**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:**

Лістинг коду:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
input\_file = "data\_singlevar\_regr.txt"  
data = np.loadtxt(input\_file, delimiter=",")  
  
X, y = data[:, :-1], data[:, -1]  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
y\_test\_pred = regressor.predict(X\_test)  
  
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))  
  
output\_model\_file = "model.pkl"  
with open(output\_model\_file, "wb") as f:  
 pickle.dump(regressor, f)  
  
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),  
)

A black screen with white text

Description automatically generated

Рис. 1. Результат виконання програми

A line with green dots

Description automatically generated

Рис. 2. Результат виконання програми

Лінійний регресор ефективно моделює залежність на основі однієї змінної, хоча окремі відхилення від лінії можуть свідчити про наявність шуму в даних.

**Завдання 2:**

Лістинг коду:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
input\_file = "data\_regr\_3.txt"  
data = np.loadtxt(input\_file, delimiter=",")  
X, y = data[:, :-1], data[:, -1]  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
y\_test\_pred = regressor.predict(X\_test)  
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))  
output\_model\_file = "model\_2.pkl"  
with open(output\_model\_file, "wb") as f:  
 pickle.dump(regressor, f)  
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),  
)

A black screen with white text

Description automatically generated

Рис. 3. Результат виконання програми

A line with green dots

Description automatically generated

Рис. 4. Результат виконання програми

Модель значно не відповідає даним, що найбільш очевидно демонструє негативний показник R2. Це вказує на те, що точність передбачень моделі є гіршою, ніж у випадкових прогнозів.

**Завдання 3:**

Лістинг коду:

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
  
data = np.loadtxt("data\_multivar\_regr.txt", delimiter=",")  
X, y = data[:, :-1], data[:, -1]  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
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))  
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))

A screenshot of a computer

Description automatically generated

Рис. 5. Результат виконання програми

Обидві моделі добре відповідають цим даним, однак поліноміальна регресія демонструє кращу точність прогнозування, що вказує на наявність складних трендів у наборі даних.

**Завдання 4:**

Лістинг коду:

import matplotlib.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 =", regr.intercept\_)  
print("Mean absolute error =", round(mean\_absolute\_error(ytrain, ypred), 2))  
print("Mean squared error =", round(mean\_squared\_error(ytrain, ypred), 2))  
print("R2 score =", round(r2\_score(ytrain, 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()

A black screen with white numbers

Description automatically generated

Рис. 6. Результат виконання програми

A graph showing a line with blue dots

Description automatically generated with medium confidence

Рис. 7. Результат виконання програми

Модель виявляється непридатною для цих даних.

**Завдання 5:**

Лістинг коду:

import numpy as np  
import matplotlib.pyplot as plt  
import sklearn.metrics as sm  
from sklearn import linear\_model  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import PolynomialFeatures  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 4  
y = 0.5 \* X\*\*2 + X + 2 + np.random.randn(m, 1)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
polynomial = PolynomialFeatures(degree=2, include\_bias=False)  
X\_train\_transformed = polynomial.fit\_transform(X\_train)  
model = linear\_model.LinearRegression()  
model.fit(X\_train\_transformed, y\_train)  
X\_test\_transformed = polynomial.fit\_transform(X\_test)  
y\_test\_predict = model.predict(X\_test\_transformed)  
print("Polynomial сoefficient:\n", model.coef\_, model.intercept\_)  
print("\nPolynomial regressor performance:")  
print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_predict), 2))  
print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_predict), 2))  
print(  
 "Median absolute error =",  
 round(sm.median\_absolute\_error(y\_test, y\_test\_predict), 2),  
)  
print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_predict), 2))  
print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_predict), 2))  
fig, ax = plt.subplots()  
ax.scatter(y\_test, y\_test\_predict, edgecolors=(0, 0, 0))  
ax.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], "k--", lw=4)  
ax.set\_xlabel("Виміряно")  
ax.set\_ylabel("Передбачено")  
plt.show()

A computer screen with white text

Description automatically generated

Рис. 8. Результат виконання програми

A graph with blue dots

Description automatically generated

Рис. 9. Результат виконання програми

Модель у вигляді математичного рівняння: y = 0.5x21 + x2 + 2 + *гаусовий шум.*

Коефіцієнти нашої моделі: y = 0.536x21 + 1.072x2 + 1.78

**Завдання 6:**

Лістинг коду:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import mean\_squared\_error  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.pipeline import Pipeline  
  
def generate\_data(m, random\_state=None):  
 if random\_state is not None:  
 np.random.seed(random\_state)  
 X = 6 \* np.random.rand(m, 1) - 4  
 y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
 return X, y  
  
def plot\_learning\_curves(model, X\_train, y\_train, X\_test, y\_test):  
 train\_errors = []  
 test\_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\_test\_predict = model.predict(X\_test)  
 train\_errors.append(mean\_squared\_error(y\_train\_predict, y\_train[:m]))  
 test\_errors.append(mean\_squared\_error(y\_test\_predict, y\_test))  
  
 plt.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label="Train")  
 plt.plot(np.sqrt(test\_errors), "b-", linewidth=3, label="Test")  
 plt.xlabel("Training set size")  
 plt.ylabel("RMSE")  
 plt.legend()  
 plt.title("Learning Curves")  
 plt.show()  
  
def create\_polynomial\_pipeline(degree):  
 return Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=degree, include\_bias=False)),  
 ("lin\_reg", LinearRegression()),  
 ])  
  
  
X, y = generate\_data(100, random\_state=42)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)  
  
lin\_reg = LinearRegression()  
plot\_learning\_curves(lin\_reg, X\_train, y\_train, X\_test, y\_test)  
  
degrees = [10, 2]  
for degree in degrees:  
 poly\_pipeline = create\_polynomial\_pipeline(degree)  
 plot\_learning\_curves(poly\_pipeline, X\_train, y\_train, X\_test, y\_test)

A graph showing the results of training

Description automatically generated

Рис. 10. Результат виконання програми

A graph of a training set

Description automatically generated

Рис. 11. Результат виконання програми

A graph showing the results of a training set

Description automatically generated

Рис. 12. Результат виконання програми

Посилання на репозиторій на GitHub: <https://github.com/vladyslavgeyna/artificial-intelligence-systems/tree/main/lab3>.

***Висновки:*** в ході виконання лабораторної роботи ми,використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили методи регресії даних у машинному навчанні.