Function and Parameters

1. **retrieveEnsemble.ensemble**(layer_name, models, train_x, train_y, test_x, n_classes, n_folds, metric_for_classifier='log_loss', metric_for_regressor= 'mean_squared_error')

Parameter	***Description***	***datatype***
layer_name	name of the layer	string
models	list of models	list
train_x	variable part of train data	pandas.dataframe
train_y	target of train data	pandas.series
test_x	variable part of test data	pandas.dataframe
n_classes	number of classes in classification problem; in regression problem, set it to 1	int
n_folds	number of folds that train dataset will be split to generate new input for next layer	int
metric_for_classifier	metric for classifiers in list of models, default='log_loss'	string
metric_for_regressor	metric for regressors in list of models, default=' mean_squared_error '	string

return:

new_blend_train	train dataset for next layer	pandas.dataframe
new_blend_test	test dataset for next layer,	pandas.dataframe
models_info	list of models information for this layer	list

*Parameter metric can be chosen from below:

‘accuracy_score', 'auc', 'f1_score', 'log_loss', 'roc_auc_score', 'mean_absolute_error', 'mean_squared_error', 'median_absolute_error' , which can get information from http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

2. **retrieveEnsemble.retrieve(**all_infoList_list, metric='log_loss', maximize= False)

Parameter	***Description***	***datatype***
all_infoList_list	list of lists, each list in it contain models' information of the layer	list
metric	the metric for chosing the same type and best models	list
maximize	whether to maximize the metric, if False, will minimize the matric to choose best models	boolean

return:

good	list of output layer models combined with models from previous layers but better than worst model in output layer	list

3. **opt_weight**(modelInfo_list,train_y,metric='log_loss',method='SLSQP',maximize=False)

Parameter	***Description***	***datatype***
modelInfo_list	list of models that wanted to be ensembled	list
train_y	the target in train dataset	pandas.series
metric	the metric used to find optimized weights	string
method	the method used to find optimized weights	string
maximize	whether to maximize the metric	bolean

return:

best_score	list of models to ensemble	list
weights	the optimized weights for ensemble	numpy.array

*Parameter method can be selected from below:

- 'Nelder-Mead' *(see here)*
- 'Powell' *(see here)*
- 'CG' *(see here)*
- 'BFGS' *(see here)*
- 'Newton-CG' *(see here)*
- 'L-BFGS-B' *(see here)*
- 'TNC' *(see here)*
- 'COBYLA' *(see here)*
- 'SLSQP' *(see here)*
- 'dogleg' *(see here)*
- 'trust-ncg' *(see here)*

4. **predict**(modelInfo_list,weights,majority_voting=False)

Parameter	***Description***	***datatype***
modelInfo_list	list of model that used to ensemble prediction	list
weights	weights to ensemble predictions	numpy.array
majority_voting	whether to use majority voting to ensemble prediction, if False, will use weighted average to ensemble predictions	boolean

return:

pred	final prediction	numpy.array

5. **predict_proba**(modelInfo_list,weights)

Parameter	***Description***	***datatype***
modelInfo_list	list of model that used to ensemble prediction	list
weights	weights to ensemble predictions	numpy.array

return:

pred	probability of each class in classification problem	numpy.array