ShaoXing 绍兴

Package for Post-tuning and Model Evaluation

24/06/2023

Background

The purpose of this package is to streamline the model analysis project and provide a wholesome framework for post-model tuning actions.

Functionalities of ShaoXing include building models based on the best hyperparameters, making future predictions, calculating more detailed statistics than the tuning outputs of i.e. JiXi and YangZhou, as well as cross validation statistics. It is also capable of producing residual plots for regressions and confusion matrices for classifications.

It can also help visualise feature importance for a range of different classes of models

Class

Class	Purpose
`ShaoXing`	Object that stores full, train, val, test and potentially future data, as well as feature column names, label name, model class, best hyperparameters to perform analysis and prediction based on the best model Able to get: -extensive model train val test evaluation statistics -cross validated evaluation statistics -residual plots/evaluation matrices -feature importance data and plot

Class: ZhongShan

Methods:

Methods	Purpose
`ShaoXing()`	
`read_in_features_label(self, features, label)`	Reads in values feature column name list and label Parameters: features – list label - str
`read_in_full_train_test_data(full_data, train_data, val_data, test_data)`	Reads in Full, Training, Validated and Test data along with features list, auto transforming to X, y Parameters: full_data – DataFrame train_data – DataFrame val_data – DataFrame test_data - DataFrame
`read_in_future_data(future_data)`	Read in Future data (X only – for predictions) Parameters: future_data - DataFrame
`read_in_untrained_model(model_class, best_params, type, model_class_type, model_name)`	Reads in underlying model object for tuning, as well as what type of model it is and its name (arbitrary)

	Parameters: model_class – class object Best_params – dictionary of str:int (or str) type – str – either ‘Classification’ or ‘Regression’ model_class_type - str– either ‘Sklearn Tree’ or ‘LGB’ or ‘OLS LR’ or ‘Sklearn linear kernel’ model_name - str
`read_in_fitted_model(fitted_model, type, model_class_type, model_name)`	Reads in a fitted model Parameters: fitted_model – model object type - str – either ‘Classification’ or ‘Regression’ model_class_type - str– either ‘Sklearn Tree’ or ‘LGB’ or ‘OLS LR’ or ‘Sklearn linear kernel’ model_ name - str
`fit_model()`	Fit the model based on train x and trian y for model class with previously inputted hyperparameters
`export_model(model_export_address)`	Export the model as a pickle Parameters:

	model_export_address – str – does not need to include '.pickle'
`predict_using_future_data(self, return_pred = False)`	Make predictions for future x data, with option for function to output result (instead of just storing internal dataframe) Parameters: return_pred – bool – optional, default False
`export_future_data_and_predictions(future_data_and_pred_saving_address)`	Export future predictions (joined up to future x) Parameters: Future_data_and_pred_saving_address – str – does not need to include '.csv'
`view_future_data_and_predictions(return_df = False)`	View the Dataframe of future X and future labels, with option for function to output result (instead of just just displaying) Parameters: return_df – bool – optional, default False
`get_analysis(address = None)`	Get all model statistics, with option to save the first two dataframes. For clf_type == 'Regression' -r2, RMSE, 4 quantile r2, 4 quantile RMSE, 10 quantile r2, 10 quantile rmse (Train and Val and Test) -Cross validated aforementioned stats (Train and Test) -how well does X predict residuals? (r2, corr and pvals) -how well does each xi predict residuals (r2, corr and pvals)

	-Residual plots For clf_type == 'Classification' -accuracy, balanced accuracy, precision, recall, f1 (last three weighted, macro, micro) (Train and Val and Test) -Cross validated aforementieond stats (Train and Test) -Confusion Matrix Saving: will save first two dataframe as '*address*_regular.csv' and '*address*_CV.csv' Parameters: address – str – optional, does not need to include '.csv'
`export_analysis_dfs(address)`	Export the analysis data Parameters: address – str – does not need to include '.csv'
`export_residual_plot(feature, data_type, address)`	Saves the specified residual plot as a png Parameters: feature – str data_type – str – either 'Train' or 'Val' or 'Test' address – str – does not need to include '.png'
`display_model_analysis_stats()`	Displays the model analysis statistics and relevant plots (if exists)

`get_feature_importance(address = None)`	Calculates and displays feature importance bar graphs, with option to save the bar graph. Parameters: address- str – optional, does not need to include ‘.png’

Objects:

Objects	Purpose
full_data	DataFrame
train_data	DataFrame
val_data	DataFrame
test_data	DataFrame
features	list
label	str
train_x	DataFrame
train_y	DataFrame
val_x	DataFrame
val_y	DataFrame
test_x	DataFrame
test_y	DataFrame
future_data	DataFrame
train_analyse_df	DataFrame
val_analyse_df	DataFrame
test_analyse_df	DataFrame
model	Model object
future_pred	np.array
future_data_and_pred	DataFrame
regular_stats_df	DataFrame

`CV_stats_df`	DataFramae
`RESIDUAL_PLOT_OBJECTS`	Dict – str:dict of str:matplotlib.figures
`feature_importance_df`	DataFrame

Dependencies

```
pandas

numpy

matplotlib
```