Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

Завдання №1:

Напишемо код для регресора однієї змінної:

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
# Input file containing data
input_file = 'data_singlevar_regr.txt'
# Read data
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
# Train and test split
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# Training data
X_train, y_train = X[:num_training], y[:num_training]
# Test data
X_test, y_test = X[num_training:], y[num_training:]
# Create linear regressor object
regressor = linear_model.LinearRegression()
# Train the model using the training sets
regressor.fit(X_train, y_train)
# Predict the output
y_test_pred = regressor.predict(X_test)
# Plot outputs
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()
# Compute performance metrics
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))
# Model persistence
output_model_file = 'model.pkl'
# Save the model
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
# Load the model
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
# Perform prediction on test data
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))
```

Результати:

```
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. Результати оцінки якості регресора однієї змінної

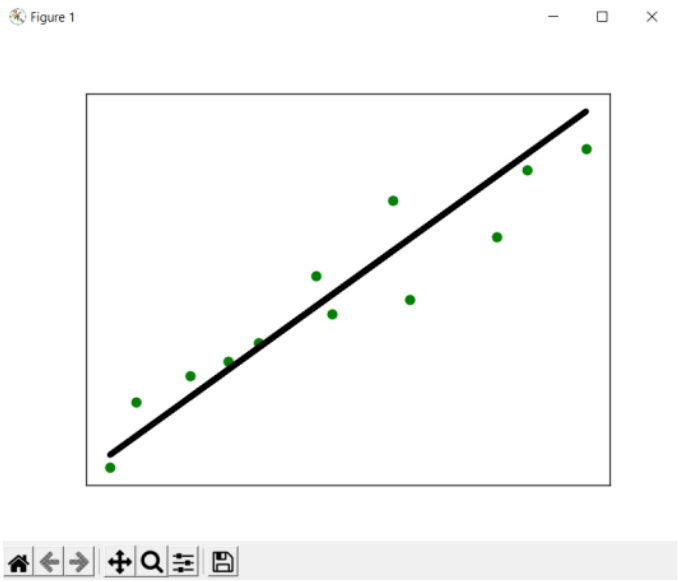

Рис. 2. Результати графіка регресора однієї змінної

Так, з урахуванням оцінок якості та аналізу графіка, можна зробити висновок, що суттєвих відхилень у результатах не спостерігається.

Завдання №2:

Код побудованої регресії за своїм варіантом:

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
# Input file containing data
input_file = 'data_regr_14.txt'
# Read data
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
# Train and test split
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# Training data
X_train, y_train = X[:num_training], y[:num_training]
# Test data
X_test, y_test = X[num_training:], y[num_training:]
# Create linear regressor object
regressor = linear_model.LinearRegression()
# Train the model using the training sets
regressor.fit(X_train, y_train)
# Predict the output
y_test_pred = regressor.predict(X_test)
# Plot outputs
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()
# Compute performance metrics
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))
# Model persistence
output_model_file = 'model.pkl'
# Save the model
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
# Load the model
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
# Perform prediction on test data
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))
```

Результати:

```
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. 3. Результати оцінки якості регресора однієї змінної за своїм варіантом

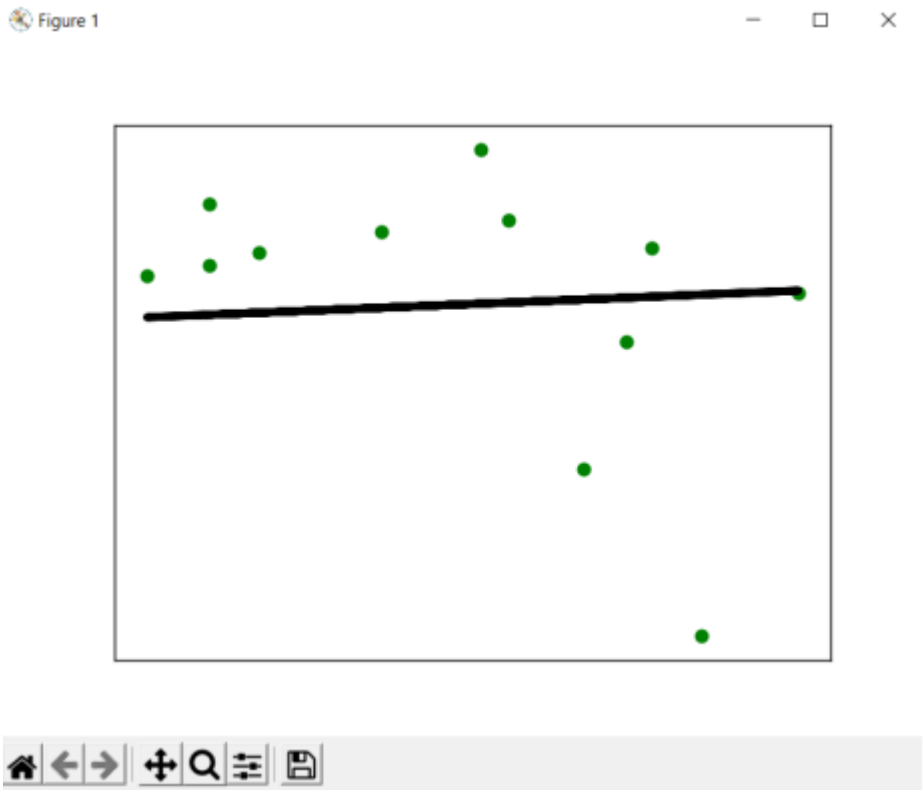

Рис. 4. Результати графіка регресора однієї змінної за своїм варіантом

Отже, судячи з оцінок якості та графіка, можемо спостерігати значні відхилення.

Завдання №3:

Код поліноміального регресора:

```
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
# Input file containing data
input_file = 'data_multivar_regr.txt'
# Load the data from the input file
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
# Split data into training and testing
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# Training data
X_train, y_train = X[:num_training], y[:num_training]
# Test data
X_test, y_test = X[num_training:], y[num_training:]
# Create the linear regressor model
linear_regressor = linear_model.LinearRegression()
# Train the model using the training sets
linear_regressor.fit(X_train, y_train)
```

```
# Predict the output
y_test_pred = linear_regressor.predict(X_test)
# Measure performance
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 regression
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))
```

Результат:

```
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:
 [41.46504705]
```

Рис. 5. Результати оцінки якості поліноміального регресора

Завдання №4:

Напишемо код для лінійного регресора, використвуючи набір даних по діабету:

```
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("Linear Regressor performance:")
print("Mean absolute error =", np.round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =", np.round(mean_squared_error(ytest, ypred), 2))
print("R2 score =", np.round(r2_score(ytest, ypred), 2))
print("Coefficients = ", regr.coef_)
print("Intercept = ", regr.intercept_)
ig, 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()
```

Результат:

```
Linear Regressor performance:
Mean absolute error = 44.8
Mean squared error = 3075.33
R2 score = 0.44
Coefficients =  [ -20.41129305 -265.88594023  564.64844662  325.55650029 -692.23796104
  395.62249978   23.52910434  116.37102129  843.98257585   12.71981044]
Intercept =  154.35898821355153
```

Рис. 6. Результати лінійного регресора, використовуючи дані по діабету

Завдання №5:

Напишемо код для побудови регресій, використовуючи дані за своїм варіантом:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
np.random.seed(42)
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)
```

```
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
print("X[0] = ", X[0])
print("X_poly = ", X_poly)
lin_reg = linear_model.LinearRegression()
lin_reg.fit(X_poly, y)
print("Coefficients = ", lin_reg.coef_)
print("Intercept = ", lin_reg.intercept_)
X_new = np.linspace(-3, 3, 100).reshape(100, 1)
X_new_poly = poly_features.transform(X_new)
y_new = lin_reg.predict(X_new_poly)
plt.plot(X, y, "b.")
plt.plot(X_new, y_new, "r-", linewidth=2, label="Predictions")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.axis([-3, 3, 0, 10])
plt.show()
```

Результати:

```
Coefficients =  [[1.19191947 0.76456263]]
Intercept =  [2.90693421]
```

Рис. 7. Коефіцієнти за створеною моделлю

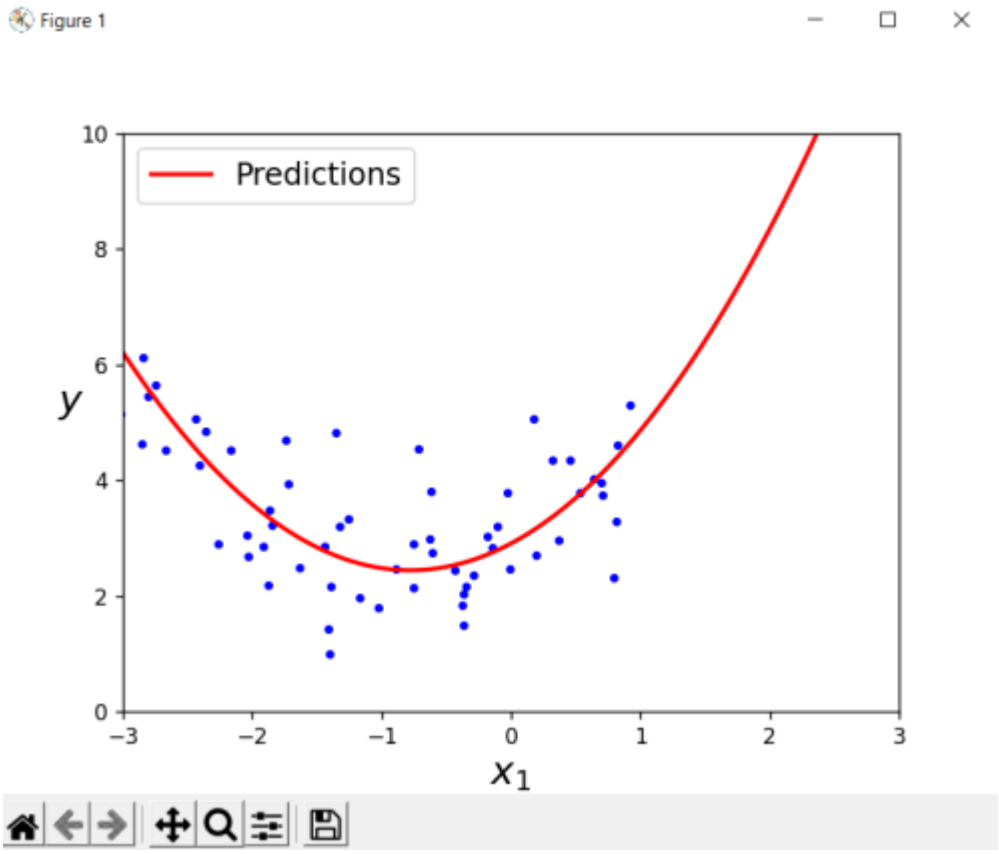

Рис. 8. Отриманий графік, використовуючи створену модель та дані за варіантом

Отже, запишемо модель у вигляді математичного рівняння: $y = 0.76x_1{}^2 + 1.19x_1 + 2.9$.

Завдання №6:

Код створення кривих навчання моделі для встановлених навчальних даних:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
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=10)
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[:m], y_train_predict))
    val_errors.append(mean_squared_error(y_val, y_val_predict))
    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="train")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")
    plt.legend(loc="upper right", fontsize=14)
    plt.xlabel("Training set size", fontsize=14)
    plt.ylabel("RMSE", fontsize=14)
    plt.axis([0, 80, 0, 3])
    plt.show()
np.random.seed(42)
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)
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)
plot_learning_curves(lin_reg, X, y)
polynomial_regression = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", linear_model.LinearRegression()),
])
plot_learning_curves(polynomial_regression, X, y)
```

Результати:

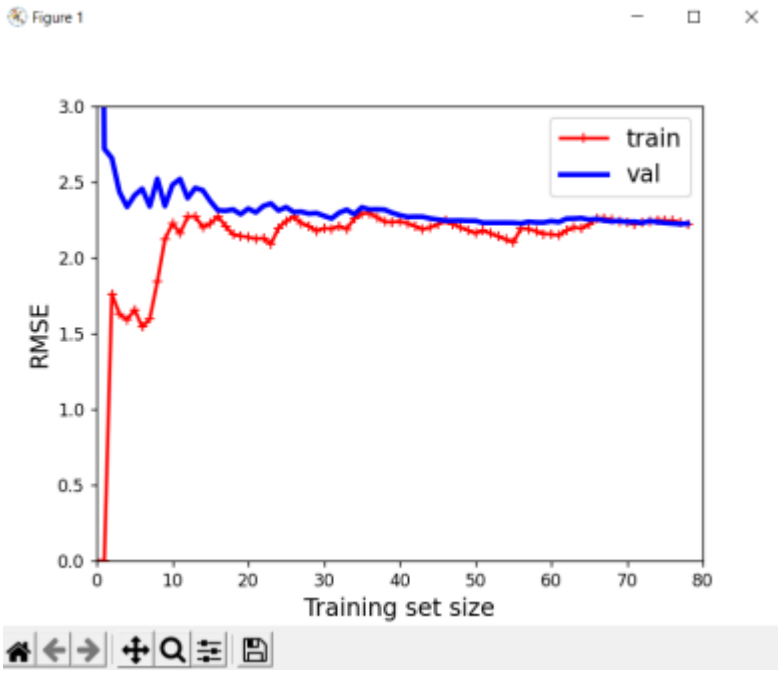

Рис. 9. Криві навчання для лінійної моделі

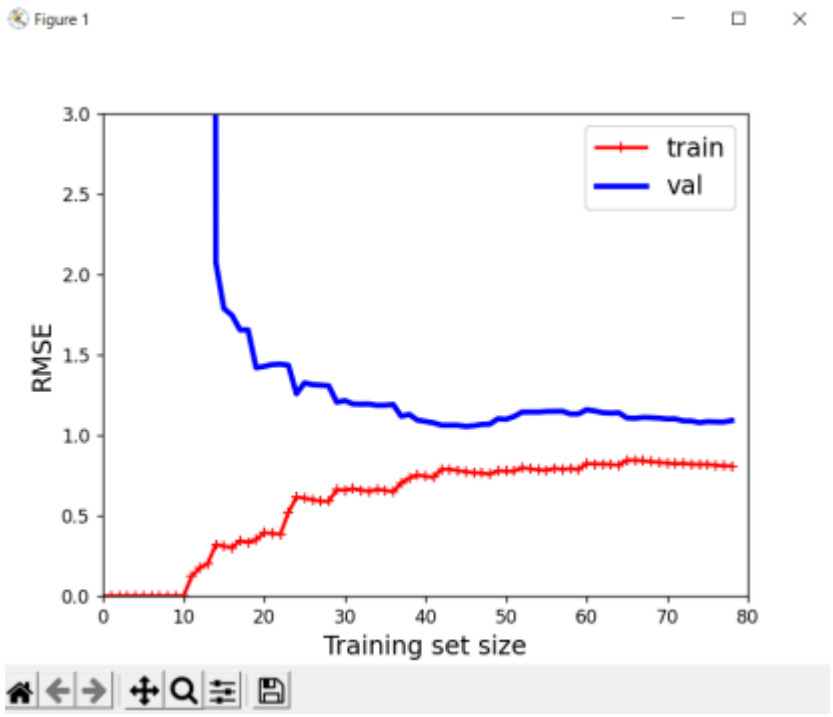

Рис. 10. Криві навчання для поліноміальної моделі 10-ступеня

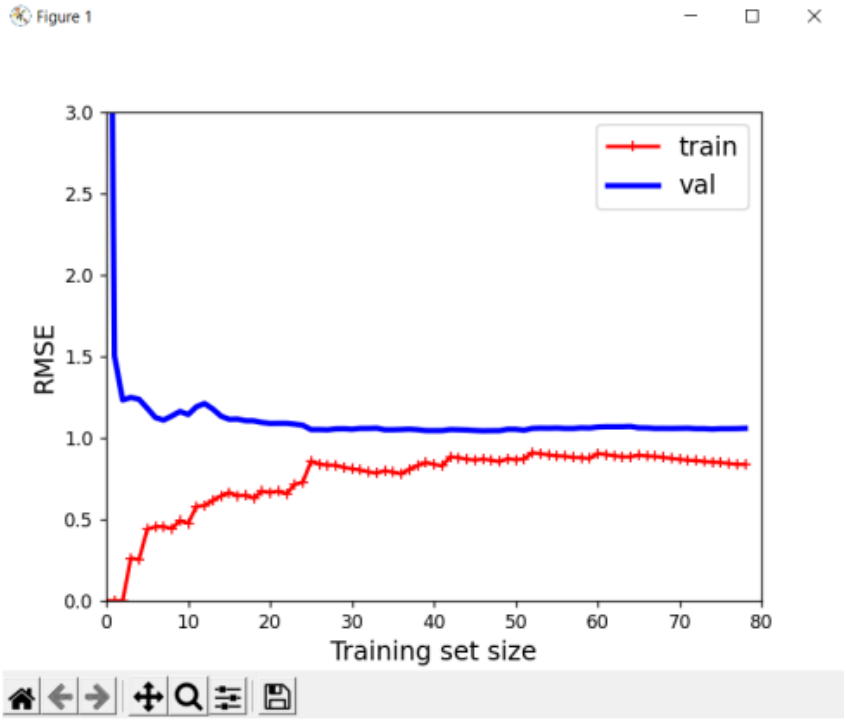

Рис. 11. Криві навчання для поліноміальної моделі 2-ступеня

Висновок: на цій лабораторній роботі я, використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідив методи регресії даних у машинному навчанні.