Талавер Олег ПІ-60[2] Репозиторій: https://gitlab.com/bachelors2022/pi-60/talaver-oleg/other/ai.git

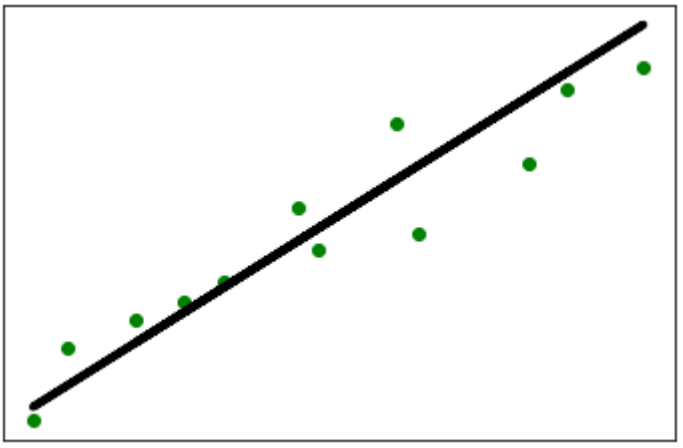
ДСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

Створення регресора однієї змінної

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

In [85]:

```
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_
```



```
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
```

Як видно з графіку, модель регресії змогла побудувала пряму що досить близько проходить до точок, хоча так як це лінійна регресія вона не змогла пройти через усі точки. Також потрібно зазначити, що збереження коду класу у файл повнітю зберігає

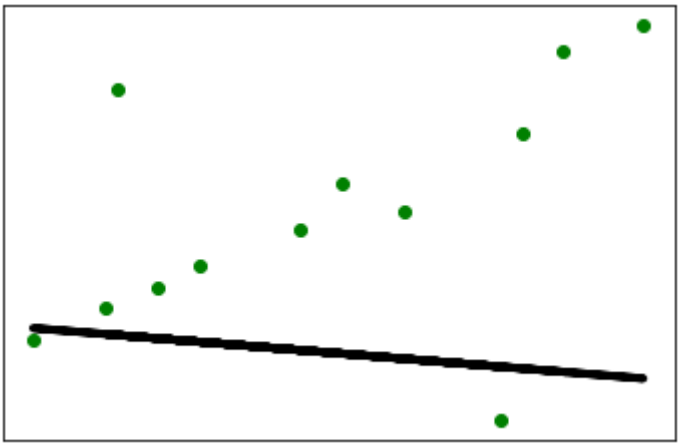
Передбачення за допомогою регресії однієї змінної

Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі (таблиця 2.1).

2 варіант (17 по списку)

In [86]:

```
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 = 'model.pkl'
```



```
Linear regressor performance:
Mean absolute error = 2.42
Mean squared error = 9.02
Median absolute error = 2.14
Explain variance score = -0.15
R2 score = -1.61
```

Регресія не пройшла успішно через присутність шуму, точок що сильно відхиляються від норми

Створення багатовимірного регресора

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

In [87]:
```
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)
print("Mean squared error =", round(sm.mean_squared_error(y_test, y_test_pred),
print("Median absolute error =", round(sm.median_absolute_error(y_test, y_test_p
print("Explain variance score =", round(sm.explained_variance_score(y_test, y_te
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

```
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
```

In [88]:
```
# 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("Linear regression: ", regressor.predict(datapoint))
print("Polynomial regression: ", poly_linear_model.predict(poly_datapoint))
```

```
Linear regression:  [36.05286276]
Polynomial regression:  [41.46504705]
```

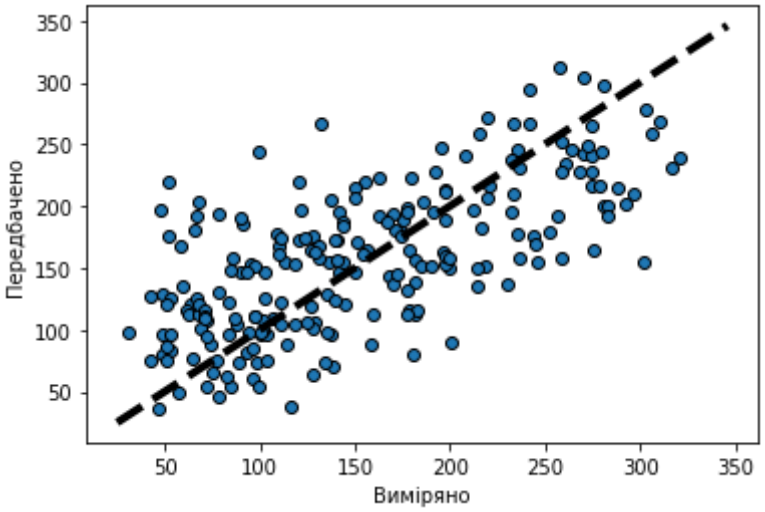
За рахунок гнучкості поліноміального регресора, модель може краще підлаштуватись під точки, що збільшує точність

Регресія багатьох змінних

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

In [89]:
```
import matplotlib.pyplot as plt
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

# Завантаження даних
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

# Поділ на навчальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_

# Створення та тренування
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

print("Regression coefficient", regr.coef_)
print("Regression intercept", regr.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()
```

```
Regression coefficient [ -20.41129305 -265.88594023  564.64844662  325.55650029
-692.23796104
  395.62249978   23.52910434  116.37102129  843.98257585   12.71981044]
Regression intercept 154.35898821355153
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
```



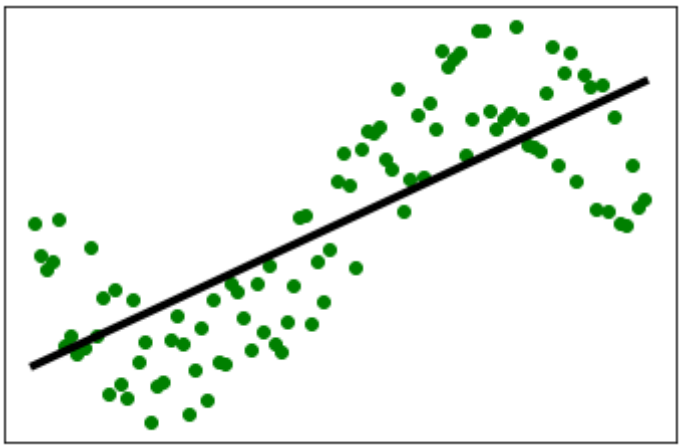
Абсолютна похибка досить велика, про це каже й графік, на якому видно значні багаточисленні відхилення від лінії співпадіння

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

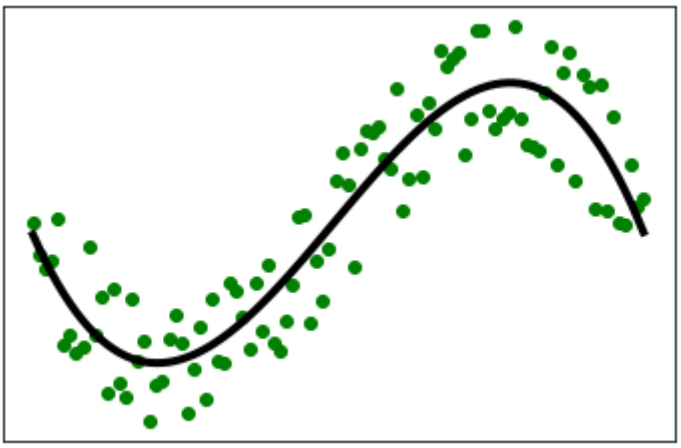
7 варіант (17 номер за списком)

In [90]:
```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Вхідні дані
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
# Очікується 2D масив
X = X.reshape((m, 1))

# Створення об'єкта лінійного регресора
linear_regression = LinearRegression()
linear_regression.fit(X, y)

# Побудова графіка лінійної регресії
plt.scatter(X, y, color='green')
plt.plot(X, linear_regression.predict(X), color='black', linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()
```



In [91]:
```
# Поліноміальна регресія (3го тупеню)
poly_features = PolynomialFeatures(degree=3, include_bias=False)
X_poly = poly_features.fit_transform(X)

poly_regression = LinearRegression()
poly_regression.fit(X_poly, y)

print("X[0]", X[0])
print("X_poly", X_poly[:5])
print("Coefficients", poly_regression.coef_)
print("Intercept", poly_regression.intercept_)

# Побудова графіка поліноміальної регресії
plt.scatter(X, y, color='green')
plt.plot(X, poly_regression.predict(poly_features.fit_transform(X)), color='blac
plt.xticks(())
plt.yticks(())
plt.show()
```

```
X[0] [-3.]
X_poly [[ -3.          9.         -27.        ]
 [ -2.93939394   8.64003673 -25.3964716 ]
 [ -2.87878788   8.28741965 -23.85772324]
 [ -2.81818182   7.94214876 -22.38241923]
 [ -2.75757576   7.60422406 -20.96922392]]
Coefficients [ 0.88714761 -0.01287936 -0.09821319]
Intercept 0.054263048417216866
```

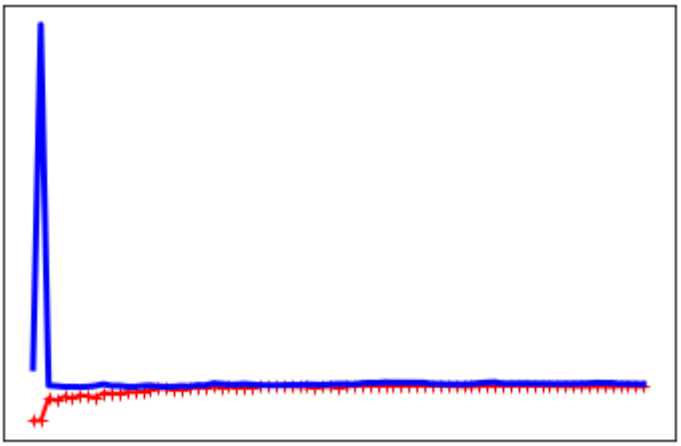


y = Sin(x) + шум Але я не впевнений як записати те що ви просите

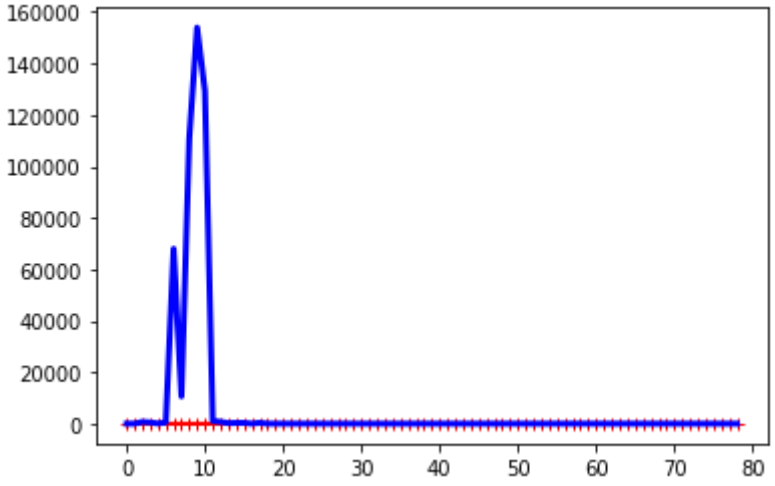
Побудова кривих навчання

Побудуйте криві навчання для ваших даних у попередньому завданні.

In [92]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures


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")

# Створення об'єкта лінійного регресора
linear_regression = LinearRegression()

# Побудова графіка лінійної регресії
plot_learning_curves(linear_regression, X, y)
plt.xticks(())
plt.yticks(())
plt.show()
```



In [93]:
```
polynomial_regression = Pipeline(
    [("poly_features", PolynomialFeatures(degree=10, include_bias=False)), ("lin
plot_learning_curves(polynomial_regression, X, y)
plt.show()
```



In [94]:
```
polynomial_regression = Pipeline(
    [("poly_features", PolynomialFeatures(degree=2, include_bias=False)), ("line
plot_learning_curves(polynomial_regression, X, y)
plt.show()
```