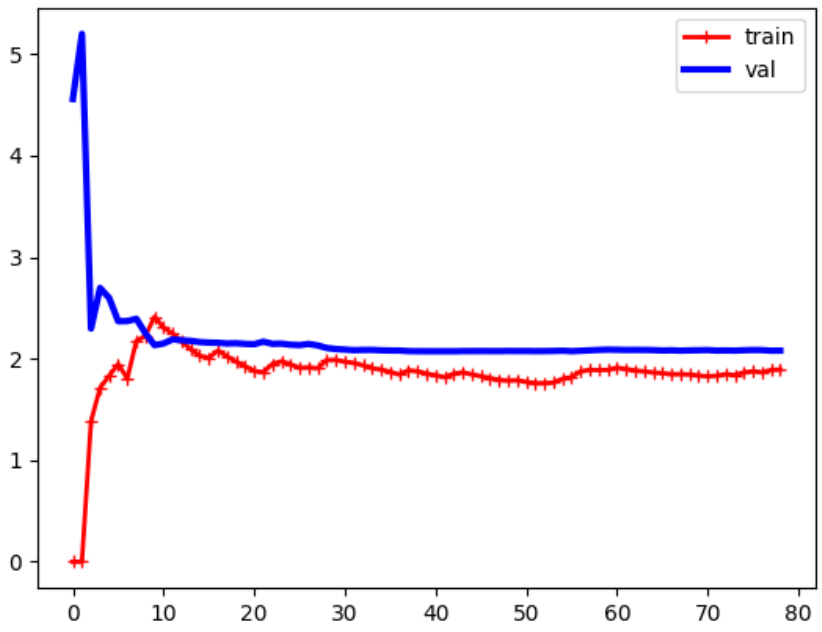


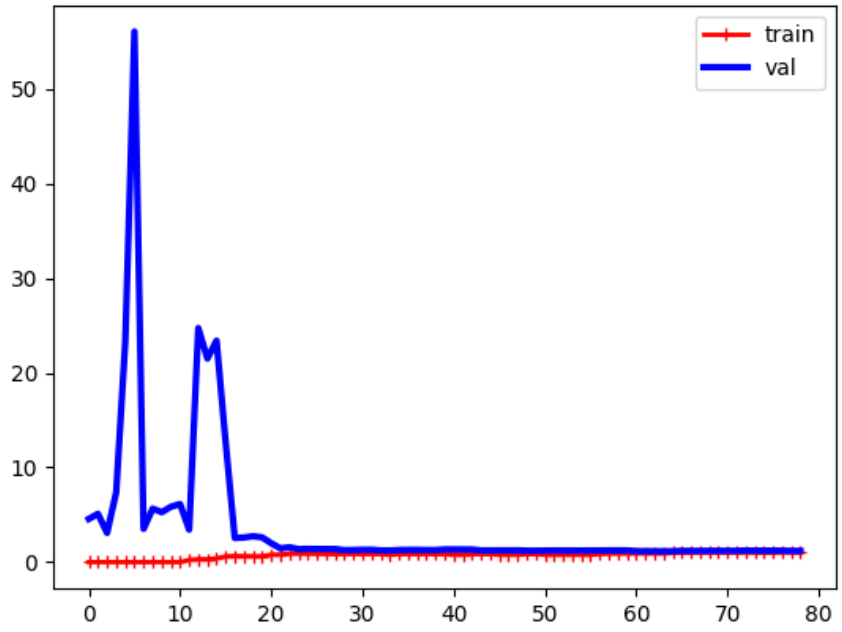
$y = 0.58208917 * x^2 + 1.03175502 * x + 2.1067843$

Побудовано лінійну та поліноміальну регресійні моделі на основі однакового набору даних, де цільові значення (залежні змінні) визначались за формулою $y = 0.6 * x^2 + x + 2 + C$.
Зробивши прогноз кожної з регресій на рандомному наборі даних зі значеннями в межах (0;6), можна побачити, що поліноміальна модель робить прогноз, який краще описує початковий набір даних, аніж лінійна.
З отриманих значень regression coefficient та interception було складено рівняння, що є наближеним до початкового.

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)
```





Висновок: виконуючи лабораторну роботу, я дослідила методи регресії даних у машинному навчанні, використовуючи спеціалізовані бібліотеки та мову програмування Python.