Practical task 9.

Ensemble models

To complete this task you're asked to implement the following functions:

```
learn_rf(X_train, y_train, X_test, y_test, num_trees):
(importance, pred_train, pred_train_ind, pred_test, pred_test_ind)
```

This function trains Random Forest model with `X_train`, `y_train` and calculates its predictions on train and test samples of the whole model and individual predictors (each tree in random forest).

The arguments:

1. `X_train` - numpy array of shape $N \times D$, storing train objects' features in rows.
2. `y_train` - numpy array of shape N, storing train objects' dependent feature values.
3. `X_test` - numpy array of shape $M \times D$, storing test objects' features in rows.
4. `y_test` - numpy array of shape M, storing test objects' dependent feature values.
5. `num_trees` - number or trees in random forest.

The return values:

1. `importance` - numpy array of shape D, storing feature importances.
2. `pred_train` - numpy array of shape N, storing model predictions on train sample.
3. `pred_train_ind` - numpy array of shape $N \times$ `num_trees`, storing predictions of each tree on train sample.
4. `pred_test` - numpy array of shape M, storing model predictions on test sample.
5. `pred_test_ind` - numpy array of shape $M \times$ `num_trees`, storing predictions of each tree on test sample.

Useful functions and classes: `RandomForestRegressor.estimators_`, `RandomForestRegressor.feature_importances_`

```
learn_gbt(X_train, y_train, X_test, y_test, num_trees, learning_rate):
(importance, pred_train, pred_train_staged, pred_test, pred_test_staged)
```

This function trains Gradient Boosting model with `X_train`, `y_train` and calculates final and staged predictions on train and test samples.

The arguments (additionally to the arguments of `learn_rf`):

1. `learning_rate` - the learning rate of each tree in GBT.

The return values:

1. `importance` - numpy array of shape D, storing feature importances.
2. `pred_train` - numpy array of shape N, storing model predictions on train sample.
3. `pred_train_staged` - numpy array of shape $N \times$ `num_trees`, storing staged predictions on train sample (after adding a tree).
4. `pred_test` - numpy array of shape M, storing model predictions on test sample.
5. `pred_test_staged` - numpy array of shape $M \times$ `num_trees`, storing staged predictions on test sample.

Useful functions and classes: `GradientBoostingRegressor.staged_predict()`

`squared_error(y_true, y_hat):`
`(error)`

This function calculates squared error of predictions.

The arguments :

1. `y_true` - numpy array of shape N(or M), storing ground-truth values of the dependent feature.
2. `y_hat` - numpy array of shape N(or M), storing predictions of the dependent feature.

The return values:

1. `error` - numpy array of shape N(or M), storing squared errors of predictions.

`fine_tuning(y_train, pred_train_ind, y_test, pred_train_ind):`
`(pred_train, pred_test)`

This function fine-tune base models' weights in the ensemble via Linear Regression model.

The arguments :

1. `y_train` - numpy array of shape N, storing train objects' dependent feature values.
2. `pred_train_ind` - numpy array of shape $N \times$ `num_trees`, storing predictions of each tree on train sample.
3. `y_test` - numpy array of shape M, storing test objects' dependent feature values.
4. `pred_test_ind` - numpy array of shape $M \times$ `num_trees`, storing predictions of each tree on test sample.

The return values:

1. `pred_train` - numpy array of shape N, storing fine-tuned model predictions on train sample.
2. `pred_test` - numpy array of shape M, storing fine-tuned model predictions on test sample.

The task:

1. Choose arbitrary `random_state` and use it in train-test splitting and model training procedures.
2. Load the dataset from `california.dat`. The dependent feature is **MedianHouseValue**. Split the dataset into train/test sets in proportion $80/20$ respectively.
3. Use `learn_rf` function to train Random Forest Regression models with 10, 20, 30, 50, 100 and 200 trees
   (a) Indicate 3 most important features (at any number of trees)
   (b) For every random forest model compare train- and test- errors' variance of individual trees and ensemble model. Explain your observations.
4. Use `learn_gbt` function to train Gradient Boosting Tree Regression model with 1000 trees and learning rate equal to 0.2.
   (a) Indicate 3 most important features. Why some importance values are significantly different from Random Forest case, despite that both are based on tree models?
   (b) Plot average train- and test- errors on each boosting iteration. Notice, that at some iteration test error changes negligibly and begins to increase, while train error keeps decreasing. Explain this effect.
5. Use `fine_tuning` function to fine-tune weights of RF and GBT models.
   (a) Compare fine-tuned predictions and predictions of Random Forest ensemble of 10, 50, 100, 250 trees. Analyse error's mean and variance, describe and explain your observations.
   (b) Compare fine-tuned predictions and predictions of Gradient Boosting Tree Regression with 100, 500, 1000 iterations. Analyse error's mean and variance, describe and explain your observations.