Modify the Gradient Boosting scratch code in our lecture such that:

- Notice that we are still using max_depth = 1. Attempt to tweak min_samples_split, max_depth for the regression and see whether we can achieve better mse on our boston data
- Notice that we only write scratch code for gradient boosting for regression, add some code so that it also works for binary classification. Load the breast cancer data from sklearn and see that it works.
- Further change the code so that it works for multiclass classification. Load the digits data from sklearn and see that it works
- Put everything into class

In [1]:
```
from scipy.special import expit
from sklearn.tree import DecisionTreeRegressor
from sklearn.dummy import DummyRegressor

from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.ensemble import GradientBoostingRegressor

import numpy as np
```

In [55]:
```
class GradientBoosting():
    def __init__(self, regression = True, multiclass = False):
        self.n_estimators = 200
        self.tree_params = {'max_depth': 1, 'min_samples_split': 4}
        self.models = [DecisionTreeRegressor(**self.tree_params) for _ in range(self.n_estimators)]
        self.regression = regression
        self.multiclass = multiclass

    def grad(self, y, h):
        return y - h


    def softmax(self, y_pred):
        return np.exp(y_pred) / np.sum(np.exp(y_pred))


    def fit(self, X, y):

        self.models_trained = []

        #using DummyRegressor is a good technique for starting model
        first_model = DummyRegressor(strategy='mean')
        first_model.fit(X, y)
        self.models_trained.append(first_model)

        #fit the estimators
        for i, model in enumerate(self.models):
            #predict using all the weak learners we trained up to
            #this point
            y_pred = self.predict(X, last_predict = False)

            #errors will be the total errors maded by models_trained
            residual = self.grad(y, y_pred)

            #fit the next model with residual
            model.fit(X, residual)

            self.models_trained.append(model)

    def predict(self, X, last_predict = True):
        learning_rate = 0.1  ##hard code for now
        f0 = self.models_trained[0].predict(X)  #first use the dummy model
        boosting = sum(learning_rate * model.predict(X) for model in self.models_trained[1:])
        yhat = f0 + boosting

        # Logistic multinomial / binary part
        if not self.regression:
            yhat = self.softmax(yhat)

            # If we are doing predictions on the test set, return argmax on the col of probabilities
            if last_predict:
                yhat = np.argmax(yhat, axis = 1)

        return yhat
```

In [43]:
```
from sklearn.model_selection import train_test_split

gb = GradientBoosting(regression = True)

X, y = load_boston(return_X_y=True)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3, random_state=42)



#fit the models
models = gb.fit(X_train, y_train)

#predict
y_pred = gb.predict(X_test)

#print metrics
print("Our MSE: ", mean_squared_error(y_test, y_pred))
```

```
Our MSE:  12.945557601580582
```

Binary

Import

In [52]:
```
from sklearn.datasets import load_breast_cancer
```

Train and predict

In [61]:
```
data_breast = load_breast_cancer()
X2 = data_breast.data
y2 = data_breast.target

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2,
                                                    test_size=0.3, random_state=42)
m, n = X2.shape
k = len(set(y2_train))
y_train_encoded = np.zeros((X2_train.shape[0], k))
for each_class in range(k):
    cond = y2_train == each_class
    y_train_encoded[np.where(cond), each_class] = 1

gb_logistic = GradientBoosting(regression = False)
breast_models = gb_logistic.fit(X2_train, y_train_encoded)

y_pred2 = gb_logistic.predict(X2_test)
print(y_pred2[:3])
```

```
[1 0 0]
```

Score

In [62]:
```
#print metrics
print("Our accuracy: ", accuracy_score(y2_test, y_pred2))
```

```
Our accuracy:  0.8947368421052632
```

Multinomial

Import

In [71]:
```
from sklearn.datasets import load_digits

data_digits = load_digits()
X3 = data_digits.data
y3 = data_digits.target

print(X3.shape, y3.shape)
# Note: 1797 samples, 64 classes
```

```
(1797, 64) (1797,)
```

Train and predict

In [76]:
```
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3,
                                                    test_size=0.3, random_state=42)


m, n = X3.shape
k = len(set(y3_train))
y3_train_encoded = np.zeros((X3_train.shape[0], k))
for each_class in range(k):
    cond = y3_train == each_class
    y3_train_encoded[np.where(cond), each_class] = 1

gb_logistic = GradientBoosting(regression = False)
breast_models = gb_logistic.fit(X3_train, y3_train_encoded)

y_pred3 = gb_logistic.predict(X3_test)
```

Score

In [77]:
```
#print metrics
print("Our accuracy: ", accuracy_score(y2_test, y_pred2))
```

```
Our accuracy:  0.8947368421052632
```