ЛАБОРАТОРНА РОБОТА № 3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

GitHub репозиторій: https://github.com/VovaYanko/SAI

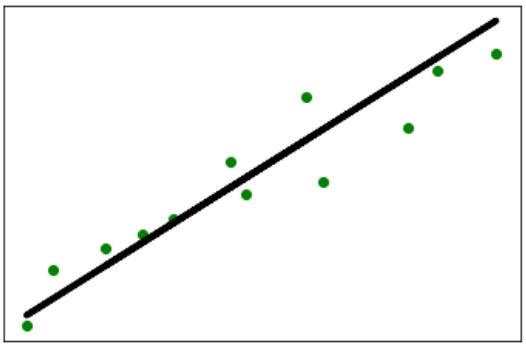
Завдання 1

Код програми:

```
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)

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

Графік функції та оцінки якості наведені на рисунку 1.1.



```
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. 1.1. Графік функції та оцінки якості

На основі отриманих даних можемо сказати, що модель добре сравляється з поставленим завданням (R2 score), та немає величезних викидів відповідно до графіку та даних отриманих з MAE та MSE.

Завдання 2

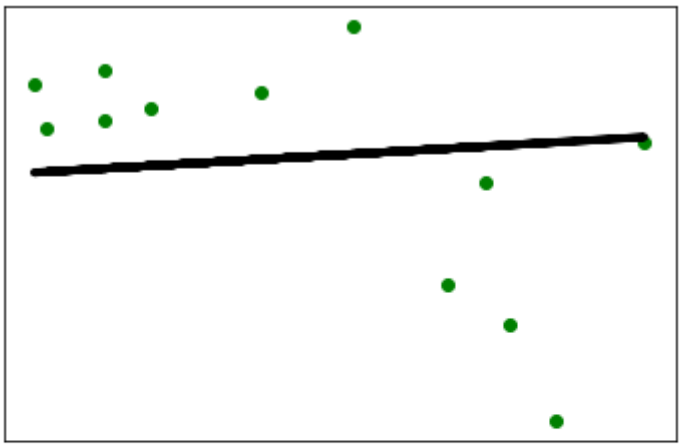
№ варіанта за списком 20, номер варіанта 5

Код програми:

```
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_5.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)

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

Графік функції та оцінки якості наведені на рисунку 1.2.



```
Linear regressor performance:
Mean absolute error = 3.31
Mean squared error = 16.98
Median absolute error = 2.66
Explain variance score = -0.14
R2 score = -0.15

New mean absolute error = 3.31
```

Рис. 1.2. Графік функції та оцінки якості

На основі отриманих даних (R2) можемо сказати, що модель передбачення моделі не сильно відрізнаються від моделі, яка базується на середніх значеннях. Крім цього, можемо бачити, що в датасеті є аномальні дані MSE vs MAE, до яких модель не пристосована.

Завдання 3

Код програми:

```
from math import degrees
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures

input_file = 'data_multivar_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:]

linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

y_test_pred = linear_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",
linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n",
poly_linear_model.predict(poly_datapoint))
```

Результат порівняння двох регресій наведений на рисунку 1.3.

```
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:
 [41.4614223]
```

Рис. 1.3. Виведення характеристик лінійного регресора та порівняння передбачення лінійної й поліноміальної моделей регресії

На основі отриманого результату, можна константувати, що поліноміальний регресор справляється краще за лінійний при регресії з декількома характеристиками.

Завдання 4

Код програми:

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

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, ran
dom_state = 0)

regression = linear_model.LinearRegression()
regression.fit(X_train, y_train)
y_pred = regression.predict(X_test)

print("Linear regressor performance:")
print("regression.coef_ =", regression.coef_)
print("regression.intercept_ =", regression.intercept_)
print("R2 score =", round(r2_score(y_test, y_pred), 2))
print("Mean absolute error =", round(mean_absolute_error(y_test, y_pred), 2))
print("Mean squared error =", round(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()
```

```
Linear regressor performance:
regression.coef_ = [ -20.41129305 -265.88594023  564.64844662  325.55650029 -692.23796104
  395.62249978   23.52910434  116.37102129  843.98257585   12.71981044]
regression.intercept_ = 154.3589882135515
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```

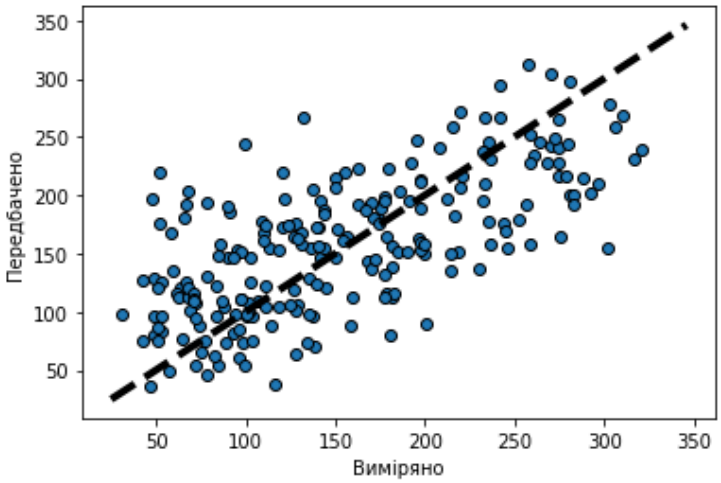


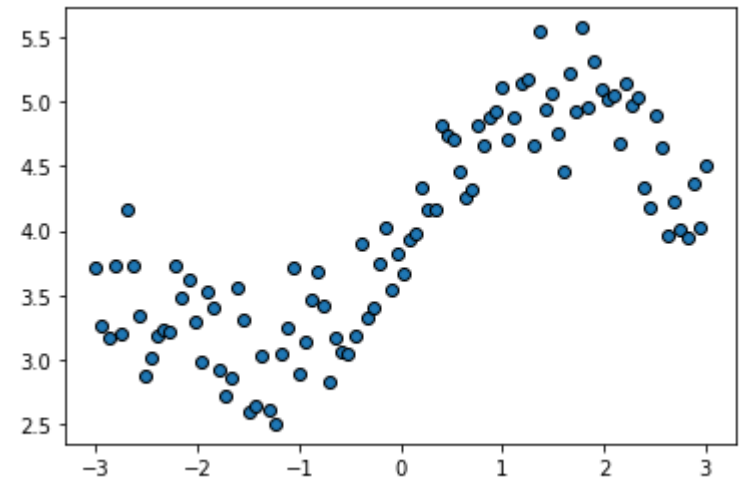
Рис. 1.4. Результати лінійої регресії

На основі отриманих результатів, можемо сказати, що дані широко розпогділені й похибка є відносно великою, але обрана регресія працює краще ніж звичайна регресія з використанням середніх значень (безуючись на R2).

Завдання 5

(Вказав номер варіанта у завданні 2)

Графіки регресії та додаткові дані наведені на рисунку 1.5.




```
-2.9393939393939394 3.2611448233257745
3.9849252547743217 [ 0.89165622 -0.00263297 -0.09869985]
```

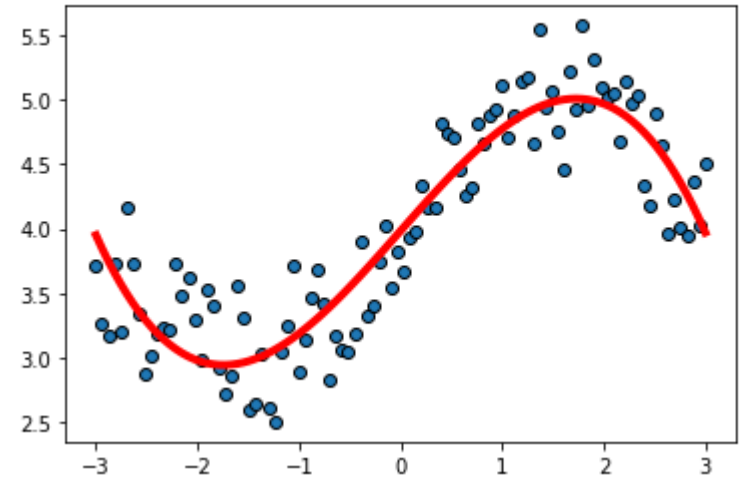


Рис. 1.5. Графіки на дані регресії

Модель варіанта: y = sin(x) + 4 + шум гаусса, отримана модель регресії з передбаченими коефіцієнтами: y = 0.88sin(x) + 3.96.

Код програми:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures

m = 100
X = np.linspace(-3, 3, m)
y = 4 + np.sin(X) + np.random.uniform(-0.6, 0.6, m)
```

```
fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors = (0, 0, 0))
plt.show()

print(X[1], y[1])

poly_features = PolynomialFeatures(degree=3, include_bias=False)
X_poly = poly_features.fit_transform(np.array(X).reshape(-1, 1))

linear_regression = linear_model.LinearRegression()
linear_regression.fit(X_poly, y)
print(linear_regression.intercept_, linear_regression.coef_)
y_pred = linear_regression.predict(X_poly)

fig, ax = plt.subplots()
ax.scatter(X, y, edgecolors = (0, 0, 0))
plt.plot(X, y_pred, color='red', linewidth=4)
plt.show()
```

Базуючись на отриманих результатах, можемо зробити висновок, що поліноміальна регресія надає змогу будувати моделі для нелінійних даних й заезпечує чудовий результат.

Завдання 6

На рисунках 1.6-1.8 наведені криві навчання для лінійної, поліноміальної 10-го ступеня та поліноміальної 3-го ступеня моделей відповідно.

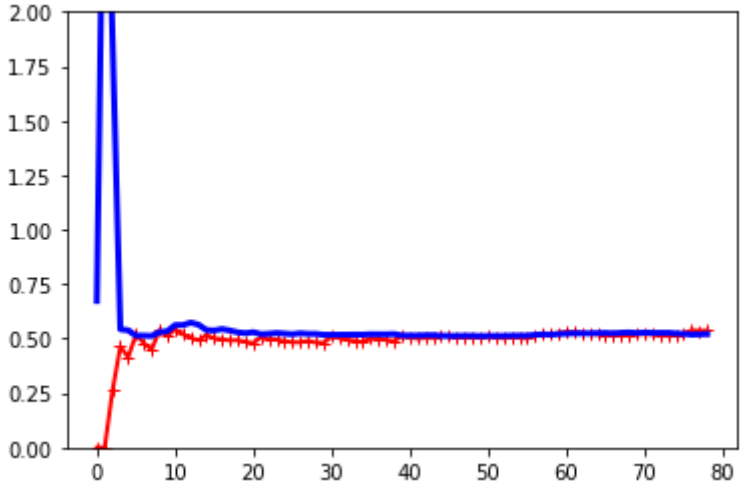


Рис. 1.6. Криві навчання для лінійної моделі

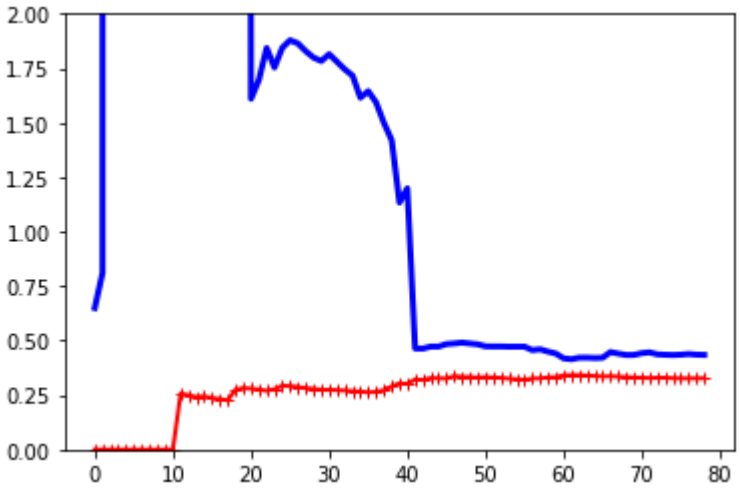


Рис. 1.7. Криві навчання для поліноміальної моделі 10-го ступеня

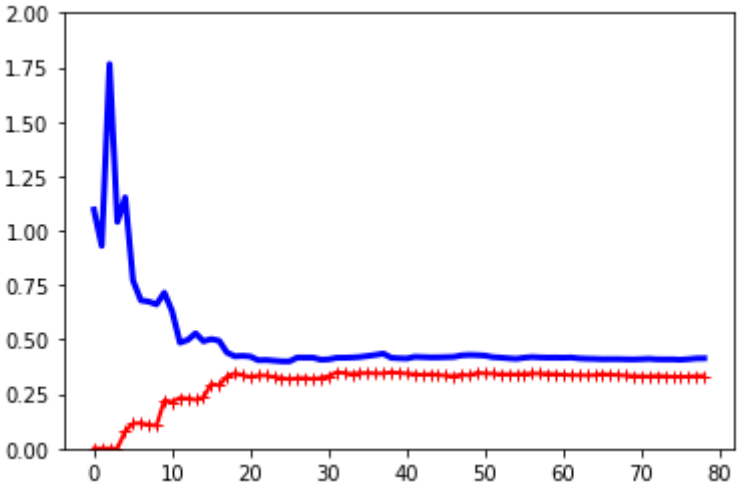


Рис. 1.8. Криві навчання для поліноміальної моделі 3-го ступеня

Код програми:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

m = 100
X = np.linspace(-3, 3, m)
y = 4 + np.sin(X) + np.random.uniform(-0.6, 0.6, 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)
  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))
  fig, ax = plt.subplots()
  plt.ylim(0, 2)
  ax.plot(np.sqrt(train_errors), "r-+", linewidth = 2, label = 'train')
  ax.plot(np.sqrt(val_errors), "b-", linewidth = 3, label = 'val')
  plt.show()

linear_regression = linear_model.LinearRegression()
plot_learning_curves(linear_regression, np.array(X).reshape(-1, 1), y)

polynomial_regression = Pipeline([
  ('poly_features', PolynomialFeatures(degree=3, include_bias=False)),
  ('lin_reg', linear_model.LinearRegression()),
])

plot_learning_curves(polynomial_regression, np.array(X).reshape(-1, 1), y)
```

Висновки: на даній лабораторній роботі я дослідив методи регресії даних, використовуючи спеціалізовані бібліотеки та мову програмування Python.