

## Лабораторна робота №4

### ІПЗ-21-1 Поворознюк Ілля Варіант-14

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

#### Хід роботи:

Завдання №1: Створення регресора однієї змінної.

Лістинг програми:

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

matplotlib.use('TkAgg')
from matplotlib import 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))

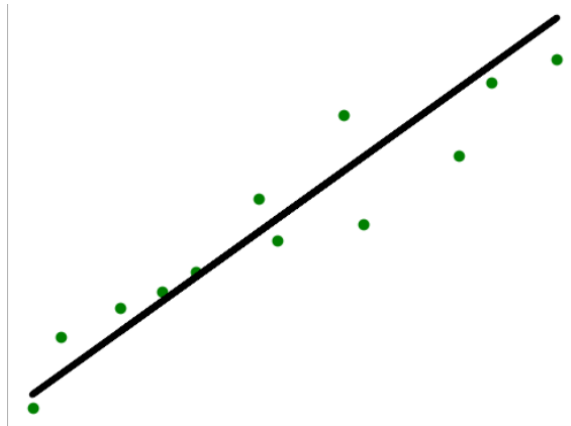
# Файл для збереження моделі
output_model_file = 'model.pkl'

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

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

# 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))

```



```

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

```

Рис. 1. Результат виконання програми

Отримані метрики свідчать про досить посередні результати для цієї регресійної моделі. Щоб досягти кращих показників, варто застосувати поліноміальну регресію.

Завдання №2:

Передбачення за допомогою регресії однієї змінної.

№ за списком 14

№ варіанту 4

Лістинг програми:

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

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt

# Вхідний файл, який містить дані
input_file = 'data_regr_4.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(0)
plt.yticks(0)
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))

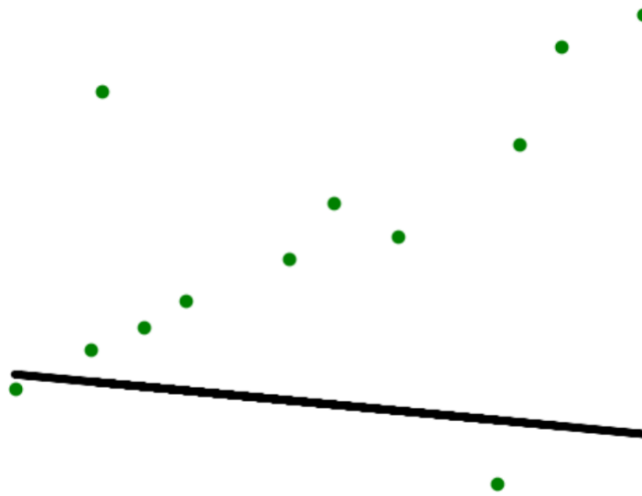
# Файл для збереження моделі
output_model_file = 'model.pkl'

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

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

# 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))

```



```

Linear regressor performance:
Mean absolute error = 2.72
Mean squared error = 13.16
Median absolute error = 1.9
Explain variance score = -0.07
R2 score = -0.07

New mean absolute error = 2.72

```

Рис. 2. Результат виконання програми

Результати свідчать про невідповідність вхідних даних. Щоб покращити роботу регресійної моделі, необхідно зібрати якісніші вхідні дані.

Завдання №3: Створення багатовимірного регресора.

Лістинг програми:

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

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt

# 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
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = 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("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)

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

# 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))

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

```

```

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

New mean absolute error = 3.58

Linear regression:
[36.05286276]

Polynomial regression:
[41.45987767]

```

Рис. 3. Результат виконання програми

Згідно отриманих результатів поліноміальний регресор забезпечує отримання кращого результату, порівняно з лінійним регресором.

Завдання №4: Регресія багатьох змінних.

Лістинг програми:

```

import matplotlib
matplotlib.use('TkAgg')

```

```

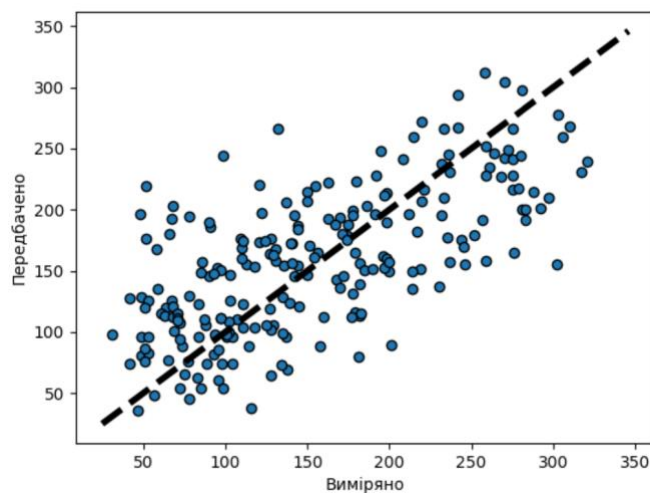
from matplotlib import pyplot as plt
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

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("Regr coef =", regr.coef_)
print("Regr intercept =", round(regr.intercept_, 2))
print("R2 score =", round(r2_score(ytest, ypred), 2))
print("Mean absolute error =", round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =", round(mean_squared_error(ytest, ypred), 2))

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()

```



```

Regr coef = [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874  23.49659361  116.36402337  843.94613929  12.71856131]
Regr intercept = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33

```

Рис. 4. Результат виконання програми

Отримані метрики вказують на доволі посередні результати для даної регресійної моделі. Для отримання кращого результату необхідно використовувати поліноміальний регресор.

Завдання №5: Самостійна побудова регресії.

№ за списком 14

№ варіанту 4

Лістинг програми:

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

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt

m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 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.title("Лінійна регресія")
plt.plot(X_test, y_test_pred, color='black', linewidth=4)
plt.xticks(())
plt.yticks(())

try:
    plt.show()
except _tkinter.TclError:
```



```

pass

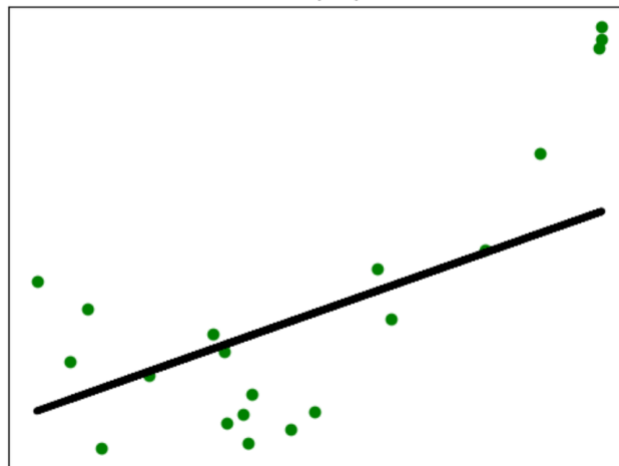
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(X.reshape(-1, 1))
poly_reg_model = linear_model.LinearRegression()
poly_reg_model.fit(poly_features, y)
y_predicted = poly_reg_model.predict(poly_features)
plt.title("Поліноміальна регресія")
plt.scatter(X, y)
plt.plot(X, y_predicted, c="red")

try:
    plt.show()
except_tkinter.TclError:
    pass

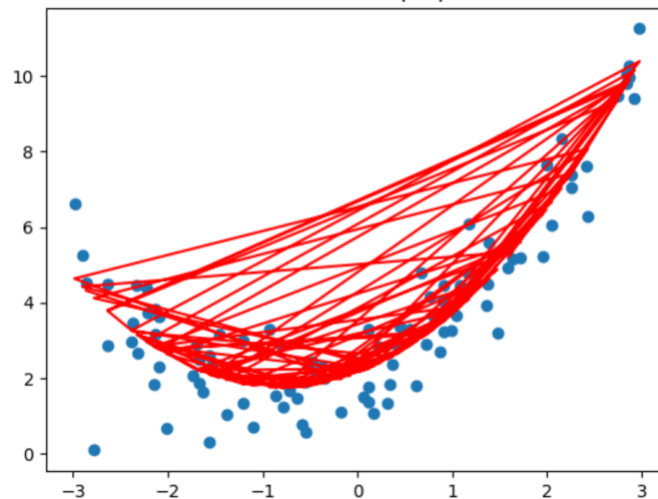
print("Intercept = ", poly_reg_model.intercept_)
print("Coef = ", poly_reg_model.coef_)

```

Лінійна регресія



Поліноміальна регресія



```
Intercept = [3.14173441]
Coef = [[1.43486965 0.79386774]]
```

Рис. 5. Результат виконання програми

Початкова модель:  $y = 0,7x + 2x + 3 + \text{гауссів шум}$ .

Отримана модель регресії:  $y = 0,79x^2 + 1,43x + 3,14$ .

Отримані коефіцієнти близькі до модельних. І це буде означає що модель навчена правильно.

Завдання №6: Побудова кривих навчання.

Лістинг програми:

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

matplotlib.use('TkAgg')
from matplotlib import pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

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.legend(['Навчальний набір', 'Перевіряючий набір'])
    plt.show()

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

polynomial_regression = Pipeline([
    ("poly_features",
     PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", linear_model.LinearRegression())
])
```

```
]
plot_learning_curves(polynomial_regression, X, y)
```

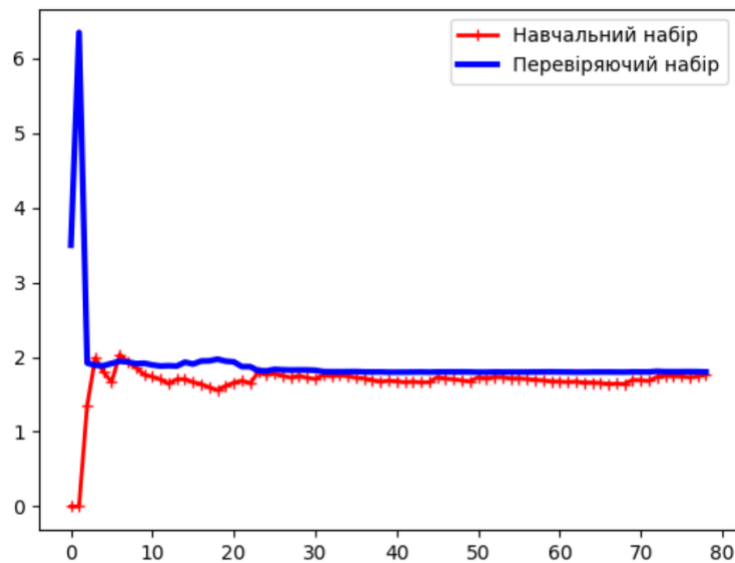


Рис. 6. Результат виконання програми

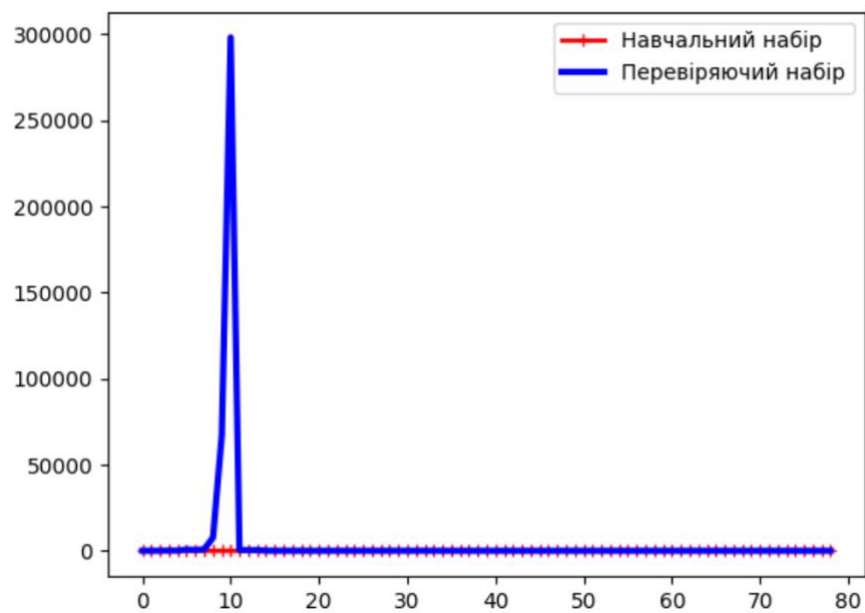


Рис. 7. Результат виконання програми

**Висновки:** було досліджено методи регресії та неконтрольованої класифікації даних у машинному навчанні, використовуючи спеціалізовані бібліотеки і мову програмування Python.