Кушер Дмитро Євгенович ЗІПЗк-22-1
Лаб 3

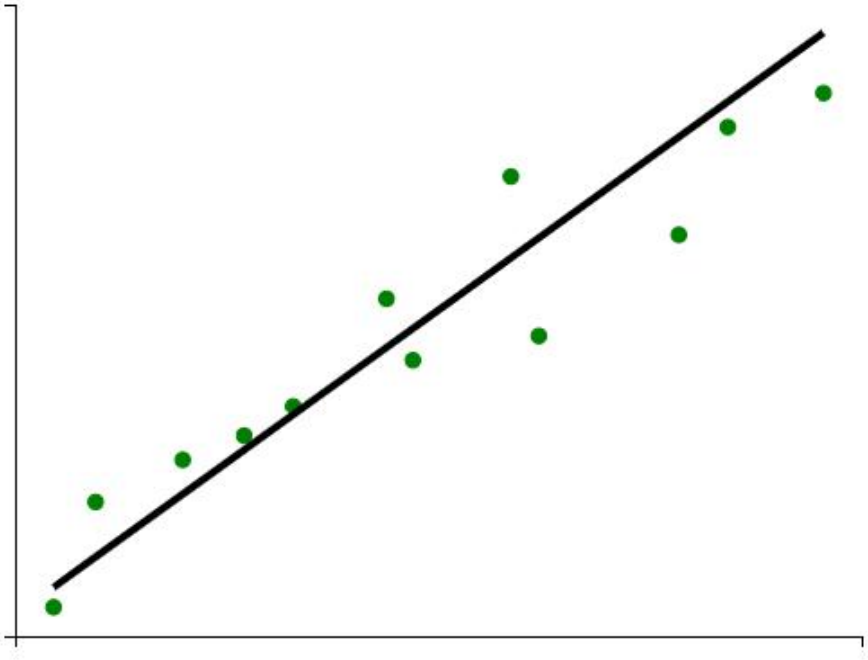
Завдання 2.1.

Створення регресора однієї змінної

```
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("Продуктивність лінійної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test, y_test_pred),
2))
print("Середня квадратична помилка =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Середня абсолютна помилка =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Пояснена оцінка дисперсії =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 оцінка =", round(sm.r2_score(y_test, y_test_pred), 2))
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("\nНова середня абсолютна помилка =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

```
Продуктивність лінійної регресії:
Середня абсолютна похибка = 0.59
Середня квадратична помилка = 0.49
Середня абсолютна помилка = 0.51
Пояснена оцінка дисперсії = 0.86
R2 оцінка = 0.86

Нова середня абсолютна помилка = 0.59
```



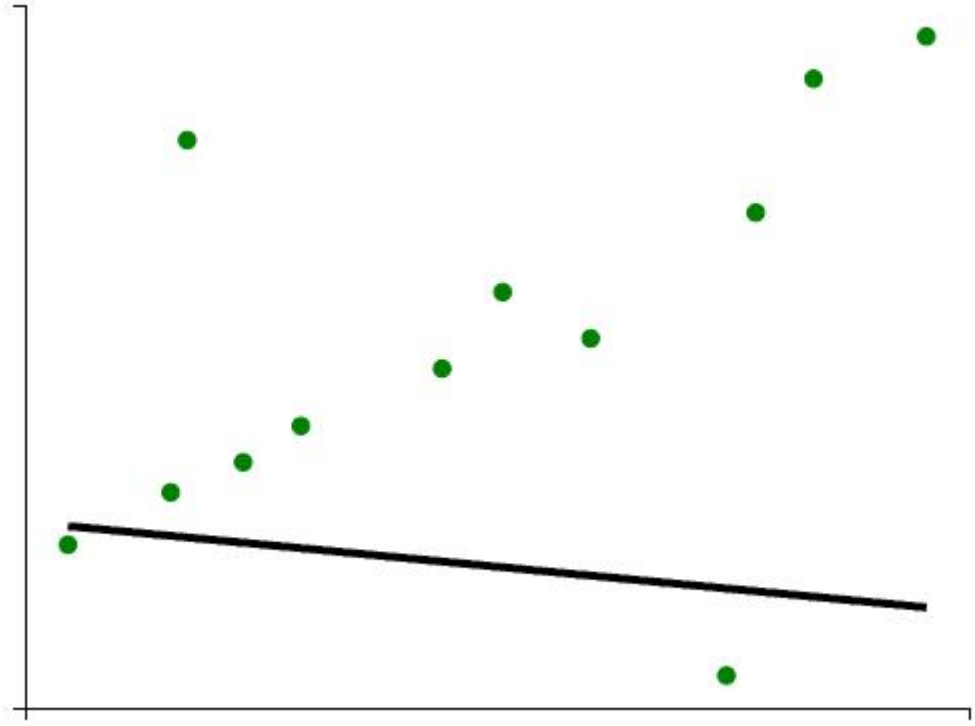
Завдання 2.2

Передбачення за допомогою регресії однієї змінної

```
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("Продуктивність лінійної регресії:")
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test, y_test_pred),
2))
print("Середня квадратична помилка =", round(sm.mean_squared_error(y_test,
y_test_pred), 2))
print("Середня абсолютна помилка =", round(sm.median_absolute_error(y_test,
y_test_pred), 2))
print("Пояснена оцінка дисперсії =", round(sm.explained_variance_score(y_test,
y_test_pred), 2))
print("R2 оцінка =", round(sm.r2_score(y_test, y_test_pred), 3))
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("\nНова середня абсолютна помилка =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

```
C:\Users\Onibi\AppData\Local\Programs\P
Продуктивність лінійної регресії:
Середня абсолютна похибка = 2.42
Середня квадратична помилка = 9.02
Середня абсолютна помилка = 2.14
Пояснена оцінка дисперсії = -0.15
R2 оцінка = -1.611

Нова середня абсолютна помилка = 2.42
```



Завдання 2.3

Створення багатовимірного регресора

```
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
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("Продуктивність лінійної регресії:")
```

```
print("Середня абсолютна похибка =", round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Середня квадратична помилка =", round(sm.mean_squared_error(y_test, y_test_pred), 2))
print("Середня абсолютна помилка =", round(sm.median_absolute_error(y_test, y_test_pred), 2))
print("Пояснена оцінка дисперсії =", round(sm.explained_variance_score(y_test, y_test_pred), 2))
print("R2 оцінка =", round(sm.r2_score(y_test, y_test_pred), 3))
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("\nНова середня абсолютна помилка =", 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))
```

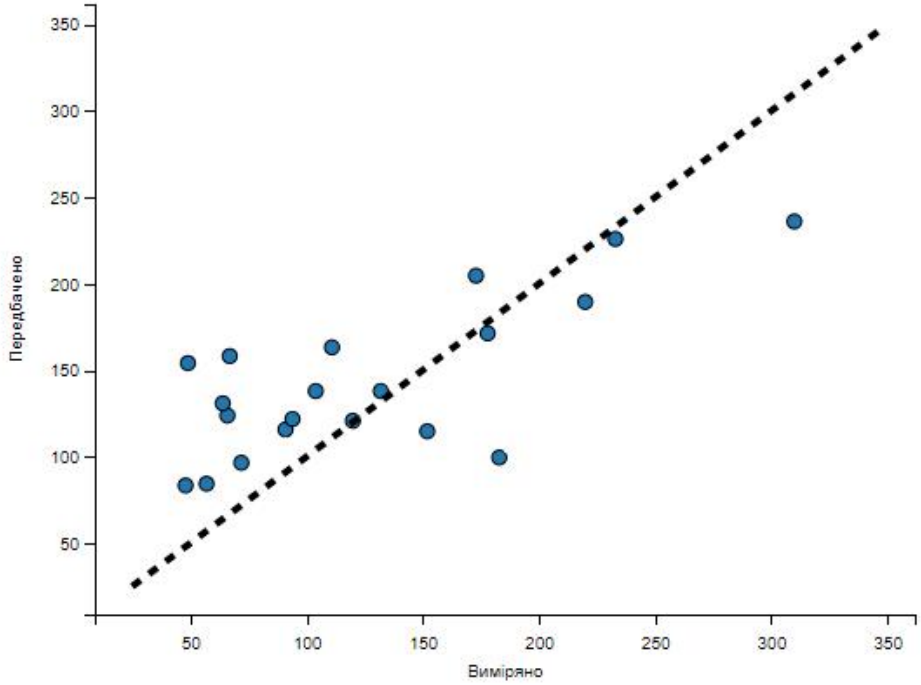
```
C:\Users\Onibi\AppData\Local\Programs\P
Продуктивність лінійної регресії:
Середня абсолютна похибка = 3.58
Середня квадратична помилка = 20.31
Середня абсолютна помилка = 2.99
Пояснена оцінка дисперсії = 0.86
R2 оцінка = 0.865

Нова середня абсолютна помилка = 3.58

Linear regression:
 [36.05286276]

Polynomial regression:
 [41.45561819]
```

Завдання 2.4

Регресія багатьох змінних

```
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
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
# Use only one feature
diabetes_X = diabetes_X[:, np.newaxis, 2]
# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)
# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)
# The coefficients
print("Regression coef: \n", regr.coef_)
print("Regression intercept: \n", regr.intercept_)
# Середня абсолютна похибка
print("Mean absolute error :", round(mean_absolute_error(diabetes_y_test, diabetes_y_pred),
2))
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("R2 score: %.2f" % r2_score(diabetes_y_test, diabetes_y_pred))
fig, ax = plt.subplots()
ax.scatter(diabetes_y_test, diabetes_y_pred, edgecolors=(0, 0, 0))
ax.plot([diabetes_y.min(), diabetes_y.max()], [diabetes_y.min(), diabetes_y.max()], 'k--',
lw=4)
ax.set_xlabel('Виміряно')
ax.set_ylabel('Передбачено')
plt.show()
```

```
C:\Users\Onibi\AppData\Local
Regression coef:
 [938.23786125]
Regression intercept:
 152.91886182616113
Mean absolute error : 41.23
Mean squared error: 2548.07
R2 score: 0.47
```



Завдання 2.5

Самостійна побудова регресії

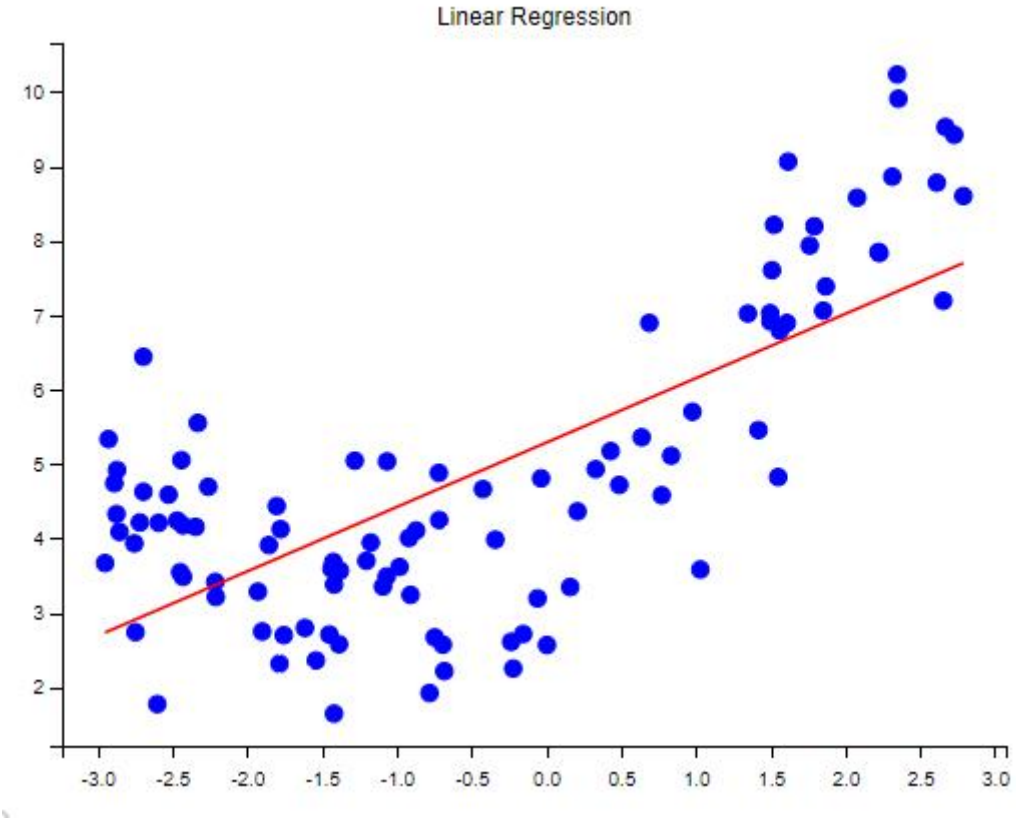
Варіант 5

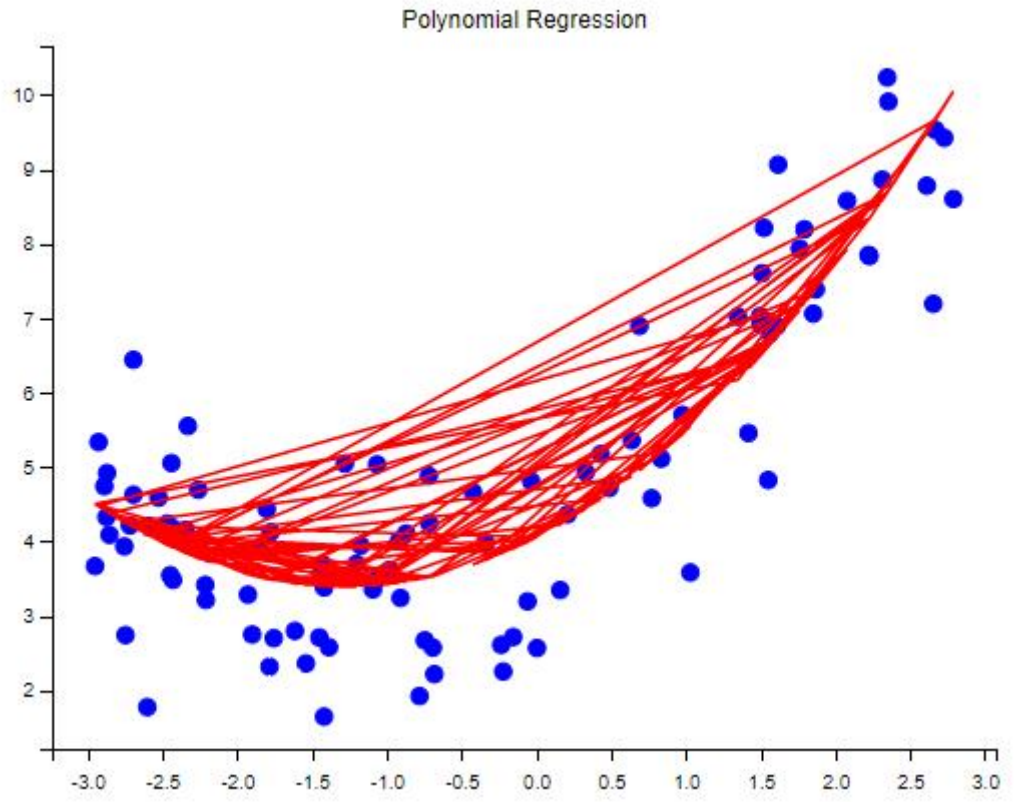
```
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)
```

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)
X = X.reshape(-1, 1)
Y = y.reshape(-1, 1)
lin = LinearRegression()
lin.fit(X, y)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
poly.fit(X_poly, y)
lin2 = LinearRegression()
lin2.fit(X_poly, y)
Y_NEW = lin2.predict(X_poly)
r2 = r2_score(Y, Y_NEW)
print('R2: ', r2)
# Visualising the Linear Regression results
```

```
plt.scatter(X, y, color='blue')
plt.plot(X, lin.predict(X), color='red')
plt.title('Linear Regression')
plt.show()
# Visualising the Polynomial Regression results
plt.scatter(X, y, color='blue')
plt.plot(X, lin2.predict(poly.fit_transform(X)), color='red')
plt.title('Polynomial Regression')
plt.show()
```

```
C:\Users\Onibi\AppData\Local\
R2:  0.7787940500359075
```



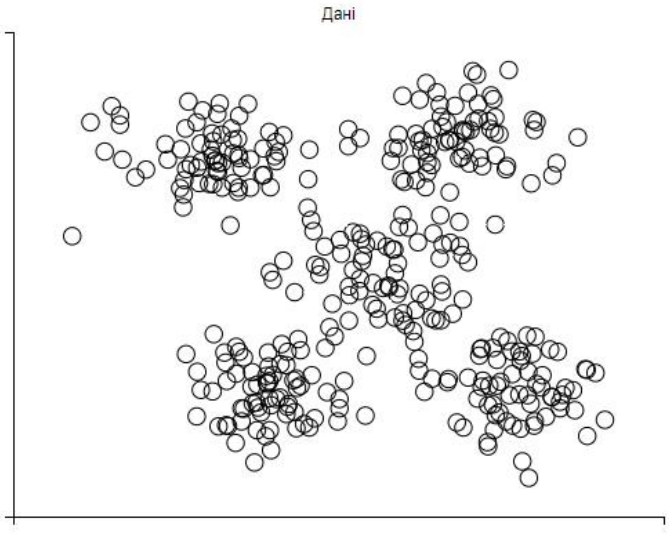


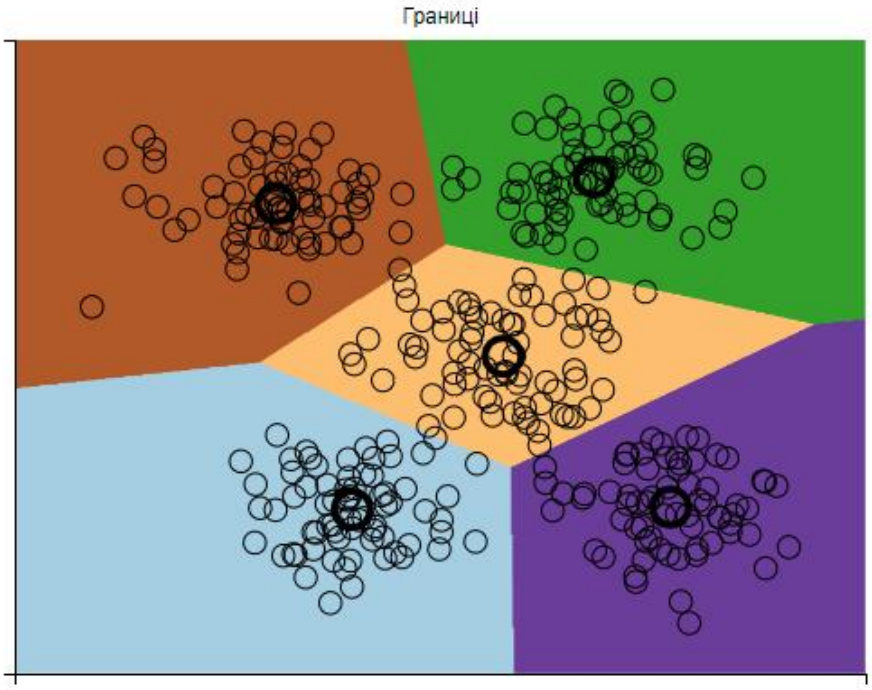
Завдання 2.7

Кластеризація даних за допомогою методу k-середніх

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics
X = np.loadtxt('data_clustering.txt', delimiter=',')
num_clusters = 5
plt.figure()
plt.scatter(X[:, 0], X[:, 1], marker='o',
            facecolors='none', edgecolors='black',  s=80)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max()+1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max()+1
plt.title('Дані')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
```

```
kmeans.fit(X)
step_size = 0.01
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max()+1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max()+1
x_vals, y_vals = np.meshgrid(
    np.arange(x_min, x_max, step_size), np.arange(y_min, y_max, step_size))
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
plt.imshow(output, interpolation='nearest', extent=(x_vals.min(), x_vals.max(
), y_vals.min(), y_vals.max()), cmap=plt.cm.Paired, aspect='auto', origin='lower')
plt.scatter(X[:, 0], X[:, 1], marker='o',
            facecolors='none', edgecolors='black',  s=80)
cluster_center = kmeans.cluster_centers_
plt.scatter(cluster_center[:, 0], cluster_center[:, 1], marker='o',
            s=210, linewidths=4, color='black', zorder=12, facecolors='none')
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max()+1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max()+1
plt.title('Границі')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```





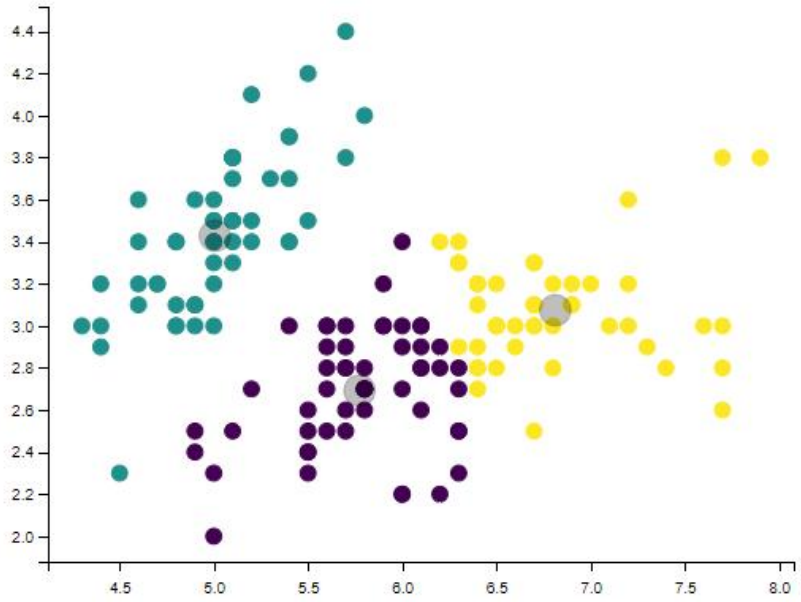
Завдання 2.8

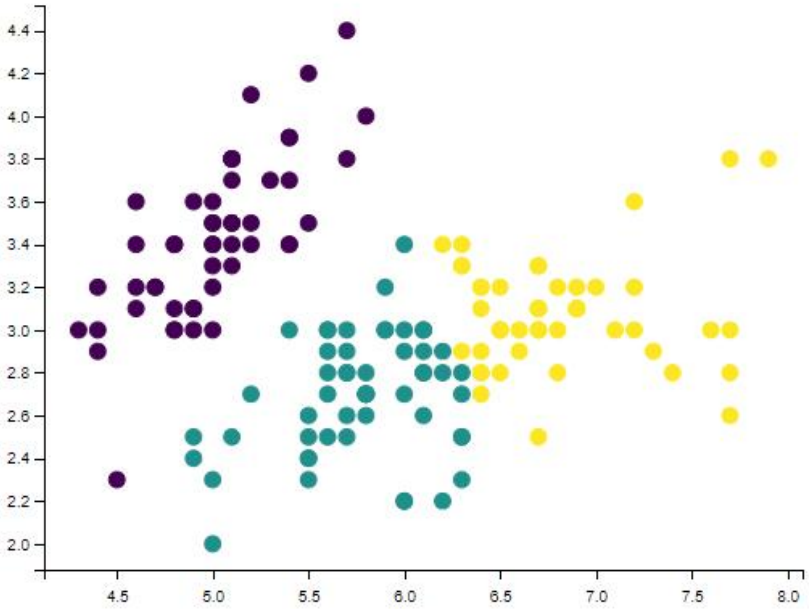
Кластеризація K-середніх для набору даних Iris

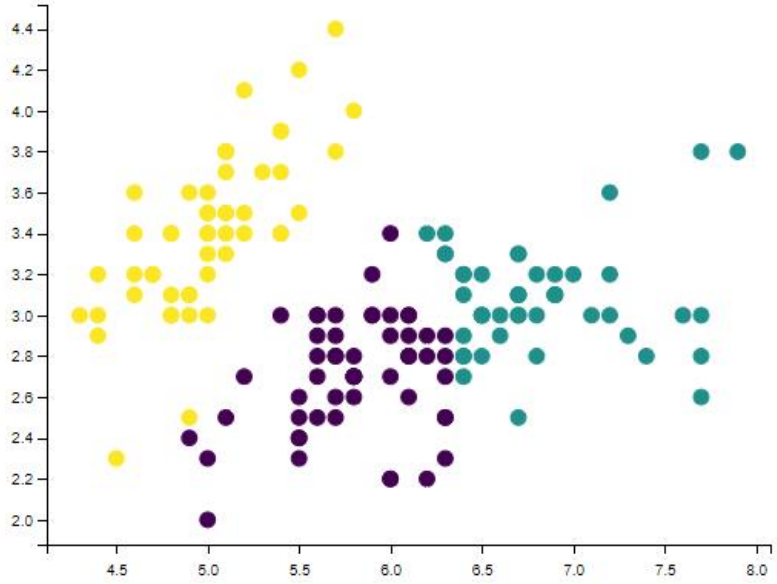
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin
import numpy as np
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target
kmeans = KMeans(n_clusters=Y.max() + 1, init='k-means++', n_init=10, max_iter=300,
                tol=0.0001, verbose=0, random_state=None, copy_x=True)
kmeans.fit(X)
y_pred = kmeans.predict(X)
print("n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, random_state:
None, copy_x: True")
print(y_pred)
plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```

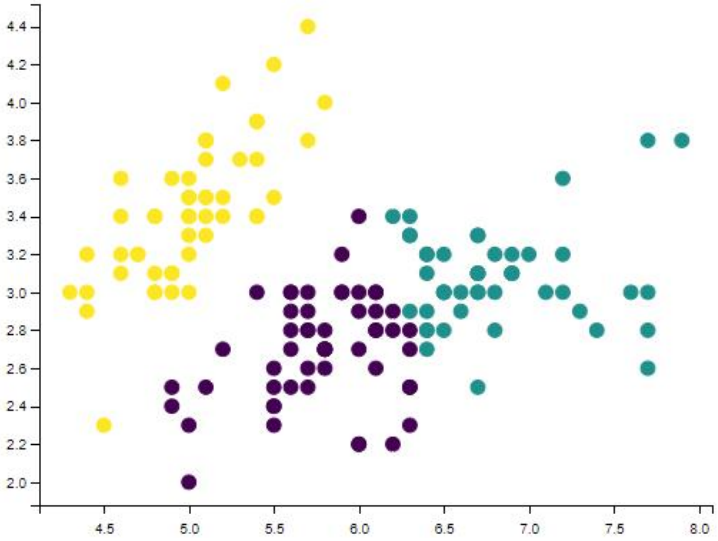
```python
def find_clusters(X, n_clusters, rseed=2):
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]
    while True:
        labels = pairwise_distances_argmin(X, centers)
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clusters)])
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels
print("using find_clusters():")
centers, labels = find_clusters(X, 3)
print("n_clusters: 3, rseed: 2")
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
centers, labels = find_clusters(X, 3, rseed=0)
print("n_clusters: 3, rseed: 0")
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
labels = KMeans(3, random_state=0).fit_predict(X)
print("n_clusters: 3, rseed: 0")
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python39\python.exe "C:/Study/4 курс/Системи штучного інтелекту/lab3/task8.py"
n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, random_state: None, copy_x: True
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1
 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0
 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 0
 0 1]
using find_clusters():
n_clusters: 3, rseed: 2
n_clusters: 3, rseed: 0
n_clusters: 3, rseed: 0

Process finished with exit code 0
```


4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
2.8
2.6
2.4
2.2
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0