Supplementary Material:
Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features

1 Implementations

Feature encoding techniques are implemented in several general preprocessing packages. We provide an overview of existing implementations for R in Table 1. For python, an extension to scikit-learn (Pedregosa et al., 2011), *category encoders* (McGinnis et al., 2018) is available. To enable a fair and reliable comparison in our study, we implemented all methods outlined above on top of the `mlrCPO` package. All code can be found in our **online repository** (https://github.com/compstat-lmu/paper_2021_categorical_feature_encodings).

encoding method	regularization	package

indicator	–	`stats`
		`embed`
		`mlrCPO`
		`mlr3pipelines`

hash	–	`FeatureHashing` `embed`

impact	smoothing	`embed`
		`mlrCPO`
		`mlr3pipelines`

impact	cross-validation	`vtreat` `mlr3pipelines (via vtreat)`
regularized impact (glmm)	–	`embed` `mlr3pipelines`

Table 1: Overview over existing encoding implementations in R. Implementations can deviate from the algorithms described in our manuscript by minor implementation details. Indicator encoding encloses a variety of methods (one-hot, dummy, helmert encoding etc.)

2 Pseudocode for all encoding strategies

Algorithm 1 to 10 contain the pseudocode for all encoding strategies studied in our manuscript in order to improve reproducibility and to provide further hints towards subtle differences in encoders and implementations.

Algorithm 1 Integer Encoding

Training:

compute random permutation $\boldsymbol{int} = (int_1, \dots, int_k, \dots, int_L)^T$ of $(1, \dots, L)^T$

for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\hat{x}_i^{train} = int_k$ with $x_i^{train} = l_k$, $k = 1, \dots, L$

Prediction:

for x^{new} **do**

  if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}^{new} = int_k$ with $x^{new} = l_k$, $k = 1, \dots, L$ **else** $\hat{x}^{new} = NA$

Algorithm 2 Frequency Encoding

Training:
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\hat{x}_i^{train} = \frac{N_l}{N}$ with $x_i^{train} = l$

Prediction:
for x^{new} **do**
  if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}^{new} = \frac{N_l}{N}$ with $x^{new} = l$ **else** $\hat{x}^{new} = 1$

Algorithm 3 One-Hot Encoding

Training:
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**
  for all $l \in \mathcal{L}^{train}$ **do** $\hat{x}_{il}^{train} = I(x_i^{train} = l)$

Prediction:
for x^{new} **do**
  for all $l \in \mathcal{L}^{train}$ **do**
    if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}_l^{new} = I(x^{new} = l)$ **else** $\hat{x}_l^{new} = 0$

Algorithm 4 Dummy Encoding

Training:
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**
  for all $l \in \mathcal{L}^{train} \setminus l_{ref}$ **do** $\hat{x}_{il}^{train} = I(x_i^{train} = l)$

Prediction:
for x^{new} **do**
  for all $l \in \mathcal{L}^{train} \setminus l_{ref}$ **do**
    if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}_l^{new} = I(x^{new} = l)$ **else** $\hat{x}_l^{new} = NA$

Algorithm 5 Hash Encoding

Training: require $hash.size \in \mathbb{N}$
for all $l \in \mathcal{L}^{train}$ **do** $ind_l = (hash(l) \mod hash.size) + 1$, $ind_l \in \mathbb{N}$, $hash(l) \in \mathbb{N}$
1. define matrix $\boldsymbol{D}^{N \times hash.size}$ with $d_{ih} = 1$ if $ind_l = h$ and $d_{ih} = 0$ if $ind_l \neq h$, $x_i^{train} = l$
2. $\boldsymbol{D} \to \tilde{\boldsymbol{D}}^{N \times V}$ with $V \leq hash.size$, by dropping constant columns in $\boldsymbol{D}$
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\hat{x}_{iv}^{train} = \tilde{d}_{iv}$

Prediction:
for x^{new} **do**
  $ind^{new} = (hash(x^{new}) \mod hash.size) + 1$
  $\boldsymbol{d}^{new}$ of length $hash.size$ with $d_h^{new} = 1$ if $ind^{new} = h$ and $d_h^{new} = 1$ if $ind^{new} \neq h$
  $\boldsymbol{d}^{new} \to \tilde{\boldsymbol{d}}^{new}$ of length V, by dropping columns which were constant in $\boldsymbol{D}$
  for all V columns in $\tilde{\boldsymbol{D}}$ **do** $\hat{x}_v^{new} = \tilde{d}_v^{new}$

Algorithm 6 Leaf Encoding

Training: require number of cross-validation folds $K \in \mathbb{N}$
fit CART tree on $\mathcal{D}^{train}$ with complexity pruning based on K-fold cross-validation
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\tilde{x}_i = t$ with x_i^{train} in terminal node t

Prediction:
for x^{new} **do**
  if $x^{new} \in \mathcal{L}^{train}$ **then** $\tilde{x}^{new} = t$ with x^{new} in terminal node t
  else $\tilde{x}^{new} = b$ where b indicates the biggest terminal node

Algorithm 7 Impact Encoding Regression

Training: require smoothing parameter $\epsilon \in \mathbb{R}$

for all $l \in \mathcal{L}^{train}$ **do** $\delta_l = \frac{\sum_{i:x_i^{train}=l} y_i^{train} + \epsilon \cdot \bar{y}^{train}}{N_l + \epsilon} - \bar{y}^{train}$ with $\bar{y}^{train} = \frac{\sum_{i=1}^{N} y_i^{train}}{N}$

for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\hat{x}_i^{train} = \delta_l$ with $x_i^{train} = l$

Prediction:

for x^{new} **do**

 if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}^{new} = \delta_l$ with $x^{new} = l$ **else** $\hat{x}^{new} = 0$

Algorithm 8 Impact Encoding Classification

Training: require smoothing parameter $\epsilon \in \mathbb{R}$

for all $c \in \mathcal{C}$ **do**

 $p_c = \frac{N_c}{N}$, $p_c^{new} = \frac{N_c + \epsilon}{N + 2\epsilon}$, $logit_c = \log(\frac{p_c}{1-p_c})$, $logit_c^{new} = \log(\frac{p_c^{new}}{1-p_c^{new}})$

 $\delta_c^{new} = logit_c^{new} - logit_c$

 for all $l \in \mathcal{L}^{train}$ **do**

  $p_{lc} = \frac{\sum_{i:x_i^{train}=l} I(y_i^{train}=c) + \epsilon}{N_l + 2\epsilon}$, $logit_{lc} = \log(\frac{p_{lc}}{1-p_{lc}})$

  $\delta_{lc} = logit_{lc} - logit_c$

for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\hat{x}_{ic}^{train} = \delta_{lc}$ with $x_i^{train} = l$

Prediction:

for x^{new} **do**

 for all c in $\mathcal{C}$ **do**

  if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}_c^{new} = \delta_{lc}$ with $x^{new} = l$ **else** $\hat{x}_c^{new} = \delta_c^{new}$

Algorithm 9 GLMM Encoding Regression

Training: require $n.folds \in \mathbb{N}$

fit simple random intercept model: $y_i^{train} = \beta_{0l} + \epsilon_i = \gamma_{oo} + u_l + \epsilon_i$ on $\mathcal{D}^{train}$

with $u_l \overset{iid}{\sim} N(0, \tau^2)$, $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$ and $x_i^{train} = l$, $l \in \mathcal{L}^{train}$

if $n.folds = 1$ **then**

 for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

  $\hat{x}_i^{train} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x_i^{train} = l$

else use $n.folds$ cross-validation scheme to make training sets $\mathcal{D}_1^{train}, \ldots, \mathcal{D}_{n.folds}^{train}$

 and fit simple random intercept model on each $\mathcal{D}_m^{train}$

 for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

  $\hat{x}_i^{train} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x_i^{train} = l$ based on the model m

  with $(x_i^{train}, y_i^{train}) \notin \mathcal{D}_m^{train}$

Prediction:

for x^{new} **do**

 if $x^{new} \in \mathcal{L}^{train}$ **then**

  $\hat{x}^{new} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x^{new} = l$ based on full model fitted on $\mathcal{D}^{train}$

 else $\hat{x}^{new} = \hat{\gamma}_{00}$ based on full model fitted on $\mathcal{D}^{train}$

Algorithm 10 GLMM Encoding Binary Classification

Training: require $n.folds \in \mathbb{N}$

fit simple glmm: $E(y_i^{train}) = h(\eta_i) = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}$, $\eta_i = \beta_{0l} = \gamma_{00} + u_l$ on $\mathcal{D}^{train}$

with $u_l \overset{iid}{\sim} N(0, \sigma^2)$, $y_i^{train} \overset{ind}{\sim} Be(h(\eta_i))$ and $x_i^{train} = l$, $l \in \mathcal{L}^{train}$

if $n.folds = 1$ **then**

  for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

    $\hat{x}_i^{train} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x_i^{train} = l$

else use $n.folds$ cross-validation scheme to make training sets $\mathcal{D}_1^{train}, \dots, \mathcal{D}_{n.folds}^{train}$

  and fit simple glmm on each $\mathcal{D}_m^{train}$

  for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

    $\hat{x}_i^{train} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x_i^{train} = l$ based on the model m

    with $(x_i^{train}, y_i^{train}) \notin \mathcal{D}_m^{train}$

Prediction:

for x^{new} **do**

  if $x^{new} \in \mathcal{L}^{train}$ **then**

    $\hat{x}^{new} = \hat{\beta}_{0l}^{\mathcal{D}^{train}}$ with $x^{new} = l$ based on full model fitted on $\mathcal{D}^{train}$

  else $\hat{x}^{new} = \hat{\gamma}_{00}$ based on full model fitted on $\mathcal{D}^{train}$

Algorithm 11 GLMM Encoding Multiclass Classification

Training: require $n.folds \in \mathbb{N}$

define response matrix $\boldsymbol{Y}^{N \times C}$ with $y_{ic} = 1$ if $y_i^{train} = c$ and $y_{ic} = 0$ if $y_i^{train} \neq c$:

$\tilde{\mathcal{D}}^{train} = \{(x_i^{train}, y_{i1}^{train}, \dots, y_{iC}^{train}), \dots, (x_N^{train}, y_{N1}^{train}, \dots, y_{NC}^{train})\}$

for all C classes **do**

  fit simple glmm: $E(y_{ic}^{train}) = h(\eta_i) = \frac{\exp(\eta_i)}{1+\exp(\eta_i)}$, $\eta_i = \beta_{0l} = \gamma_{00} + u_l$ on $\boldsymbol{y}_c^{train}$ from $\tilde{\mathcal{D}}^{train}$

  with $u_l \overset{iid}{\sim} N(0, \sigma^2)$, $y_{ic}^{train} \overset{ind}{\sim} Be(h(\eta_i))$ and $x_i^{train} = l$, $l \in \mathcal{L}^{train}$

if $n.folds = 1$ **then**

  for all C models **do**

    for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

      $\hat{x}_{ic}^{train} = \hat{\beta}_{0l}^{\tilde{\mathcal{D}}^{train}}$ with $x_i^{train} = l$

else use $n.folds$ cross-validation scheme to make training sets $\tilde{\mathcal{D}}_1^{train}, \dots, \tilde{\mathcal{D}}_{n.folds}^{train}$

  for all C classes **do**

    fit simple glmm on $\boldsymbol{y}_c^{train}$ from each $\tilde{\mathcal{D}}_m^{train}$

    for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**

      $\hat{x}_{ic}^{train} = \hat{\beta}_{0l}^{\tilde{\mathcal{D}}^{train}}$ with $x_i^{train} = l$ based on the model m for class c

      with $(x_i^{train}, y_i^{train}) \notin \tilde{\mathcal{D}}_m^{train}$

Prediction:

for x^{new} **do**

  for all C class models **do**

    if $x^{new} \in \mathcal{L}^{train}$ **then**

      $\hat{x}_c^{new} = \hat{\beta}_{0l}^{\tilde{\mathcal{D}}^{train}}$ with $x^{new} = l$ based on full model for class c

    else $\hat{x}_c^{new} = \hat{\gamma}_{00}$ based on full model for class c

Note on GLMM Encoders: To speed up the computation of the GLMM encoders, we followed the performance tips from the lme4 vignette (https://cran.r-project.org/web/packages/lme4/vignettes/lmerperf.html): we did not compute derivations (`calc.derivs = FALSE`) and used the `NLOPT_LN_BOBYQA` optimizer from the nloptr package with liberal stopping criteria (`maxeval = 1000`, `xtol_abs = 1e-6`, `ftol_abs = 1e-6`).

Table 2: Win Percentages of One-hot over Dummy Encoding

Learner	Binary Class	Multi Class	Regression
LASSO	80	94	75
RF	83	71	71
GB	63	71	54
KNN	67	59	50
SVM	68	82	52

Note. Percentage of encoding conditions in which one-hot performs better than dummy per task setting.

3 Comparison of One-Hot and Dummy Encoding

A frequently asked question is, whether one-hot encoding or dummy encoding is to be preferred. While dummy encoding clearly is commonly preferred for non-regularized linear models, the answer is less clear for general machine learning algorithms studied in our setting.

From the results shown in Table 2 we can observe, that for most algorithms and datasets, one-hot encoding is to be preferred over dummy-encoding.

4 Additional encoders not mentioned in the manuscript

In addition to the encoders mentioned in the manuscript, the benchmark results available in our online repository (https://github.com/compstat-lmu/paper_2021_categorical_feature_encodings) contain two additional experimental encoders that were newly developed. We do not mention them in our manuscript because they did not perform well, and in hindsight, we are not satisfied with their design. We briefly mention them here for transparency.

Algorithm 12 Cluster Encoding

Training: require number of desired levels $V \in \mathbb{N}$
if task is regression **then**
  for all $l \in \mathcal{L}^{train}$ **do** $\bar{y}_l^{train} = \frac{\sum_{i:x_i^{train}=l} y_i^{train}}{N_l}$
    define $\boldsymbol{v}_l = (\bar{y}_l^{train}, \delta_l)^T$ with $\delta_l = \left| N_l - \frac{\sum_{k=1}^{L} N_k}{L} \right|$
if task is classification **then**
  for all $l \in \mathcal{L}^{train}$ **do**
    for all $c \in \mathcal{C}$ **do** $s_{lc} = \sum_{i:x_i^{train}=l} I(y_i^{train} = c)$
    define $\boldsymbol{v}_l = (s_{l1}, \ldots, s_{lC}, \delta_l)^T$ with $\delta_l = \left| N_l - \frac{\sum_{k=1}^{L} N_k}{L} \right|$

1. compute distance matrix $\boldsymbol{D}^{L \times L}$ with $d_{jk} = ||\boldsymbol{v}_j - \boldsymbol{v}_k||$
2. fit hierarchical cluster analysis on $\boldsymbol{D}$
3. prune dendrogram to obtain V combined levels

for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do** $\tilde{x}_i^{train} = v$ with x_i^{train} in leaf v
Prediction:
for x^{new} **do**
  if $x^{new} \in \mathcal{L}^{train}$ **then** $\tilde{x}^{new} = v$ with x^{new} in leaf v **else** $\tilde{x}^{new} = NA$

Algorithm 13 RF Encoding Regression and Binary Classification

Training: require number of trees $B \in \mathbb{N}$
fit random forest with trees $T_1, ..., T_B$
for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**
  if x_i^{train} inbag for all B trees **then** $\hat{x}_i^{train} = \frac{1}{B} \sum_{b=1}^{B} T_b(x_i^{train})$
  else $\hat{x}_i^{train} = \frac{1}{B_i^{OOB}} \sum_{b:\ x_i^{train}\ OOB\ T_b} T_b(x_i^{train})$
Prediction:
for x^{new} **do**
  if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}^{new} = \frac{1}{B} \sum_{b=1}^{B} T_b(x^{new})$
  else $\hat{x}^{new} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{L} \sum_{l \in \mathcal{L}^{train}} T_b(l)$

Algorithm 14 RF Encoding Multiclass Classification

Training: require number of trees $B \in \mathbb{N}$
fit random forest with trees $T_1, ..., T_B$
for all C classes **do**
  for all $x_i^{train} \in \boldsymbol{x}^{train}$ **do**
    if x_i^{train} inbag for all B trees **then** $\hat{x}_{ic}^{train} = \frac{1}{B} \sum_{b=1}^{B} T_b^c(x_i^{train})$
    else $\hat{x}_{ic}^{train} = \frac{1}{B_i^{OOB}} \sum_{b:\ x_i^{train}\ OOB\ T_b} T_b^c(x_i^{train})$
Prediction:
for x^{new} **do**
  for all C classes **do**
    if $x^{new} \in \mathcal{L}^{train}$ **then** $\hat{x}_c^{new} = \frac{1}{B} \sum_{b=1}^{B} T_b^c(x^{new})$
    else $\hat{x}_c^{new} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{L} \sum_{l \in \mathcal{L}^{train}} T_b^c(l)$

4.1 Additional datasets not mentioned in the manuscript

In addition to the datasets mentioned in the manuscript, the benchmark results available in our online repository (https://github.com/compstat-lmu/paper_2021_categorical_feature_encodings) contain three additional datasets: *KDD98* (OmlId: 41435), *sf-police-incidents* (OmlId: 41436), *Traffic_violations* (OmlId: 41443). A substantive amount of conditions failed for those rather big datasets due to memory problems, which is why we remove them from the results discussed in our manuscript. We briefly mention them here to provide full transparency.

References

McGinnis, W., hbghhy, Tao, W., andrethrill, Siu, C., Davison, C., Bollweg, N., 2018. scikit-learn-contrib/categorical-encoding: Release for zenodo. Zenodo. doi:10.5281/zenodo.1157110

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., others, 2011. Scikit-learn: Machine learning in python. Journal of machine learning research 12, 2825–2830.