Package 'causalKNN'

February 20, 2023

Type Package

Title Causal KNN Treatment Effect Estimation

Version 0.3

Description The causalKNN package provides causal KNN regression and treatment effect projection algorithms following Hitsch, Misra, and Zhang (2023).

License GPL-3

Depends R (>= 3.1.0)

Imports Rcpp (>= 0.12.16),
data.table,
glmnet,
KernelKnn,
stats

Suggests parallel,
knitr,
rmarkdown,
testthat

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/walterwzhang/causalKNN-dev

BugReports https://github.com/walterwzhang/causalKNN-dev/issues

R topics documented:

`causalKNN_direct` *Performs direct Causal KNN regression*

Description

Wraps the direct Causal KNN regression into one function. The function sequentially runs `knn_index_mat`, `knn_optimal_k`, and `causalknn_treatment_effect` functions and returns a list of treatment results and optimal parameters. Since all of the important parameters are returned, the user can re-run certain functions easily.

Usage

```
causalKNN_direct(
  DF,
  W,
  Y,
  key,
  DF_test,
  key_test,
  threads = 1L,
  knn_index_mat_options = list(k = floor(sqrt(nrow(DF)))),
  knn_optimal_k_options = list(N_step = ifelse(knn_index_mat_options$k > 300L,
   floor(knn_index_mat_options$k/25), 14L), K_step = ifelse(knn_index_mat_options$k >
    300L, 25L, floor(knn_index_mat_options$k/14))),
  causalknn_treatment_effect_options = list(),
  verbose = FALSE
)
```

Arguments

`DF`	A data frame of the features of the training sample (data.frame)
`W`	A vector of the treatment indicator of the training sample (1/0 coded) (integer)
`Y`	A vector of the outcome values of the training sample (numeric)
`key`	A vector of the indices of the training sample (integer)
`DF_test`	A data frame of the features in the test set (data.frame)
`key_test`	A vector of the indices of the test sample (integer)
`threads`	The value of number of threads to use (integer)
`knn_index_mat_options`	A list of parameters to pass into `knn_index_mat` (list)
`knn_optimal_k_options`	A list of parameters to pass into `knn_optimal_k` (list)
`causalknn_treatment_effect_options`	A list of parameters to pass into `causalknn_treatment_effect` (list)
`verbose`	A boolean flag for verbose output (logical)

Details

This off-the-shelf implementation does not implement bootstrapping for finding the optimal K value and for the treatment effect projection.

Value

A list containing function results (list)

`causalKNN_TEP` *Performs Causal KNN Regression and Treatment Effect Projection*

Description

Wraps the causal KNN Regression and the Treatment Effect Projection into one function. The treatment effect projection step is performed with a Elastic-Net Sequentially runs `knn_index_mat`, `knn_optimal_k`, `causalknn_treatment_effect`, and `tep_projection_e_net` functions and returns a list of treatment results and optimal parameters. Since all of the important parameters are returned, the user can re-run certain functions easily. (i.e. re-running `causalknn_treatment_effect` with a larger K value).

Usage

```
causalKNN_TEP(
  DF,
  W,
  Y,
  key,
  DF_test,
  key_test,
  threads = 1L,
  knn_index_mat_options = list(k = floor(sqrt(nrow(DF)))),
  knn_optimal_k_options = list(N_step = ifelse(knn_index_mat_options$k > 300L,
   floor(knn_index_mat_options$k/25), 14L), K_step = ifelse(knn_index_mat_options$k >
     300L, 25L, floor(knn_index_mat_options$k/14))),
  tep_e_net_options = list(),
  verbose = FALSE
)
```

Arguments

`DF`	A data frame of the features (data.frame)
`W`	A vector of the treatment indicator (1/0 coded) (integer)
`Y`	A vector of the outcome values (numeric)
`key`	A vector of the indices (integer)
`DF_test`	A data frame of the features in the test set (data.frame)

`key_test` A vector of the indices of the test sample (integer)

`threads` The value of number of threads to use (integer)

`knn_index_mat_options`
A list of parameters to pass into `knn_index_mat` (list)

`knn_optimal_k_options`
A list of parameters to pass into `knn_optimal_k` (list)

`tep_e_net_options`
A list of parameters to pass into `tep_projection_e_net` (list)

`verbose` A boolean flag for verbose output (logical)

Details

This off-the-shelf implementation does not implement bootstrapping for finding the optimal K value and for the treatment effect projection.

Value

A list containing function results (list)

`causalknn_treatment_effect`
Treatment effect estimation with causal KNN for K values.

Description

In separate `K` value case, supply a vector of two elements to the `K` parameter. The first element of vector will be `K` for untreated nearest neighbors, and the second elements will be the K for treated nearest neighbors. Supply a single value for K otherwise.

Usage

```
causalknn_treatment_effect(
  W,
  Y,
  key,
  K,
  knn_index_list,
  key_test = NULL,
  return_outcomes = FALSE,
  bootstrap_keys = NULL,
  bootstrap_keys_test = NULL
)
```

Arguments

W	A vector of the treatment indicator (1/0 coded) in the training set (integer)
Y	A vector of the outcome values in the training set (numeric)
key	A vector of the indices of the training set (integer)
K	Value(s) for K (integer)
knn_index_list	A list of the KNN index matrices provided by the knn_index_mat function (list)
key_test	A vector of the indices of the test set (Defaults to NULL) (integer)
return_outcomes	Whether to return predicted potential (treated/untreated) outcomes (logical)
bootstrap_keys	A vector of the bootstrapped keys for the training set (default to NULL for no bootstrap case) (integer)
bootstrap_keys_test	A vector of the bootstrapped keys for the test set (default to NULL for no bootstrap case) (integer)

Details

Supply a key_test if looking to estimate treatment effects for the validation set. Otherwise, the default value of NULL will instruct the function to estimate treatment effects for the training set.

Allows for bootstrapped case - see the bootstrap vignette.

Value

A data.frame of the predicted treatment effect, treatment outcome, and untreated outcome by key (data.frame)

---

knn_index_mat *Compute the KNN index matrix*

Description

Find the nearest neighbor indices for the treated and untreated for a supplied data.frame Uses the knn.index.dist function from the KernelKnn package Currently only applicable to a binary treatment coded as (1/0) In the returned matrices, the rows are indexed by NN, and the columns are the DF observations

Usage

```
knn_index_mat(
  DF,
  W,
  k = floor(sqrt(nrow(DF))),
  DF_test = NULL,
  distance_metric = "euclidean",
```

```
  standardize = TRUE,
  keep_dist = FALSE,
  threads = 1L,
  cov_DF = NULL
)
```

Arguments

DF	A data frame of the features in the training sample (data.frame)
W	A vector of the treatment indicator (1/0 coded) in the training sample (integer)
k	A integer for the max k value for the of kNN. Defaults to floor(sqrt(nrow(DF))) (integer)
DF_test	A data frame of the features in the test sample (data.frame)
distance_metric	String supplying the method (character)
standardize	A boolean flag for whether to standardize the features (logical)
keep_dist	A boolean flag for keeping generated distance matrices (logical)
threads	A integer determining the number of threads used - uses openMP (integer)
cov_DF	A custom supplied covariance matrix for the mahalanobis distance metric (data.frame)

Value

A list containing the index (and distance) matrices (list)

---

knn_optimal_k *Find the optimal K for causal KNN regression*

Description

Finds the optimal K by choosing a grid of possible K values and finding the K values that has the smallest RMSE to the Transformed Outcome. We can also Bootstrap the procedure to find a more robust optimal K value.

Usage

```
knn_optimal_k(
  DF,
  W,
  Y,
  key,
  N_step,
  K_step,
  knn_index_list,
  propensity = mean(W),
  bootstrap_keys = NULL
)
```

Arguments

DF	A data frame of the features (data.frame)
W	A vector of the treatment indicator (1/0 coded) (integer)
Y	A vector of the outcome values (numeric)
key	A vector of the indices (integer)
N_step	The number of steps to take (integer)
K_step	The value between steps of K and the initial K value (integer)
knn_index_list	A list of the KNN index matrices provided by the knn_index_mat function (list)
propensity	A vector of the propensity scores (defaults to RCT setting) (numeric)
bootstrap_keys	A vector of the bootstrapped keys (default to NULL for no bootstrap case) (integer)

Value

A list containing the optimal K value (list)

---

knn_optimal_k_separate

Find the optimal K separately for treated/untreated for causal KNN regression

---

Description

Finds the optimal K separately for treated and untreated nearest neighbors by choosing a grid of possible K values and finding the K values that has the smallest RMSE to the Transformed Outcome. The combination of the optimal K value for the treated and the untreated that yields the smallest transformed error loss will be returned.

Usage

```
knn_optimal_k_separate(
  DF,
  W,
  Y,
  key,
  N_step,
  K_step,
  knn_index_list,
  propensity = mean(W),
  bootstrap_keys = NULL,
  threads = 1L
)
```

Arguments

DF	A data frame of the features (data.frame)
W	A vector of the treatment indicator (1/0 coded) (integer)
Y	A vector of the outcome values (numeric)
key	A vector of the indices (integer)
N_step	The number of steps to take (integer)
K_step	The value between steps of K and the initial K value (integer)
knn_index_list	A list of the KNN index matrices provided by the knn_index_mat function (list)
propensity	A vector of the propensity scores (defaults to RCT setting) (numeric)
bootstrap_keys	A vector of the bootstrapped keys (default to NULL for no bootstrap case) (integer)
threads	The value of number of threads to use (integer)

Details

We can also bootstrap the procedure to find a more robust optimal K value.

Value

A list containing the optimal K values (list)

---

tep_enet *Performs the TEP step for the Elastic-Net family of models*

Description

We search over a grid of values for the elastic-net to find the optimal alpha parameter. Set alpha to NULL to find the optimal alpha value. The Elastic-Net family of regressions is estimated with 'glmnet' package. Parallelization is done with the 'parallel' package.

Usage

```
tep_enet(
  training_X,
  training_TE,
  test_X,
  alpha = 1,
  alpha_seq = seq(0, 1, by = 0.01),
  threads = 1L,
  cv_folds = 10L,
  est_seed = 1234L,
  ...
)
```

Arguments

`training_X`	A data frame of the training set features (data.frame)
`training_TE`	A vector of the individually estimated treatment effects (numeric)
`test_X`	A data frame of the test set features (data.frame)
`alpha`	The alpha value for the elastic net (numeric)
`alpha_seq`	The a vector alpha value to consider (numeric)
`threads`	The value of number of threads to use (integer)
`cv_folds`	The number of folds for the cross-validation step (integer)
`est_seed`	A value for CV estimation seed (integer)
`...`	Additional parameters to be passed to `glmnet::cv.glmnet`

Value

A vector of the TEP projection values for the test set (numeric)

Index