ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

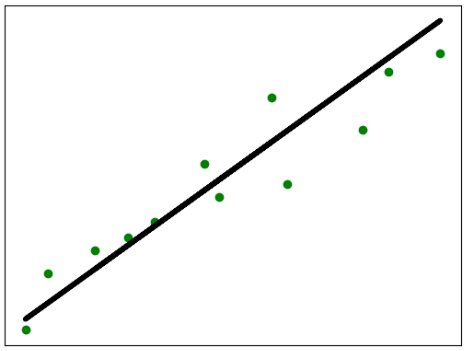
Сахно Андрій, 18 варіант

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()

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))
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab4\LR_4_task_1.py"
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
```

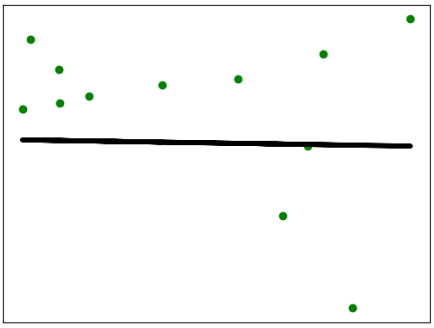


Лінійний регресор був успішно створений, збережений і завантажений для подальшого використання. Значення середньої абсолютної помилки (mean absolute error) залишилося незмінним при застосуванні завантаженої моделі.

Завдання 2.2. Передбачення за допомогою регресії 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_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.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))
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\СШІ\projs\lab4\LR_4_task_2.py"
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

Process finished with exit code 0
```



Використовуючи данні з файлу за варіантом, лінійний регресор був успішно створений, збережений і завантажений для подальшого використання. Значення середньої абсолютної помилки (mean absolute error) залишилося незмінним при застосуванні завантаженої моделі.

Завдання 2.3. Створення багатовимірного регресора

```python
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)
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\СШІ\projs\lab4\LR_4_task_3.py"
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:
 [41.46543157]

Process finished with exit code 0
```

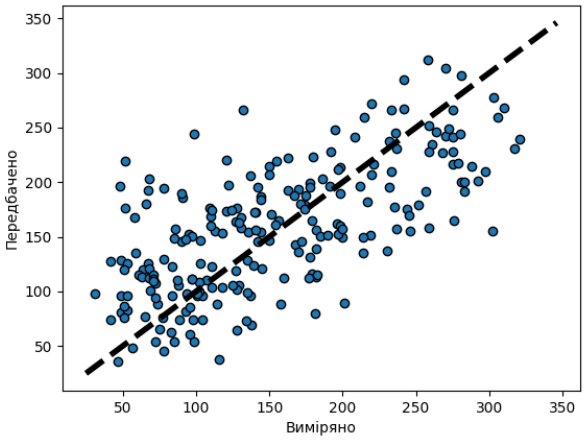
Під час виконання завдання було створено лінійну регресійну модель та поліноміальну регресійну модель 10-го ступеня для аналізу багатовимірних даних. Результати моделей були порівняні, використовуючи прогноз для конкретної точки даних, близької до значення в початковому наборі. Поліноміальна модель продемонструвала вищу точність, оскільки її прогноз

був ближчим до реального значення, що свідчить про її здатність краще враховувати нелінійні залежності в даних.

Завдання 2.4. Регресія багатьох змінних

```
import matplotlib.pyplot as plt
from sklearn import datasets, linear_model
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()
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\СШІ\projs\lab4\LR_4_task_4.py"
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

Process finished with exit code 0
```



Було розроблено лінійний регресор на основі набору даних про діабет із модуля “sklearn.datasets”. Для оцінки якості моделі побудовано графік, який ілюструє залежність між фактичними значеннями цільової змінної (спостереженнями) та значеннями, передбаченими лінійною регресією. На графіку також зображено пряму лінію, що відображає ідеальне співпадіння, оскільки лінійна регресія мінімізує залишкову суму квадратів між спостережуваними та прогнозованими значеннями.

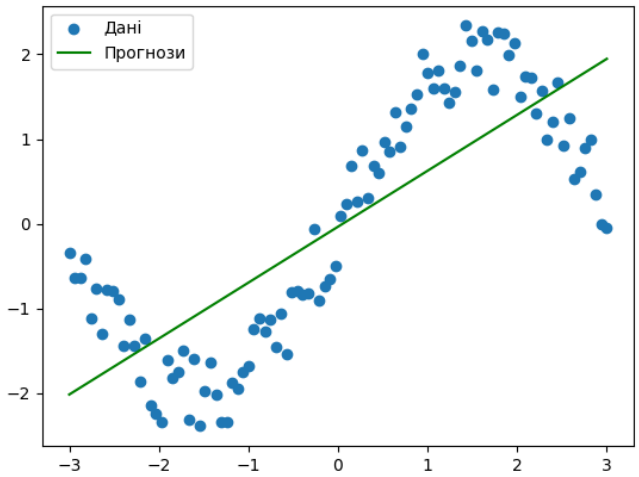
Коефіцієнт детермінації R^2 = 0.44 свідчить про те, що модель пояснює 44% варіацій залежної змінної. Середня абсолютна помилка (MAE) становить 44.8, що вказує на середнє відхилення прогнозу від реальних значень. Високе значення середньоквадратичної помилки (MSE = 3075.33) свідчить про наявність значних помилок у прогнозах, що може вказувати на обмеження лінійної моделі для цього набору даних.

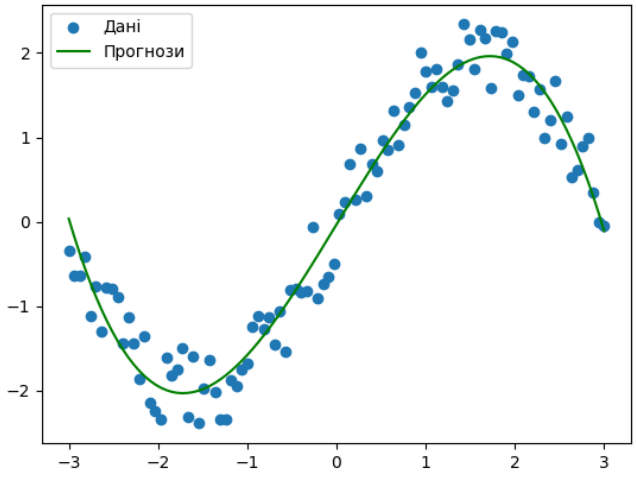
Завдання 2.5. Самостійна побудова регресії

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

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

```
X = X.reshape(-1, 1)
lin_reg = linear_model.LinearRegression()
lin_reg.fit(X, y)

X_plot = np.linspace(-3, 3, 300)
y_plot = lin_reg.predict(X_plot.reshape(-1, 1))
plt.scatter(X, y, label="Дані")
plt.plot(X_plot, y_plot, label="Прогнози", color="green")
plt.legend()
plt.show()

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

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

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

X_plot = np.linspace(-3, 3, 100)
y_plot = lin_reg.predict(poly_features.transform(X_plot.reshape(-1, 1)))
plt.scatter(X, y, label="Дані")
plt.plot(X_plot, y_plot, label="Прогнози", color="green")
plt.legend()
plt.show()
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\projs\lab4\LR_4_task_5.py"
Features X[0]: -3.0
Features after transformation: [ -3.   9. -27.]
Regression coefficient = [ 1.74135857e+00 -5.85872573e-04 -1.96226270e-01]
Regression interception = -0.03323274254651372

Process finished with exit code 0
```





y = 1.74135857·x^2 − 5.85872573·10−4·x − 0.03323274254651372

Побудував лінійну та поліноміальну регресійні моделі для даних, де цільові значення визначались як:

y = 2sin(X) + шум

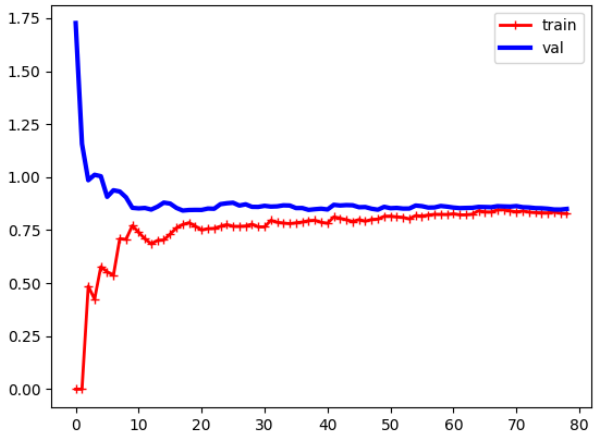
Порівнюючи коефіцієнти, отримані після навчання поліноміальної регресії, з початковими значеннями, ми бачимо, що вони наближені до значень, які створюють синусоїдальну функцію.

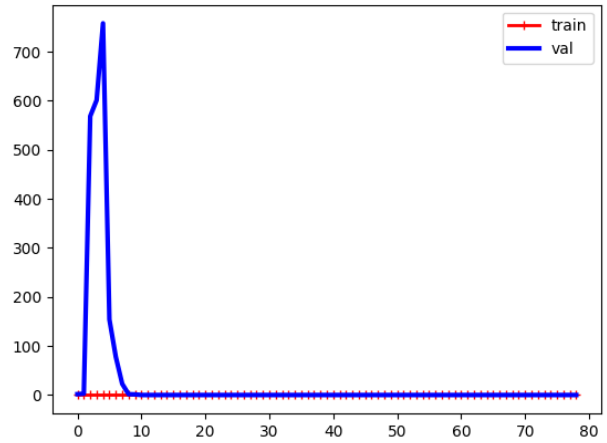
Лінійна модель не змогла адекватно описати ці дані через їх нелінійність, у той час як поліноміальна модель забезпечила кращий прогноз. Хоча поліноміальна модель не змогла точно відтворити форму синусоїди через свої обмеження, вона все ж таки дала хорошу апроксимацію залежності між змінними, що свідчить про правильність навчання моделі.

Завдання 2.5. 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 = np.linspace(-3, 3, m).reshape(-1, 1)
y = 2 * np.sin(X) + np.random.uniform(-0.5, 0.5, m)

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: На лабораторній роботі, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив різні методи регресії даних у машинному навчанні.

https://github.com/andreylion06/artificial-intelligence