Для реализованной модели градиентного бустинга построить графики зависимости ошибки от количества деревьев в ансамбле и от максимальной глубины деревьев. Сделать выводы о зависимости ошибки от этих параметров.

In [28]:

```
from sklearn.tree import DecisionTreeRegressor

from sklearn import model_selection
import numpy as np
```

In [29]:

```
from sklearn.datasets import load_diabetes
```

In [30]:

```
X, y = load_diabetes(return_X_y=True)
```

In [31]:

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.25)
```

In [32]:

```
def gb_predict(X, trees_list, coef_list, eta):
    # Реализуемый алгоритм градиентного бустинга будет инициализироваться нулевыми значения
    # поэтому все деревья из списка trees_list уже являются дополнительными и при предсказа
    return np.array([sum([eta* coef * alg.predict([x])[0] for alg, coef in zip(trees_list,
```

В качестве функционала ошибки будем использовать среднеквадратичную ошибку. Реализуем соответствующую функцию.

In [33]:

```
def mean_squared_error(y_real, prediction):
    return (sum((y_real - prediction)**2)) / len(y_real)
```

Используем L_2 loss $L(y, z) = (y - z)^2$, ее производная по z примет вид $L'(y, z) = 2(z - y)$. Реализуем ее также в виде функции (коэффициент 2 можно отбросить).

In [34]:

```
def bias(y, z):
    return (y - z)
```

Реализуем функцию обучения градиентного бустинга.

In [35]:

```
def gb_fit(n_trees, max_depth, X_train, X_test, y_train, y_test, coefs, eta):

    # Деревья будем записывать в список
    trees = []

    # Будем записывать ошибки на обучающей и тестовой выборке на каждой итерации в список
    train_errors = []
    test_errors = []

    for i in range(n_trees):
        tree = DecisionTreeRegressor(max_depth=max_depth, random_state=42)

        # инициализируем бустинг начальным алгоритмом, возвращающим ноль,
        # поэтому первый алгоритм просто обучаем на выборке и добавляем в список
        if len(trees) == 0:
            # обучаем первое дерево на обучающей выборке
            tree.fit(X_train, y_train)

            train_errors.append(mean_squared_error(y_train, gb_predict(X_train, trees, coef
            test_errors.append(mean_squared_error(y_test, gb_predict(X_test, trees, coefs,
        else:
            # Получим ответы на текущей композиции
            target = gb_predict(X_train, trees, coefs, eta)

            # алгоритмы начиная со второго обучаем на сдвиг
            tree.fit(X_train, bias(y_train, target))

            train_errors.append(mean_squared_error(y_train, gb_predict(X_train, trees, coef
            test_errors.append(mean_squared_error(y_test, gb_predict(X_test, trees, coefs,

        trees.append(tree)

    return trees, train_errors, test_errors
```

In [36]:

```
def evaluate_alg(X_train, X_test, y_train, y_test, trees, coefs, eta, depth, trs):
    train_prediction = gb_predict(X_train, trees, coefs, eta)
    mse_train = mean_squared_error(y_train, train_prediction)

    print(f'Ошибка алгоритма из {trs} деревьев глубиной {depth} \
    с шагом {eta} на тренировочной выборке: {mse_train}')

    test_prediction = gb_predict(X_test, trees, coefs, eta)
    mse_test = mean_squared_error(y_test, test_prediction)

    print(f'Ошибка алгоритма из {trs} деревьев глубиной {depth} \
    с шагом {eta} на тестовой выборке: {mse_test}')

    return mse_train, mse_test
```

In [37]:

```
trees_f = []
depth_f = []
mse_ftrain = []
mse_ftest = []

n_trees = [10,15,25,50]

# Максимальная глубина деревьев
max_depth = [3,4,5]

# Шаг
eta = 1

for trs in n_trees:
    # для простоты примем коэффициенты равными 1
    coefs = [1] * trs
    for depth in max_depth:

        trees, train_errors, test_errors = gb_fit(trs, depth, X_train, X_test, y_train, y_t

        mse_train, mse_test = evaluate_alg(X_train, X_test, y_train, y_test, trees, coefs,

        print(trs, depth, mse_train, mse_test)
        trees_f.append(trs)
        depth_f.append(depth)
        mse_ftrain.append(mse_train)
        mse_ftest.append(mse_test)
```

```
Ошибка алгоритма из 10 деревьев глубиной 3     с шагом 1 на тренировочной вы
борке: 1151.3961165021497
Ошибка алгоритма из 10 деревьев глубиной 3     с шагом 1 на тестовой выборк
е: 4815.351568691609
10 3 1151.3961165021497 4815.351568691609
Ошибка алгоритма из 10 деревьев глубиной 4     с шагом 1 на тренировочной вы
борке: 628.9915274461605
Ошибка алгоритма из 10 деревьев глубиной 4     с шагом 1 на тестовой выборк
е: 6240.5379926584765
10 4 628.9915274461605 6240.5379926584765
Ошибка алгоритма из 10 деревьев глубиной 5     с шагом 1 на тренировочной вы
борке: 116.84653983401923
Ошибка алгоритма из 10 деревьев глубиной 5     с шагом 1 на тестовой выборк
е: 6637.473783558616
10 5 116.84653983401923 6637.473783558616
Ошибка алгоритма из 15 деревьев глубиной 3     с шагом 1 на тренировочной вы
борке: 740.4393449743125
Ошибка алгоритма из 15 деревьев глубиной 3     с шагом 1 на тестовой выборк
е: 5593.922532947803
15 3 740.4393449743125 5593.922532947803
Ошибка алгоритма из 15 деревьев глубиной 4     с шагом 1 на тренировочной вы
борке: 237.0582620694747
Ошибка алгоритма из 15 деревьев глубиной 4     с шагом 1 на тестовой выборк
е: 7011.667301364789
15 4 237.0582620694747 7011.667301364789
Ошибка алгоритма из 15 деревьев глубиной 5     с шагом 1 на тренировочной вы
борке: 19.612359441662804
Ошибка алгоритма из 15 деревьев глубиной 5     с шагом 1 на тестовой выборк
е: 7010.664360345182
```

```
15 5 19.612359441662804 7010.664360345182
Ошибка алгоритма из 25 деревьев глубиной 3     с шагом 1 на тренировочной вы
борке: 381.30313122307456
Ошибка алгоритма из 25 деревьев глубиной 3     с шагом 1 на тестовой выборк
е: 6767.687219427726
25 3 381.30313122307456 6767.687219427726
Ошибка алгоритма из 25 деревьев глубиной 4     с шагом 1 на тренировочной вы
борке: 61.81493933484978
Ошибка алгоритма из 25 деревьев глубиной 4     с шагом 1 на тестовой выборк
е: 7115.678159238092
25 4 61.81493933484978 7115.678159238092
Ошибка алгоритма из 25 деревьев глубиной 5     с шагом 1 на тренировочной вы
борке: 1.4598576003104813
Ошибка алгоритма из 25 деревьев глубиной 5     с шагом 1 на тестовой выборк
е: 7056.379881541467
25 5 1.4598576003104813 7056.379881541467
Ошибка алгоритма из 50 деревьев глубиной 3     с шагом 1 на тренировочной вы
борке: 64.9151385702185
Ошибка алгоритма из 50 деревьев глубиной 3     с шагом 1 на тестовой выборк
е: 7444.2097443497405
50 3 64.9151385702185 7444.2097443497405
Ошибка алгоритма из 50 деревьев глубиной 4     с шагом 1 на тренировочной вы
борке: 0.977231116211707
Ошибка алгоритма из 50 деревьев глубиной 4     с шагом 1 на тестовой выборк
е: 7353.419259681866
50 4 0.977231116211707 7353.419259681866
Ошибка алгоритма из 50 деревьев глубиной 5     с шагом 1 на тренировочной вы
борке: 0.000751524023261705
Ошибка алгоритма из 50 деревьев глубиной 5     с шагом 1 на тестовой выборк
е: 7071.189723141963
50 5 0.000751524023261705 7071.189723141963
```

Построим графики зависимости ошибки на обучающей и тестовой выборках от числа итераций.

In [38]:

```
import matplotlib.pyplot as plt
```

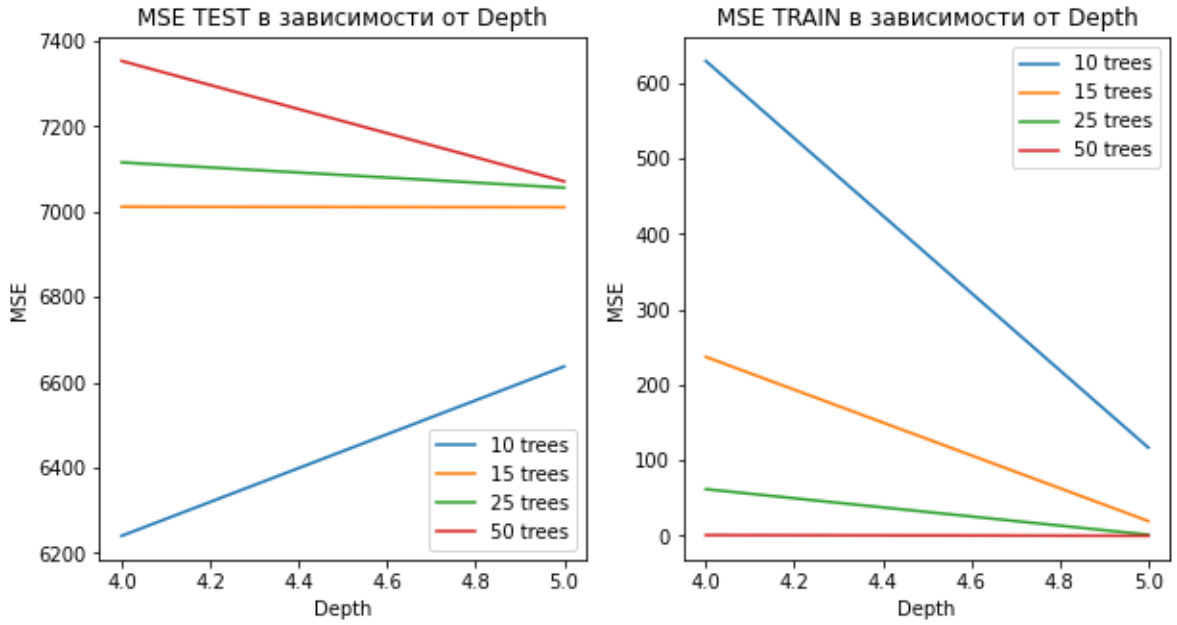
In [39]:

```
fig = plt.figure(figsize=(10, 5))
plt.subplot(121)
plt.title('MSE TEST в зависимости от Depth')
plt.xlabel(r'Depth')
plt.ylabel(r'MSE')
plt.plot(depth_f[1:3], mse_ftest[1:3], depth_f[4:6], mse_ftest[4:6], depth_f[7:9], mse_ftes
plt.legend(("10 trees", "15 trees", "25 trees", "50 trees"))

plt.subplot(122)
plt.title('MSE TRAIN в зависимости от Depth')
plt.xlabel(r'Depth')
plt.ylabel(r'MSE')
plt.plot(depth_f[1:3], mse_ftrain[1:3], depth_f[4:6], mse_ftrain[4:6], depth_f[7:9], mse_ft
plt.legend(("10 trees", "15 trees", "25 trees", "50 trees"))
```

Out[39]:

```
<matplotlib.legend.Legend at 0x1e3e1233108>
```



При параметре деревьев ниже 15, модель ведет себя разнонаправлено: увеличение глубины дает большую ошибку на тестовой выборке. При параметре деревьев выше 15-25 идет существнное падение MSE TRAIN and MSE Test при увеличении глубины. При количестве деревьев более 25, при увеличении глубины, ошибка тестовой выборки существенно снижается, train снижается не существенно.

In [ ]:

```
```