Лабораторна робота №4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Посилання на гітхаб: https://github.com/PanchukPetro/SShILabsPanchuk/tree/main/Lab4

ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

Завдання 2.1. Створення регресора однієї змінної

Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

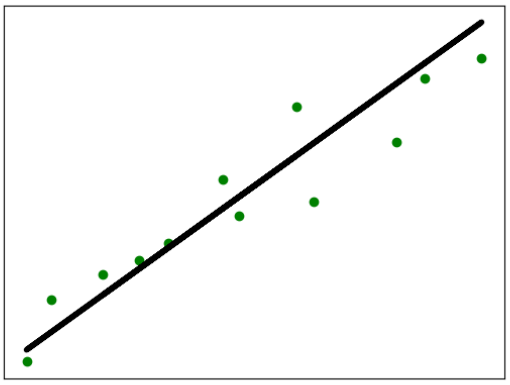
Програмний код:

```
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
```

Результати оцінювання



Графік

Висновок: Значення R2 показує число доволі близьке до 1, що свідчить про достатньо високу точність моделі лінійної регресії. На графіку видно, що модель, яка зображена лінією, вцілому передбачає тенденцію за якою йдуть дані, але певні відхилення є, бо модель є лінійною і не може врахувати нелінійні зв'язки між змінними.

Завдання 2.2. Передбачення за допомогою регресії однієї змінної

Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі

№ списку 17, варіант 2

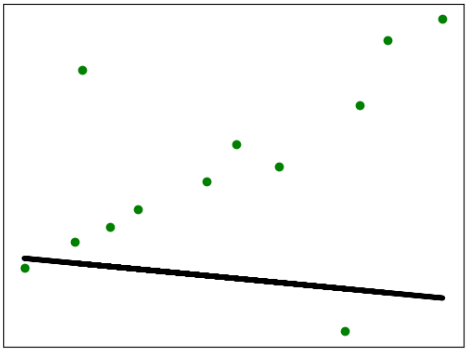
№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	1	2	3	4	5

Програмний код:

```
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 = 'model2.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
```

Результат оцінювання



Графік

Висновок: Не дивлячись на високі показники метрик R2, MAE, MSE, що мали б свідчити про високу точність моделі, на графіку видно, що модель не відображає загальну тенденцію даних.

Завдання 2.3. Створення багатовимірного регресора

Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Програмний код:

```
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
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))

# 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

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46177229]
```

Результати оцінювання

Завдання 2.4. Регресія багатьох змінних

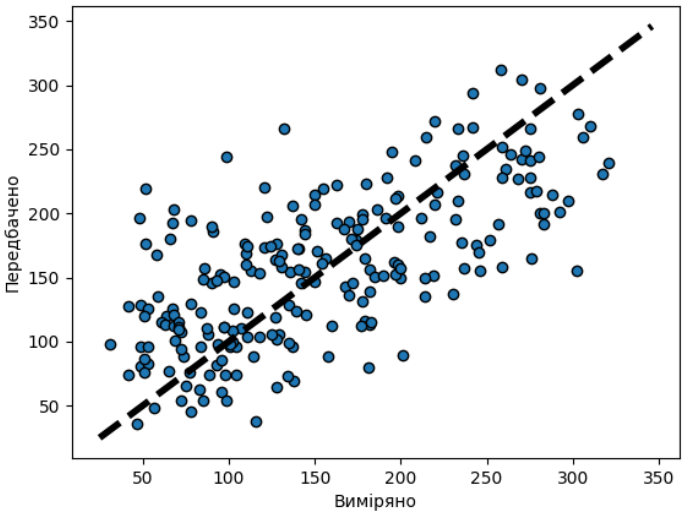
Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets.

Програмний код:

```
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

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 =",
round(sm.mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =",
round(sm.mean squared error(ytest, ypred), 2))
print("Median absolute error =",
round(sm.median_absolute_error(ytest, ypred), 2))
print("Explain variance score =",
round(sm.explained_variance_score(ytest, ypred), 2))
print("R2 score =", round(sm.r2 score(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()
```

```
Linear regressor performance:
Mean absolute error = 44.8
Mean squared error = 3075.33
Median absolute error = 38.21
Explain variance score = 0.44
R2 score = 0.44
```

Результати оцінювання якості



Отриманий графік

Висновок: R2=0.44 свідчить про те, що модель пояснює лише 44% варіації цільової змінної, що є середнім результатом. Середня абсолютна помилка становить 44.8. Це доволі середні значення. По графіку видно, що багато точок лежать далеко від лінії, модель передбачає значення не дуже добре.

Завдання 2.5. Самостійна побудова регресії

Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

Варіант 7

№ за списком	11	12	13	14	15	16	17	18	19	20
№ варіанту	1	2	3	4	5	6	7	8	9	10

Варіант 7

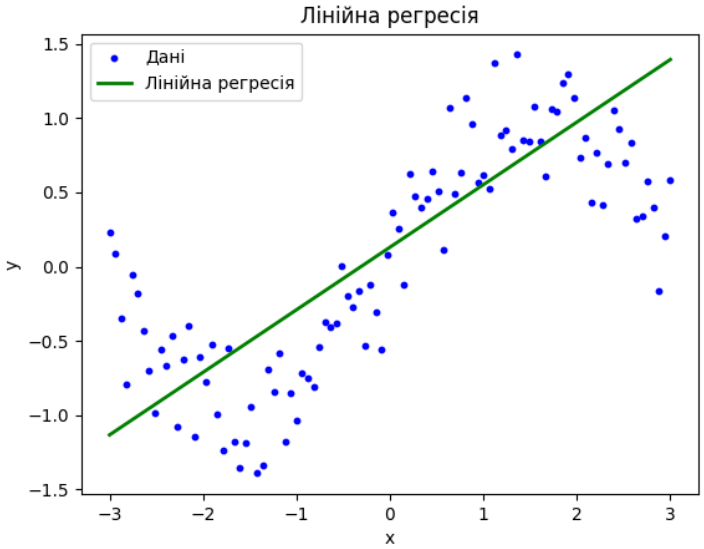
```
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

Рівняння моделі: $y = 0.1855x^2 + 0.6259x - 0.1722$

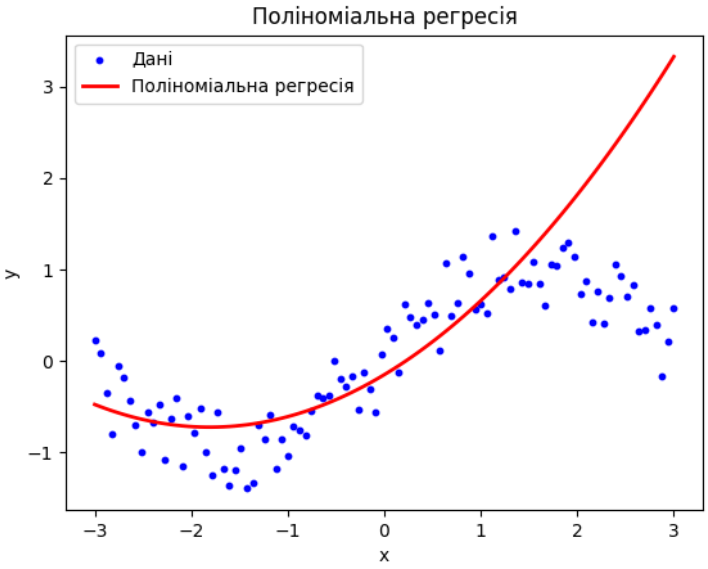
Програмний код:

```
import pickle
import numpy as np
from sklearn import linear model
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt

# Створення даних
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

X = X.reshape(-1, 1)

# Розділення даних
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:]

# Лінійна регресія
linear model reg = linear model.LinearRegression()
linear model reg.fit(X train, y train)

# Прогноз лінійної регресії
y_linear_pred = linear_model_reg.predict(X)

# Побудова графіка для лінійної регресії
plt.scatter(X, y, color='blue', s=10, label='Дані')  # Крапки
plt.plot(X, y linear pred, color='green', linewidth=2, label='Лінійна регресія')  # Лінійна крива
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.title("Лінійна регресія")
plt.show()

# Поліноміальна регресія
degree = 2
poly_model = make_pipeline(PolynomialFeatures(degree), linear_model.LinearRegression())
poly model.fit(X train, y train)

# Прогноз поліноміальної регресії
y_poly_pred = poly_model.predict(X)

# Побудова графіка для поліноміальної регресії
plt.scatter(X, y, color='blue', s=10, label='Дані')  # Крапки
plt.plot(X, y_poly_pred, color='red', linewidth=2, label='Поліноміальна регресія')  #
Поліноміальна крива
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.legend()
plt.title("Поліноміальна регресія")
plt.show()
```



Отриманий графік лінійної регресії



Отриманий графік поліноміальної регресії

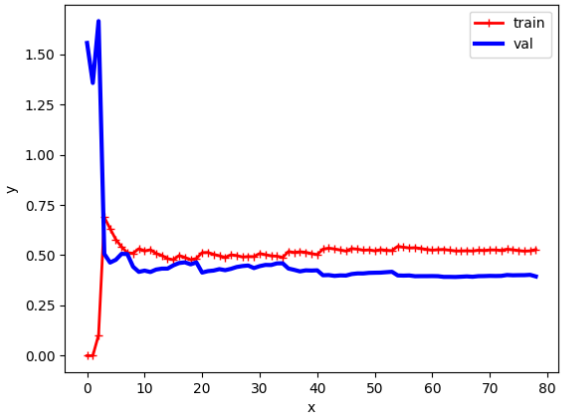
Висновок: Було побудовано моделі лінійної та поліноміальної регресії. Лінійна регресія погано описує дані через їхню нелінійну залежність. Поліноміальна регресія другого ступеня значно краще відповідає розподілу точок, що підтверджує її придатність для аналізу таких даних.

Завдання 2.6. Побудова кривих навчання

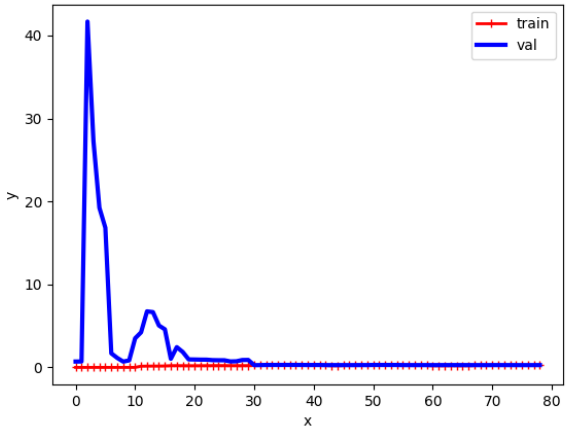
Побудуйте криві навчання для ваших даних у попередньому завданні.

Програмний код:

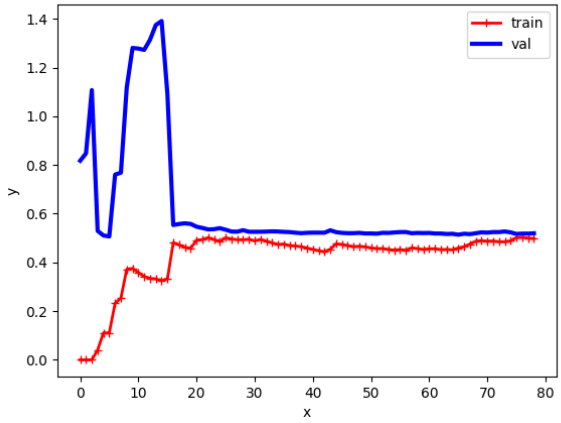
```
from sklearn.metrics import mean squared error
from sklearn.model selection import train test split
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear model
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

def plot_learning_curves(model,X,y):
    X_train,X_val,y_train,y_val = train_test_split(X,y,test_size=0.2)
    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.xlabel("x")
    plt.ylabel("y")
    plt.legend()
    plt.show()


# Створення даних
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

X = X.reshape(-1, 1)

# Розділення даних
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:]

lin reg = linear model.LinearRegression()
plot learning curves(lin reg,X,y)

polynominal_regression = Pipeline([
    ("poly_features",
PolynomialFeatures(degree=10,include bias=False)),("lin reg",linear model.LinearRegression()),
])
plot_learning_curves(polynominal_regression,X,y)
polynominal_regression2 = Pipeline([
    ("poly_features",
PolynomialFeatures(degree=2,include_bias=False)),("lin_reg",linear_model.LinearRegression()),
])
plot_learning_curves(polynominal_regression2,X,y)
```



Графік кривих навчання для лінійної моделі



Графік кривих навчання для поліноміальної моделі 10 ступня



Графік кривих навчання для поліноміальної моделі 2 ступня

Висновки:

Лінійна модель: має високу похибку як на тренувальній, так і на валідаційній вибірках, що вказує на недоадаптацію (underfitting) через низьку складність моделі.

Поліноміальна модель 2-го ступеня: показує кращий баланс між похибками тренувальної та валідаційної вибірок, що свідчить про її адекватну складність для опису даних.

Поліноміальна модель 10-го ступеня: має низьку похибку на тренувальній вибірці, але значно більшу на валідаційній, що вказує на перетренування через надмірну складність моделі.