CRISPRclassify Model Retraining Guide

This purpose of this document is to outline the process to retrain and update the CRISPRclassify tool with an outside dataset.

Building the feature dataset

A feature dataset must first be developed using the structural features derived from the repeat sequence as well as the extracted kmer tokens (or ngrams). Structural features include the proportion of GC pairs within the repeat string: *gc*, the palindromic index of the repeat: *palIdx*, and the character length of the repeat: *length*. PalIdx is calculated the fraction of matching nucleotides between the repeat sequence and its reverse complement at the same index position. The training dataset should be in matrix format with the unique repeat sequences stored as rows and the feature set stored as columns. An example format of the global training file (subsequently referred to as the "*training_df*" in following code) is below:

cas_type	gc	palIdx	length	ACGCT	ACGGG	AGCCG	AGCGA	…
I-A	3.44827	0.24137	29	1	0	0	0	…
I-C	10.3448	0.44827	29	0	0	0	1	…
III-F	16.6666	0.41666	24	0	0	0	0	…
II-A	17.2413	0.31034	29	0	0	1	0	…

CRISPRclassify uses a stratified modeling approach and therefore a model specific to each subtype type must be trained. The subtype label must be transformed into a binary indicator for each model stratum. The code snippet below illustrates the stratum-specific transformation:

```
library(tidyverse)
library(Matrix) #Builds sparse matrix
library(xgboost) #Used for xgboost training

cas_type <- "I-C"
n_char_cnt <- 5 #The kmer length used by the model. This should match the kmer length of the training features.

cas_training_df <- training_df %>%
mutate(response = ifelse(type == cas_type, 1, 0)) %>%
select(-cas_type)
```

The resulting matrix will take the following form below. We will refer to this dataframe as the "*cas_training_df*" in the following code:

response	gc	palIdx	length	ACGCT	ACGGG	AGCCG	AGCGA	…
0	3.44827	0.24137	29	1	0	0	0	…
1	10.3448	0.44827	29	0	0	0	1	…
0	16.6666	0.41666	24	0	0	0	0	…

0	17.2413	0.31034	29	0	0	1	0	…

Training Model Strata

Formatting training data for xgboost

The xgboost package requires that the feature dataset be represented as a matrix and that the response labels be represented as vector. The matrix is further transformed into a sparse matrix using the Matrix package. Example code is below:

```
train_feat_sparse_mat <- as(as.matrix(cas_training_df), "dgCMatrix")
train_response_vect <- cas_training_df$response

class(train_feat_sparse_mat)[1] #should be of type dgCMatrix
class(train_response_vect) #should be numeric
```

Training

The xgboost function supports various hyperparameters. The tuning of these hyperparameters will be specific to the feature dataset used to train the model.

```
bstSparse <- xgboost(data = train_feat_sparse_mat,
                     label = train_response_vect,
                     max.depth = 10,
                     eval_metric = "auc",
                     eta = .3,
                     nthread = 12,
                     early_stopping_rounds = 10,
                     nrounds = 50,
                     objective = "binary:logistic")


model_name <- str_c(cas_type,"_",Sys.Date(),"_nchar_", n_char_cnt)
xgb.save(bstSparse, model_name)
```

The xgboost model object for each subtype stratum should be saved using the following naming convention:

[CAS TYPE]_[YYYY-MM-DD]_nchar_[KMER SIZE]

Example: I-C_2020-12-04_nchar_5

Model Metadata

CRISPRclassify additionally requires metadata files related to feature importance and the format of the feature matrix used by the model strata.

The **feature importance file** can be derived using the following code excerpt:

```
importance_matrix <- xgb.importance(model = bstSparse) %>%
tibble() %>%
bind_cols(type = cas_type)
```

A truncated example of the file format is provided below:

Feature	Gain	Cover	Frequency	type
GCGAC	0.38705	0.09926	0.01099	I-C
length	0.16757	0.07117	0.12333	I-C
gc	0.06777	0.0234	0.1060	I-C
ATCCA	0.05128	0.03689	0.01806	I-C
GTGGA	0.03718	0.02572	0.01178	I-C
…	…	…	…	…

The feature importance file should be saved using the following naming convention:

xg_feature_importance_[KMER SIZE]_[YYYY-MM-DD].csv

Example: xg_feature_importance_5_2020-12-04.csv

The **feature to stratum mapping file** should contain the feature to subtype mapping. For example, the features used in this I-C example model stratum can be extracted as follows:

```
tibble(cas_type = cas_type,
       nchar = n_char_cnt,
       feature_name = colnames(all_data_feat_df))
```

The file should be saved using the following naming convention and should include all feature to subtype stratum mappings. Note that the KMER length is also contained in the file as "nchar".

matrix_format_[YYYY-MM-DD]_nchar_[KMER SIZE].csv

Example: matrix_format_2020-12-04_nchar_5.csv

A truncated example of the feature to subtype stratum mapping file is below:

cas_type	nchar	feature_name
I-C	5	gc
I-C	5	palIdx
I-C	5	length
I-C	5	ACGCT
I-C	5	ACGGG
I-C	5	AGCCG
I-C	5	AGCGA
…	…	…

Updating the CRISPRclassify Project

The feature importance file and feature to stratum files should both be stored in the csvRef directory of the CRISPRclassify project and referenced in the globalVars.R file located in the R directory of the project. If additional subtypes are included in the newly-trained models, the **getCasVect()** function should be updated to include the superset of subtypes supported by the tool.

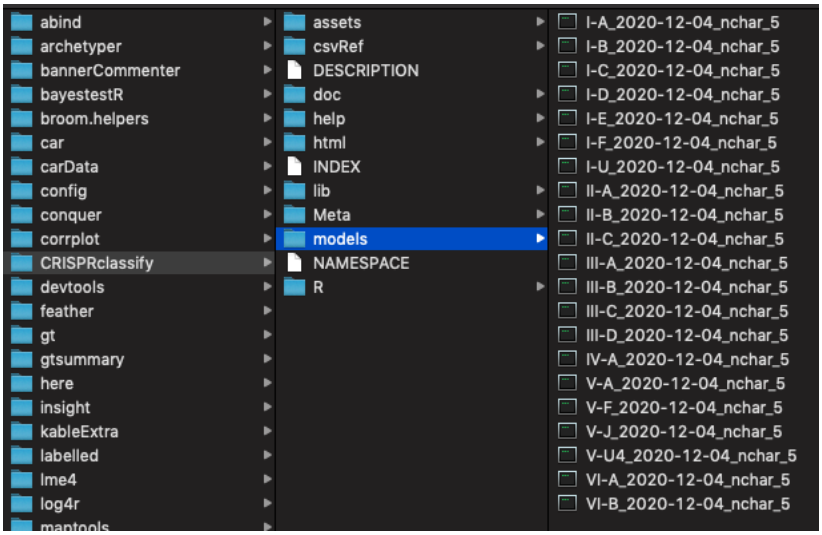
globalVars.R:

```
getMatrixFormatFile <- function(wrkDir){
  return(str_c(wrkDir,"/csvRef/matrix_format_2020-12-04_nchar_5.csv"))
}

getFeatImpFile <- function(wrkDir){
  return (str_c(wrkDir,"/csvRef/xg_feature_importance_5_2020-12-04.csv"))
}

getCasVect <- function(){
  return(c('I-A','I-B','I-C','I-D','I-E','I-F','I-U','II-A','II-B','II-C','III-A','III-B','III-C','III-D','III-E','III-F','IV-A','IV-C','V-A','V-B1','V-B2','V-F','V-J','V-U1','V-U2','V-U4','VI-A','VI-B','VI-C','VI-D'))
}
```

To replace the existing models with the newly trained models directly in the installed R package, first find where the CRISPRclassify package is installed on your system (.libPaths() function in R will show you where packages are installed). CD into that folder, then replace the models in the ./CRISPRclassify/models directory with the new models. If you have forked the repo and want to make changes there instead of directly in the installed R package, the models can be found in the ./inst/models directory. The package will need to be rebuilt (Build > Clean and Rebuild) for the changes to become active.



The **getModelFilePath()** function in the common.R file searches for each subtype provided in the getCasVect() function and specifies the suffix name of each model file (assuming the date and kmer size naming convention mentioned above). Make sure to update the "_2020-12-04_nchar_5" string to match the name of your new models. If this function cannot find the models, they will be skipped during the analysis.

common.R:

```
getModelFilePath <- function(cas, wrkDir) {
  return(str_c(wrkDir,"/models/",cas,"_2020-12-04_nchar_5"))
} # Ex. returns I-A_2020-12-04_nchar_5
```