Deriving variables: Speed

Don Li

05/06/2020

Data and stuff

This follows on from the imputation of distance.

```r
load( "G:/azure_hackathon/datasets2/model_subset/testing_subset1_imputedist.RData" )
load( "G:/azure_hackathon/datasets2/model_subset/testing_historicals.RData" )
historical_vars = colnames( historicals )[
    grepl( "historic", colnames( historicals ) )
    ]
training_set[ historicals, eval(historical_vars) := {
    list( i.historic_timediff,
        i.historic_crow_dist,
        i.historic_path_dist,
        i.historic_sampling_rate,
        i.historic_mean_speed,
        i.historic_log_var_speed
    )
}, on = c("weekday", "hour") ]
test_set[ historicals, eval(historical_vars) := {
    list( i.historic_timediff,
        i.historic_crow_dist,
        i.historic_path_dist,
        i.historic_sampling_rate,
        i.historic_mean_speed,
        i.historic_log_var_speed
    )
}, on = c("weekday", "hour") ]

vars_to_rm = c("trj_id", "timediff", "sampling_rate",
    "sampling_rate_var", "var_speed" )
training_set[ , eval(vars_to_rm) := NULL ]
```

What are we doing?

In this document, we want to consider predicting average journey speed. You will note that in the model inputs, speed is not a given input:

- latitude_origin
- longitude_origin
- latitude_destination
- longitude_destination
- hour_of_day
- day_of_week

Clearly, speed is a useful variable if we want to predict ETA. Speed is included in the trip summaries as `mean_pt_speed` and `var_pt_speed`. The speed is computed by taking the distance between GPS pings and dividing by the time between them.

We will first impute `mean_pt_speed` and then `var_pd_speed`.

Linear regression

```
yvar = "mean_speed"
xvars = setdiff( names(training_set), yvar )
par( mfrow = c(5,4 ) )
col  = rgb( 0, 0, 0, 0.25 )
for( x in xvars ){
    plot( training_set[[x]], training_set[[yvar]],
        pch = 16, col = col, ylab = yvar,
        xlab = x, main = x )
    lines( smooth.spline( training_set[[x]], training_set[[yvar]] ),
        col = "red")
}

model_formula_lm = mean_speed ~ . +
    rush_hour * I(crow_dist^2) +
    rush_hour * I(path_dist_impute^2) +
    I(hour^2) + I(hour^3) +
    I(start_y^2) + I(end_y^2) +
    I( azure_dist^2 ) + I( OSRM_dist^2 ) +
    I( path_dist_impute^2 ) + I(path_dist2_impute^2) +
    crow_dist:azure_dist + crow_dist:OSRM_dist + crow_dist:path_dist_impute +
    crow_dist:historic_path_dist + crow_dist:historic_crow_dist + crow_dist:historic_sampling_rate +
    weekday:start_y + weekday:start_x + weekday:end_x + weekday:end_y +
    weekday:path_dist_impute + weekday:azure_dist + weekday:OSRM_dist +
    hour * rush_hour +
    rush_hour:start_y + rush_hour:start_x + rush_hour:end_x + rush_hour:end_y

lm_ = train( form = enet_formula,
    data = training_set,
    metric = "RMSE", method = "lm", trControl = train_control)

lm_results = data.table( lm_$results )
lm_pred = data.table( lm_$pred )
setorder( lm_pred, rowIndex )

save( lm_, lm_results, lm_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/lm.RData" )
```

```
load( "G:/azure_hackathon/datasets2/expo_speed/lm.RData" )
lm_results
```

```
##    intercept        RMSE Rsquared         MAE      RMSESD RsquaredSD
## 1:      TRUE 0.002202268 0.661809 0.001736866 6.01315e-05 0.02385513
##           MAESD
## 1: 3.484545e-05
```

Elastic net

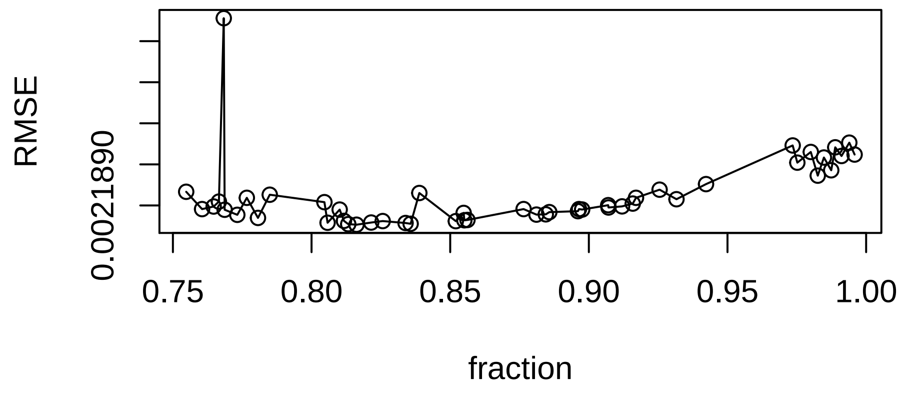
Elastic net. No interactions in the enet because it is very slow.

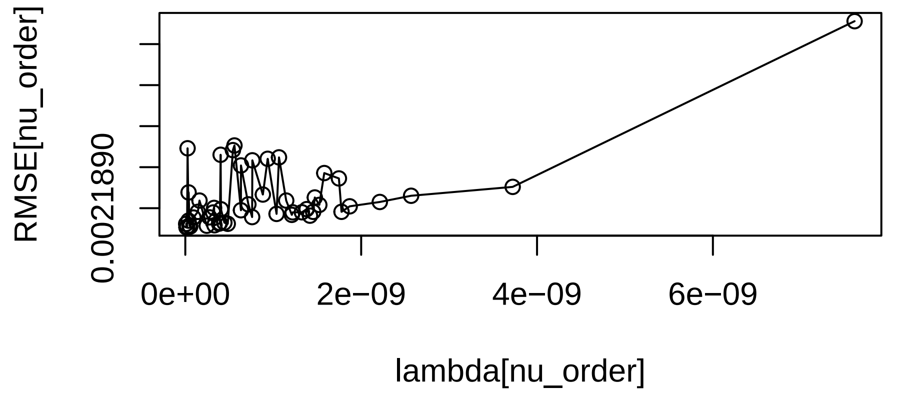
```
n_enet = 50
enet_tunegrid = data.frame(
    lambda = rexp( n_enet, 1/0.000000001 ),
    fraction = runif( n_enet, 0.75, 1 )
)
enet_ = train( form = model_formula_lm, data = training_set,
    metric = "RMSE", method = "enet", trControl = train_control,
    tuneGrid = enet_tunegrid, standardize = TRUE, intercept = TRUE
    )

enet_results = data.table( enet_$results )
enet_pred = data.table( enet_$pred )
setorder( enet_pred, rowIndex )

save( enet_, enet_results, enet_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/enet.RData" )
```

```
load( "G:/azure_hackathon/datasets2/expo_speed/enet.RData" )
enet_results[ which.min(RMSE) ]
```

```
##          lambda fraction        RMSE  Rsquared         MAE       RMSESD
## 1: 1.343621e-11 0.816212 0.002188906 0.6661094 0.001723912 5.486761e-05
##    RsquaredSD        MAESD
## 1: 0.01917431 4.423906e-05
```

```
par( mfrow = c(2, 1 ) )
enet_results[ , {
    plot( fraction, RMSE, type = "o" )
    nu_order = order(lambda)
    plot( lambda[nu_order], RMSE[nu_order], type = "o" )
} ]
```





```
## NULL
```

Partial least squares

PLS.

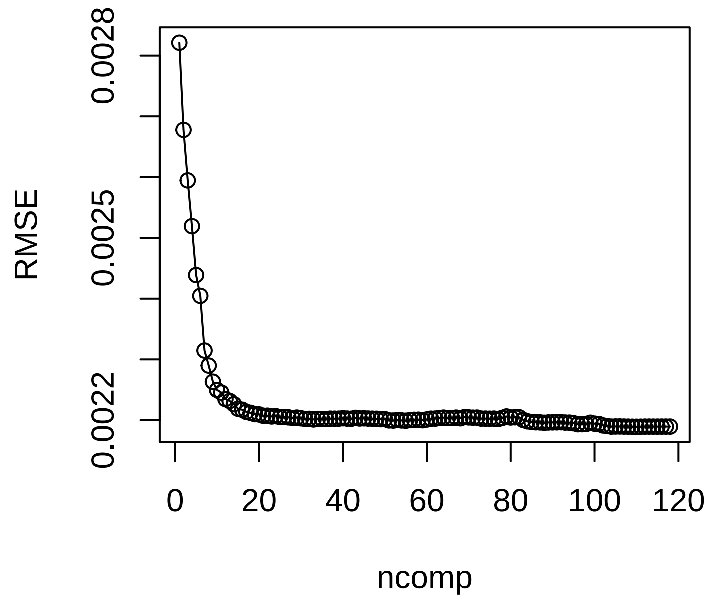
```
full_X = ncol( model.matrix( model_formula_lm, training_set) )

pls_tunegrid = data.frame( ncomp = 1:full_X )

pls_ = train( form = model_formula_lm, data = training_set,
    metric = "RMSE", method = "pls", trControl = train_control,
    tuneGrid = pls_tunegrid, scale = T
    )

pls_results = data.table( pls_$results )
```

```r
pls_pred = data.table( pls_$pred )
setorder( pls_pred, rowIndex )

save( pls_, pls_results, pls_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/pls.RData" )
```

```r
load( "G:/azure_hackathon/datasets2/expo_speed/pls.RData" )
pls_results[ which.min(RMSE) ]
```

```
##    ncomp        RMSE  Rsquared         MAE       RMSESD RsquaredSD        MAESD
## 1:   110 0.002189175 0.6660742 0.00172433 5.485825e-05 0.01917639 4.421795e-05
```

```r
pls_results[ RMSE < 0.005, {
    plot( ncomp, RMSE, type = "o" )
} ]
```



```
## NULL
```

Principal components regression

PCR.

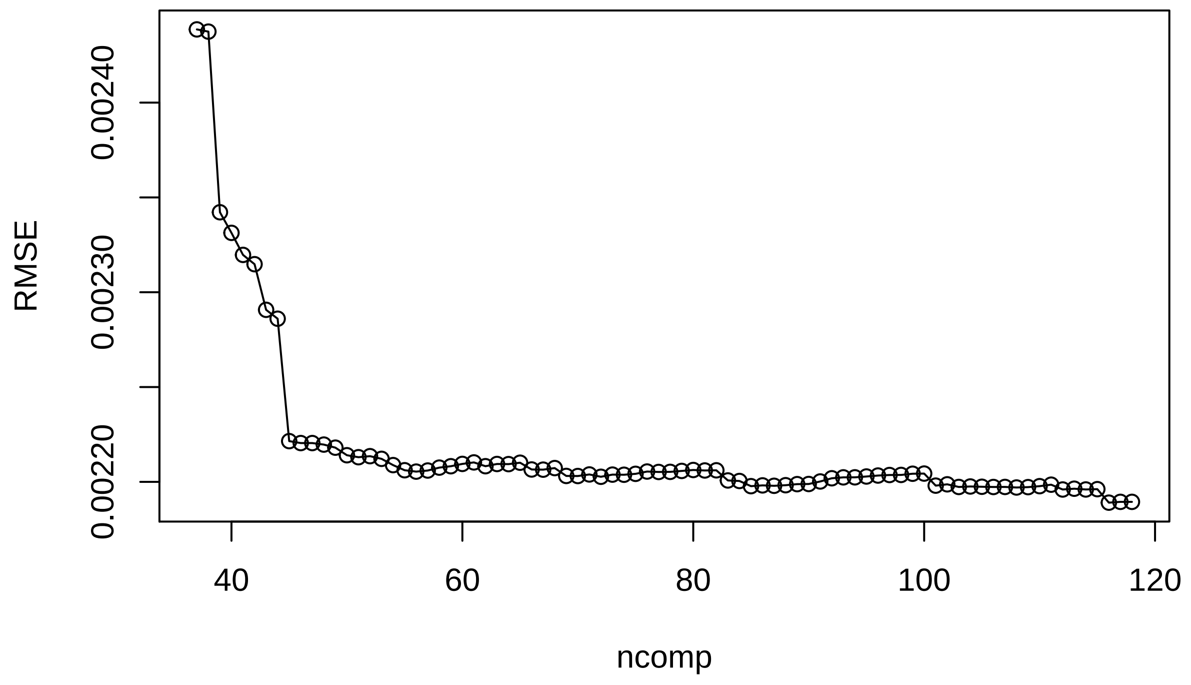
```r
full_X = ncol( model.matrix( model_formula_lm, training_set) )
pcr_tunegrid = data.frame( ncomp = 1:full_X )
pcr_ = train( form = model_formula_lm, data = training_set,
    metric = "RMSE", method = "pcr", trControl = train_control,
    tuneGrid = pcr_tunegrid, scale = T
    )
```

```r
pcr_results = data.table( pcr_$results )
pcr_pred = data.table( pcr_$pred )
setorder( pcr_pred, rowIndex )

save( pcr_, pcr_results, pcr_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/pcr.RData" )
```

```r
load( "G:/azure_hackathon/datasets2/expo_speed/pcr.RData" )
pcr_results[ which.min(RMSE) ]
```

```
##    ncomp        RMSE  Rsquared         MAE       RMSESD RsquaredSD        MAESD
## 1:   116 0.002189038 0.6661158 0.001724253 5.475017e-05 0.01917707 4.407067e-05
```

```r
pcr_results[ RMSE < 0.0025, {
    plot( ncomp, RMSE, type = "o" )
} ]
```



```
## NULL
```

KNN

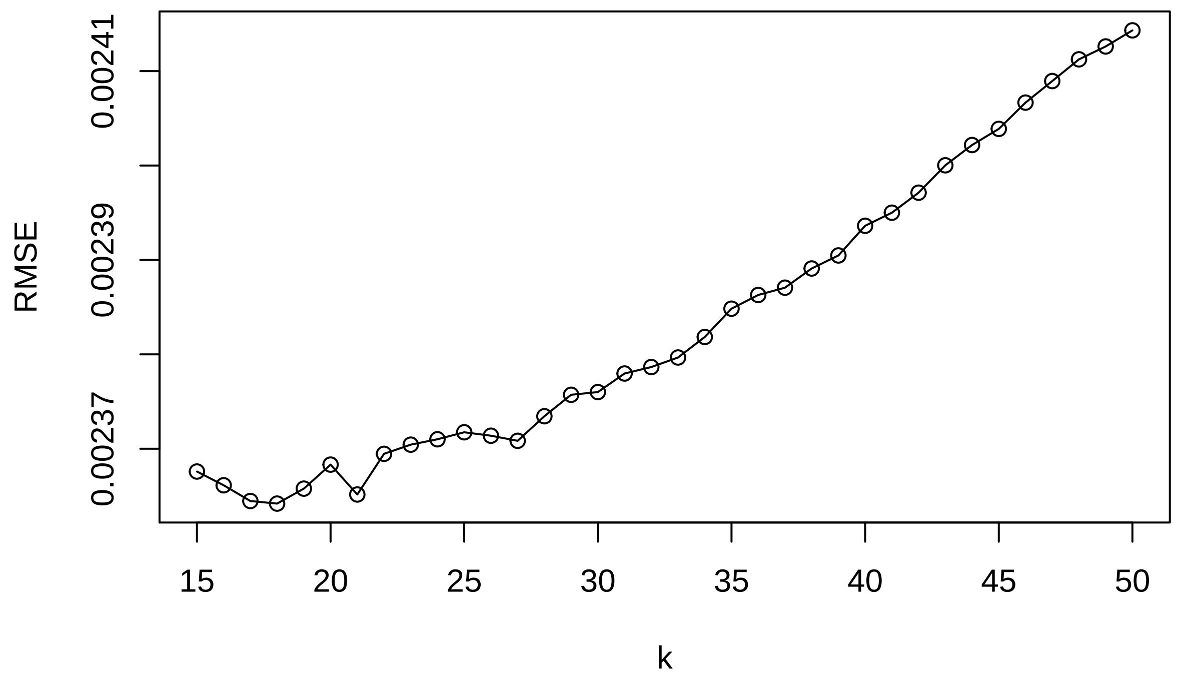
KNN.

```r
k_grid = data.frame( k = 15:50 )

knn_ = train( form = model_formula_lm, data = training_set,
    metric = "RMSE", method = "knn", trControl = train_control,
    tuneGrid = k_grid
    )

knn_results = data.table( knn_$results )
knn_pred = data.table( knn_$pred )
setorder( knn_pred, rowIndex )

save( knn_, knn_results, knn_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/knn.RData" )
```

```
load( "G:/azure_hackathon/datasets2/expo_speed/knn.RData" )
knn_results[ which.min(RMSE) ]
```

```
##     k       RMSE  Rsquared         MAE       RMSESD RsquaredSD        MAESD
## 1: 18 0.00236419 0.6114566 0.001871393 5.432699e-05 0.02493647 3.363607e-05
```

```
knn_results[ , plot( k, RMSE, type = "o" ) ]
```



```
## NULL
```

CART

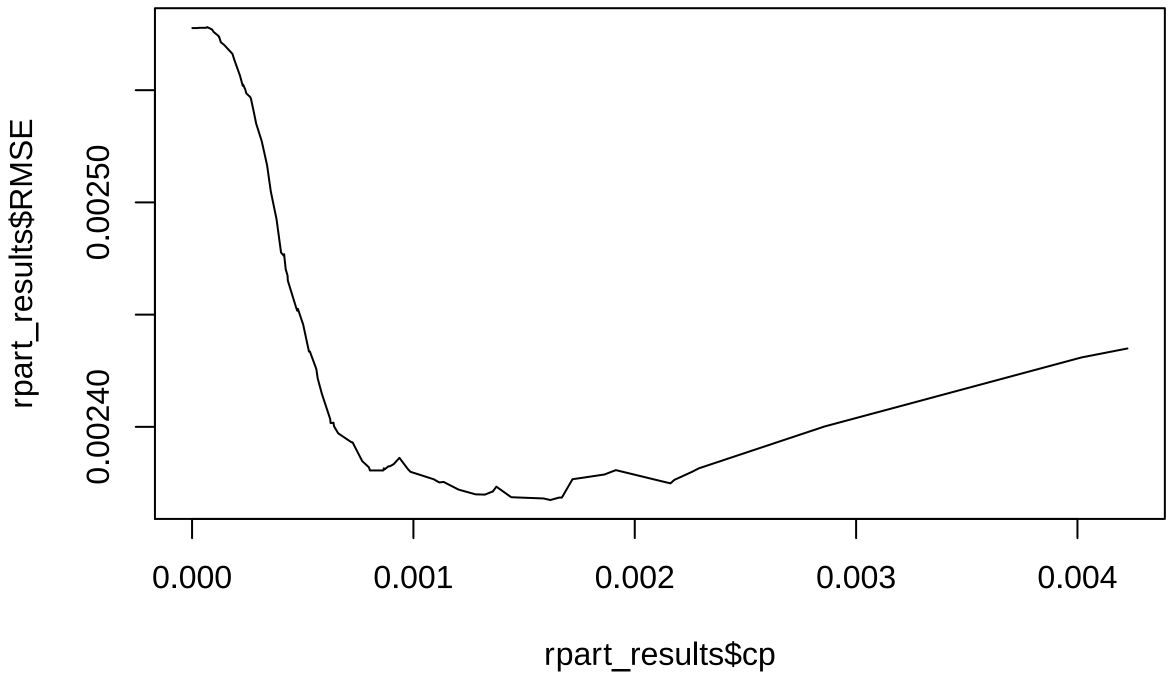
Use an Exponential distribution for our random search.

```
cp_grid = data.frame( cp = rexp( 100, 1/0.001 ) )

rpart_ = train( mean_speed ~ .,
    data = training_set,
    metric = "RMSE", method = "rpart", trControl = train_control,
    tuneGrid = cp_grid
    )

rpart_results = data.table( rpart_$results )
rpart_pred = data.table( rpart_$pred )
setorder( rpart_pred, rowIndex )

save( rpart_, rpart_results, rpart_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/rpart.RData" )
```

```
load( "G:/azure_hackathon/datasets2/expo_speed/rpart.RData" )
rpart_results[ which.min(RMSE) ]
```

```
##              cp       RMSE  Rsquared         MAE       RMSESD RsquaredSD
## 1: 0.001618599 0.00236741 0.6098749 0.001870525 7.435699e-05 0.02326198
##            MAESD
## 1: 5.990607e-05
```

```
plot( rpart_results$cp, rpart_results$RMSE, type = "l" )
```



GAM splines

Generalised additive model using splines

```
gam_grid = expand.grid(
    select = F,
    method = c( "GACV.Cp" )
)

gam_ = train( mean_speed ~ ., data = training_set,
    metric = "RMSE", method = "gam", trControl = train_control,
    tuneGrid = gam_grid
)

gam_results = data.table( gam_$results )
gam_pred = data.table( gam_$pred )
setorder( gam_pred, rowIndex )

save( gam_, gam_results, gam_pred,
    file = "G:/azure_hackathon/datasets2/expo_speed/gam_spline.RData" )
```

```
load( "G:/azure_hackathon/datasets2/expo_speed/gam_spline.RData" )
gam_results[ which.min(RMSE) ]
```

```
##    select  method        RMSE  Rsquared         MAE       RMSESD RsquaredSD
## 1:  FALSE GACV.Cp 0.002163538 0.6736461 0.001710183 5.788122e-05 0.02018379
##           MAESD
## 1: 3.965227e-05
```

Stacking

```
training_OOF = data.table(
    lm = lm_pred$pred,
    pls = pls_pred$pred,
    pcr = pcr_pred$pred,
    knn = knn_pred$pred,
    rpart = rpart_pred$pred,
    gam = gam_pred$pred,
    mean_speed = training_set$mean_speed
)

test_OOF = data.table(
    lm = predict( lm_, test_set ),
    pls = predict( pls_, test_set ),
    pcr = predict( pcr_, test_set ),
    knn = predict( knn_, test_set ),
    rpart = predict( rpart_, test_set ),
    gam = predict( gam_, test_set ),
    mean_speed = test_set$mean_speed
)

stacking_model_formula = as.formula( "mean_speed ~ ." )
```

```r
stacked_tunegrid = data.frame( ncomp = 1:(ncol(training_OOF)-1) )
stacking = train( stacking_model_formula, data = training_OOF,
    metric = "RMSE", method = "pls", trControl = train_control,
    tuneGrid = stacked_tunegrid
    )

stacking_results = data.table( stacking$results )
stack_pred_OOF = predict( stacking, test_OOF )

save( stacking_results, stack_pred_OOF, test_OOF,
    file = "G:/azure_hackathon/datasets2/expo_speed/stack.RData" )
```

```r
load( "G:/azure_hackathon/datasets2/expo_speed/stack.RData" )

stacked_rmse = sqrt( mean( ( stack_pred_OOF - test_OOF$mean_speed )^2 ) )

all_rmse = rbind(
    as.matrix(sqrt( colMeans( ( test_OOF - test_OOF$mean_speed )^2 ) )),
    stack = stacked_rmse
)

all_rmse[ order(all_rmse), ]
```

```
##  mean_speed       stack         gam         pls         pcr          lm
## 0.000000000 0.002299025 0.002315605 0.002349877 0.002350024 0.002363222
##         knn       rpart
## 0.002458646 0.002480831
```

Conclusion

Stack a bunch of models for imputing mean point speed. Also do this for the variance fo the point speed.