ISE 5103 Intelligent Data Analytics

Homework #6

(a) (50 points) You must build at least 5 different classes of models.

Linear modeling (lm): *Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.*

Hyperparameters: NA

```
data_ctrl <- trainControl(method = "cv", number = 5)
model_caret <- train(logSumRevenue ~ channelGrouping +
                       as.numeric(first_ses_from_the_period_start) +
                       as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
                       log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
                       country + region + networkDomain + source +
                       log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
                       log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
                       pageviews_median + log(session_cnt),   # model to fit
                     data = tf,
                     trControl = data_ctrl,              # folds
                     method = "lm",                      # specifying regression model
                    na.action = na.pass)# pass missing data to model-some models will handle this

model_caret
model_caret$finalModel
model_caret$resample
```

```
> model_caret
Linear Regression

47249 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 37799, 37799, 37800, 37799, 37799
Resampling results:

  RMSE       Rsquared   MAE
  0.8058229  0.6215821  0.4299174


> model_caret$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
                                  (Intercept)                 channelGroupingAffiliates
                                   -1.052e+00                                 3.656e-01
                        channelGroupingDirect                    channelGroupingDisplay
                                   -2.468e-01                                -2.740e-02
                `channelGroupingOrganic Search`             `channelGroupingPaid Search`
                                   -1.627e-01                                -1.944e-01
                      channelGroupingReferral                     channelGroupingSocial
                                    4.744e-03                                -6.813e-02
`as.numeric(first_ses_from_the_period_start)`  `as.numeric(last_ses_from_the_period_end)`
                                    1.855e-03                                 1.438e-03
                   `log(unique_date_num + 1)`                   `log(maxVisitNum + 1)`
                                    1.228e-01                                -2.689e-01
                               browserFirefox                `browserInternet Explorer`
> model_caret$resample
       RMSE  Rsquared       MAE Resample
1 0.8202312 0.6153305 0.4410175    Fold1
2 0.7993711 0.6316806 0.4285505    Fold2
3 0.8233995 0.6012156 0.4307036    Fold3
4 0.7876000 0.6175321 0.4198572    Fold4
5 0.7985127 0.6421515 0.4294583    Fold5
```

Partial Least Squares (PLS):): *In short, partial least squares regression is probably the least restrictive of the various multivariate extensions of the multiple linear regression model. This flexibility allows it to be used in situations where the use of traditional multivariate methods is severely limited, such as when there are fewer observations than predictor variables. Furthermore, partial least squares regression can be used as an exploratory analysis tool to select suitable predictor variables and to identify outliers before classical linear regression.*

Hyperparameters: NA

```
data_ctrl <- trainControl(method = "cv", number = 5)
plscvmodel <- train(logSumRevenue ~ channelGrouping +
                        as.numeric(first_ses_from_the_period_start)+
                        as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
                        log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
                        country + region + networkDomain + source +
                        log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
                        log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
                        pageviews_median + log(session_cnt),   # model to fit
                      data = tf,
                      trControl = data_ctrl,              # folds
                      method = "pls",                      # specifying regression model
)                # pass missing data to model - some models will handle this

plscvmodel
plscvmodel$finalModel
plscvmodel$resample
```

```
> plscvmodel
Partial Least Squares

47249 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 37799, 37799, 37799, 37799, 37800
Resampling results across tuning parameters:

  ncomp  RMSE       Rsquared    MAE
   1     1.3265906  0.05133033  0.7667537
   2     1.1519827  0.23316891  0.6958444
   3     0.9916082  0.44086545  0.4732686
   4     0.9643760  0.46850013  0.4460253
   5     0.9535039  0.47702375  0.4405561
   6     0.8954146  0.53472574  0.4270946
   7     0.8507220  0.57860536  0.5023220
   8     0.8465766  0.58272113  0.4832887
   9     0.8395832  0.58940400  0.4887687
  10     0.8355927  0.59331520  0.4736379

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was ncomp = 10.
> plscvmodel$finalModel
Partial least squares regression , fitted with the orthogonal scores algorithm.
Call:
plsr(formula = .outcome ~ ., ncomp = ncomp, data = dat, method = "oscorespls")
> plscvmodel$resample
       RMSE  Rsquared       MAE Resample
1 0.8290028 0.5940071 0.4725452    Fold1
2 0.8567984 0.5735829 0.4674792    Fold3
3 0.8324465 0.5962807 0.4783695    Fold4
4 0.8310393 0.6039956 0.4763420    Fold5
5 0.8286768 0.5987097 0.4734536    Fold2
```

Glment package:

We using *glmnet* package for all Ridge, Lasso, Elastic net regression. Where it has parameter to take like below

x: matrix of predictor variables

y: the response or outcome variable, which is a binary variable.

alpha: the elasticnet mixing parameter. Allowed values include:

“1”: for lasso regression

"0": for ridge regression

- a value between 0 and 1 (say 0.3) for elastic net regression.

lambda: a numeric value defining the amount of shrinkage. Should be specify by analyst.

In penalized regression, we need to specify a constant lambda to adjust the amount of the coefficient shrinkage. The best lambda for your data, can be defined as the lambda that minimize the cross-validation prediction error rate. This can be determined automatically using the function *cv.glmnet()*. We can also use caret function using method as a glmnet.

Ridge regression: *Ridge regression shrinks the regression coefficients, so that variables, with minor contribution to the outcome, have their coefficients close to zero. The shrinkage of the coefficients is achieved by penalizing the regression model with a penalty term called L2-norm, which is the sum of the squared coefficients.*

Hyperparameters: Here we call them as lambda. We have to take it in a range. So that model will fit to the best minimum lambda value.

```
lambda <- seq(0.01,0.1 ,length = 100)
set.seed(123)
# Find the best lambda using cross-validation
ridge <- train(logSumRevenue ~ channelGrouping + as.numeric(first_ses_from_the_period_start) +
    as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
    log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
    country + region + networkDomain + source +
    log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
    log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
    pageviews_median + log(session_cnt),   # model to fit
  data = tf, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 0, lambda = lambda)
)
```

```
ridge$bestTune$lambda
```

[1] 0.09272727

```
> coef(ridge$finalModel, ridge$bestTune$lambda) # best model coeffients
```

Appendix

```
(Intercept)                                  -9.370004e-01
channelGroupingAffiliates                     1.435751e-01
channelGroupingDirect                        -5.255527e-02
channelGroupingDisplay                        1.037095e-01
channelGroupingOrganic Search                -5.729221e-02
channelGroupingPaid Search                   -9.643599e-02
channelGroupingReferral                       1.479894e-01
channelGroupingSocial                         2.046670e-02
as.numeric(first_ses_from_the_period_start)   4.634674e-04
as.numeric(last_ses_from_the_period_end)      5.789726e-05
log(unique_date_num + 1)                      2.823471e-01
log(maxVisitNum + 1)                          3.424874e-02
browserFirefox                                1.429348e-02
browserInternet Explorer                      3.148888e-02
browserSafari                                -5.834081e-02
browserOther                                  3.970519e-02
operatingSystemBlackBerry                     1.651493e-01
operatingSystemChrome OS                      1.005387e-01
operatingSystemFirefox OS                     1.061465e-01
```

```
predictions <- ridge %>% predict(tf)
# Model prediction performance
data.frame(
  RMSE = RMSE(predictions, tf$logSumRevenue),
  Rsquare = caret::R2(predictions, tf$logSumRevenue)
)
```

	RMSE	Rsquare
1	0.8295985	0.5990012

Resampled Rsquared and RMS values

```
> ridge$resample
         RMSE  Rsquared       MAE Resample
1  0.8808356 0.5655831 0.4815369   Fold09
2  0.7969837 0.6153086 0.4601428   Fold05
3  0.8344111 0.5966121 0.4700110   Fold01
4  0.8435888 0.5605082 0.4900705   Fold08
5  0.8336563 0.5824750 0.4690597   Fold03
6  0.8350183 0.5985940 0.4804787   Fold07
7  0.8378343 0.5912706 0.4845969   Fold10
8  0.8125419 0.6246620 0.4665132   Fold04
9  0.8470996 0.5976116 0.4844960   Fold02
10 0.8271351 0.6082212 0.4798585   Fold06
```

Lasso regression: *Lasso stands for Least Absolute Shrinkage and Selection Operator. It shrinks the regression coefficients toward zero by penalizing the regression model with a penalty term called L1-norm, which is the sum of the absolute coefficients.*

Hyperparameters: Same as ridge regression. Taking a factor of lambda's to find the best using cross validation.

```
set.seed(123)
lasso <- train(
  logSumRevenue ~ channelGrouping + as.numeric(first_ses_from_the_period_start) +
    as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
    log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
    country + region + networkDomain + source +
    log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
    log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
    pageviews_median + log(session_cnt),   # model to fit
  data = tf, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = 1, lambda = lambda)
)
# Model coefficients
coef(lasso$finalModel, lasso$bestTune$lambda)
```

Appendix for coefficients

```
> coef(lasso$finalModel, lasso$bestTune$lambda)
73 x 1 sparse Matrix of class "dgCMatrix"
                                                       1
(Intercept)                                -0.8983799689
channelGroupingAffiliates                   .
channelGroupingDirect                       .
channelGroupingDisplay                      .
channelGroupingOrganic Search              -0.0064888740
channelGroupingPaid Search                 -0.0030291499
channelGroupingReferral                     0.0844310290
channelGroupingSocial                       .
as.numeric(first_ses_from_the_period_start) 0.0003069175
as.numeric(last_ses_from_the_period_end)    .
log(unique_date_num + 1)                    0.1090071179
log(maxVisitNum + 1)                        .
browserFirefox                              .
browserInternet Explorer                    .
browserSafari                              -0.0259864185
browserOther                                .
operatingSystemBlackBerry                   .
operatingSystemChrome OS                    0.0494643326
operatingSystemFirefox OS                   .
```

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.01.

```
> # Make predictions
> predictions <- lasso %>% predict(tf)
> # Model prediction performance
> data.frame(
+   RMSE = RMSE(predictions, tf$logSumRevenue),
+   Rsquare = caret::R2(predictions, tf$logSumRevenue)
+ )
```

```
      RMSE   Rsquare
1 0.8285392 0.5999455
```

Resampled scores for lasso

```
> lasso$resample
        RMSE  Rsquared       MAE Resample
1  0.8786501 0.5679124 0.4774470   Fold09
2  0.8332997 0.6001437 0.4755548   Fold07
3  0.7938905 0.6183881 0.4558801   Fold05
4  0.8485061 0.5961636 0.4812845   Fold02
5  0.8378810 0.5912076 0.4823584   Fold10
6  0.8345332 0.5963267 0.4666522   Fold01
7  0.8412138 0.5629835 0.4858255   Fold08
8  0.8362171 0.5804206 0.4656391   Fold03
9  0.8254500 0.6097210 0.4766136   Fold06
10 0.8103462 0.6266575 0.4628007   Fold04
```

Elastic net regression: *Elastic Net produces a regression model that is penalized with both the L1-norm and L2-norm. The consequence of this is to effectively shrink coefficients (like in ridge regression) and to set some coefficients to zero (as in LASSO).*

Hyperparameters: same as lasso and ridge regression models. Only alpha will be the value in between 0 and 1. That also we are checking with all possible values to get best alpha.

```
enetmodel <- train(
  logSumRevenue ~ channelGrouping + as.numeric(first_ses_from_the_period_start) +
    as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
    log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
    country + region + networkDomain + source +
    log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
    log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
    pageviews_median + log(session_cnt),   # model to fit
    data = tf, method = "glmnet",
  trControl = trainControl("cv", number = 10),
  tuneGrid = expand.grid(alpha = seq(0,1,length=10), lambda = lambda))
# Best tuning parameter
enetmodel$bestTune
```

best alpha and lambda scores are 0.1111111 and 0.01 respectivly.

Appendix for coefficients of Elasticnet

```
> coef(enetmodel$finalModel, enetmodel$bestTune$lambda)
73 x 1 sparse Matrix of class "dgCMatrix"
                                                     1
(Intercept)                                -1.138498e+00
channelGroupingAffiliates                   3.314335e-01
channelGroupingDirect                      -4.518353e-02
channelGroupingDisplay                      9.952745e-02
channelGroupingOrganic Search              -3.492970e-02
channelGroupingPaid Search                 -8.201115e-02
channelGroupingReferral                     1.477029e-01
channelGroupingSocial                       2.584285e-02
as.numeric(first_ses_from_the_period_start) 8.581188e-04
as.numeric(last_ses_from_the_period_end)    4.279589e-04
log(unique_date_num + 1)                    3.073354e-01
log(maxVisitNum + 1)                       -9.180582e-02
browserFirefox                              1.007799e-02
browserInternet Explorer                    2.336381e-02
browserSafari                              -5.641815e-02
browserOther                                3.589505e-02
```

```
predictions <- enetmodel %>% predict(tf)
# Model performance metrics
data.frame(
  RMSE = RMSE(predictions, tf$logSumRevenue),
  Rsquare = caret::R2(predictions, tf$logSumRevenue)
)
```

```
      RMSE   Rsquare
1 0.8234472 0.6046513
```

Rsampled scores for elastic net are

```
> enetmodel$resample
        RMSE  Rsquared       MAE Resample
1  0.7792630 0.6111800 0.4504834   Fold08
2  0.8397878 0.5785985 0.4742028   Fold04
3  0.7970957 0.6330348 0.4616909   Fold09
4  0.8198869 0.6074159 0.4600720   Fold02
5  0.8676469 0.5810468 0.4710370   Fold10
6  0.8712141 0.5767076 0.4701164   Fold07
7  0.8422950 0.5927631 0.4683526   Fold03
8  0.8094489 0.6138668 0.4607027   Fold05
9  0.8371004 0.5953533 0.4768485   Fold01
10 0.8525112 0.5843560 0.4883789   Fold06
```

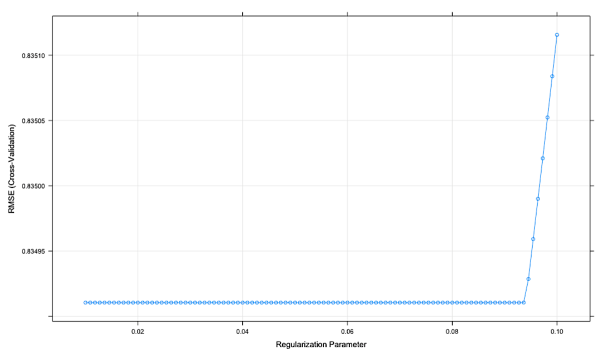
MARS: *Multivariate adaptive regression splines (MARS) provide a convenient approach to capture the nonlinearity aspect of polynomial regression by assessing cutpoints (knots) similar to step functions. The procedure assesses each data point for each predictor as a knot and creates a linear regression model with the candidate feature(s).*

This is the one we selected as our final model which is giving best performance out of all. Please refer in b(i) section for more information about MARS.

- **Choose one model with hyper-parameters and justify your choice on how you tuned the model. Please support with one or more visualizations.**

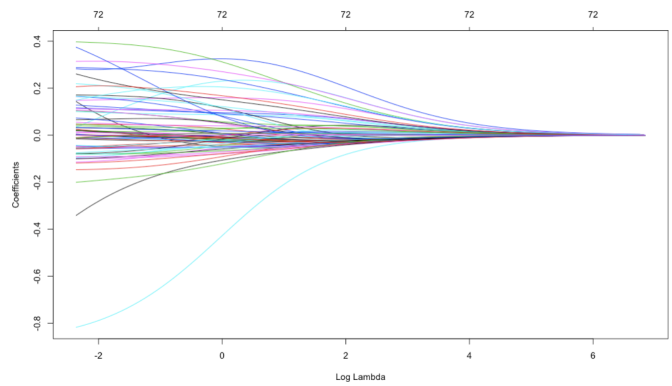
  Ridge tuning:

  In Section 1(a) we already explained about Ridge Model Implementation. Where we have to choose our hyper parameter lambda from given sequence. We have taken range between 0.01 and 0.1. for which we got 0.09 as a best hyper parameter.

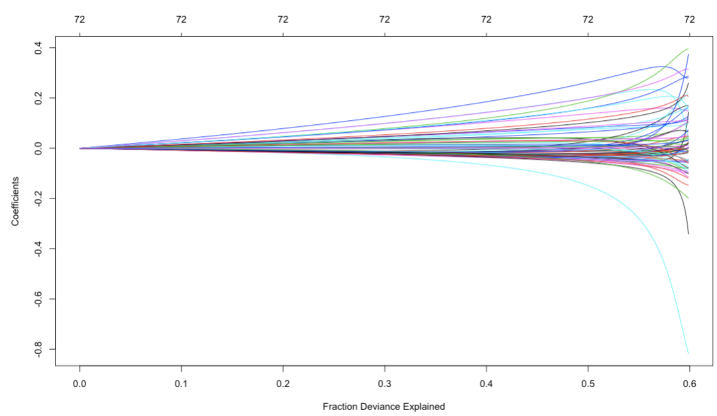


  From the plot (RMSE and regularization parameter) we can see at 0.09 RMSE is very less and then there is a drastic change in the RMSE. As the error is increasing from then it has given the 0.09 as the best hyperparameter.

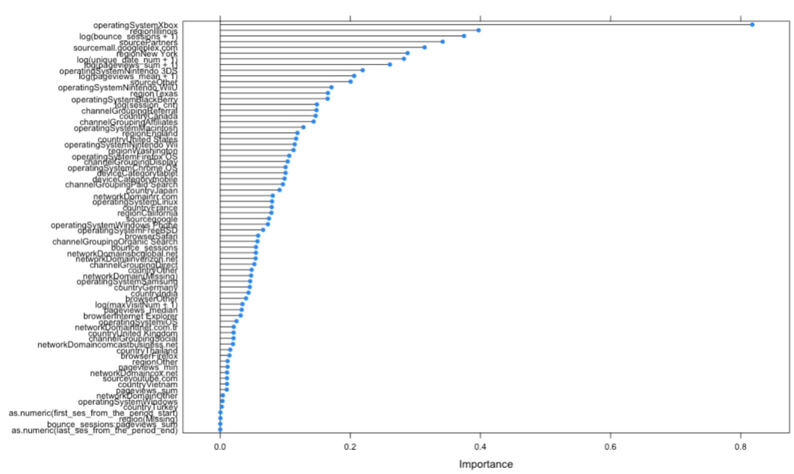
  Lets see how coefficients are changing as lambda values are changing. Bellow plot is showing the relation b/w coefficients and lambda values. As we can see there is a shrinking in coefficients as lamda is increasing. which means as we increase lamda we are increasing penalty (decreasing coefficient values shrink) When log labma is more all the coefficient are zero and as we relax labma coefficients increase. At top explaining about at what point how many independent variables we have.



We can also check for if out model is overfitting because of which variables. The below plot explains how it happens. If there is a huge deviation for coefficients it means that variables are leading to overfitting.



From the below plot we can also say what variables are impacting more on the target variable.



- **From your work, choose two different model class instances and compare/contrast the results in detail, e.g., you may discuss differences in regression coefficients, model complexity, residual diagnostics, etc.**

LM vs MARS

As above we already explained about linear model and MARS. As lm only captures linearity with the variables. But not the non-linear relationships. Whereas polynomial functions impose a global non-linear relationship. MARS works on this polynomial function.

We got huge difference in the performance with MARS when compared with lm. As you can see above (from a section lm) and below (from b section MARS) **coefficients** are different. For lm we got linear **coefficients** for all variables. But on the other hand, for MARS it has knots and pruned all not needed knots and got polynomial functionality with variables.

	R2	**RMSE**
MARS	0.694	0.718
lm	0.621	0.805

Another huge difference is hyperparameter. In lm we don't have anything like that to reduce the **coefficients.**

Coming to complexity from us personally execution took less time for lm compared to MARS. Which is totally reasonable. As there are many calculations to be done in MARS which has to consider degree, nprune along with cross validation. Infact we took only 3 folds and upto 3 degrees of hyperparameters only. Still it took quite a long to execute.

- **Summarize all model performances in a table that identifies: R method and underlying library (not caret), specifics with respect to tuning parameters, and re-sampled performance metrics. Include results from your Homework #5 OLS model**.

Model	**Method**	**Package**	**Hyperparameter**	**Selection**	R^2	**RMSE**
OLS -HW5	lm	Stats	NA	NA	0.621	0.805
PLS	pls	pls	NA	NA	0.57	0.82
Ridge	glmnet	glmnet	factor	0.092	0.599	0.829
Lasso	glmnet	glmnet	factor	0.01	0.599	0.828
Elasticnet	glmnet	glmnet	factor	0.01	0.604	0.823
MARS	earth	earth	degree	3	0.694	0.718

(b) (50 points) Build the best possible regression model(s) to predict the target value.

i. (15 points) For your best model, report the variables, coefficient estimates, and p-values (you may do this in the appendix if it is a large model) Additionally, report the re-sampled RMSE and R^2 values as well as any tuning parameter values. Describe your modeling approach, e.g., did you examine interactions? did you create use more than one model? what was your secret sauce?

As we see in the above table. We got better performance by using MARS model from earth package.

We took same formula as we took for rest of Method. MARS model can be applied by using earth package and earth model. I trained data without degree which is the hyperparameter here. Then we got better results than another other model we got till now – Rsquared – 0.67. Then gave degree as 2 to train and test out model. Now we got Rsquared – 0.717.

The procedure assesses each data point for each predictor as a knot and creates a linear regression model with the candidate feature(s). Consequently, once the full set of knots have been created, we can sequentially remove knots that do not contribute significantly to predictive accuracy. This process is known as “pruning” and we can use cross-validation for this. In additionally this n-pruning, allows us to also assess potential interactions between different functions.
As we are not sure about what degree to take we are considering more degree values (1:3) and doing cross validation.

Since there are two tuning parameters associated with our MARS model: the degree of interactions and the number of retained terms, we need to perform a grid search to identify the optimal combination of these hyperparameters that minimize prediction error.

```
set.seed(123)
hyper_grid <- expand.grid(
  degree = 1:3,
  nprune = seq(2, 100, length.out = 10) %>% floor()
)
# cross validated model
set.seed(123)
tuned_mars <- train(
   logSumRevenue ~ channelGrouping + as.numeric(first_ses_from_the_period_start) +
    as.numeric(last_ses_from_the_period_end) +  log(unique_date_num+1) +
    log(maxVisitNum+1) + browser + operatingSystem + deviceCategory +
    country + region + networkDomain + source +
    log(bounce_sessions+1) + bounce_sessions*pageviews_sum +
    log(pageviews_sum+1) + log(pageviews_mean+1) + pageviews_min +
    pageviews_median + log(session_cnt),
  data = tf,
  method = "earth",
  metric = "RMSE",
  trControl = trainControl(method = "cv", number = 3),
  tuneGrid = hyper_grid
)
```

```
# best model
tuned_mars$bestTune
```

nprune degree
23 3

```
tuned_mars$finalModel$coefficients
```

```
> tuned_mars$finalModel$coefficients
                                                                                                              y
(Intercept)                                                                                        1.999755e-03
h(log(pageviews_sum + 1)-2.19722)                                                                  1.556553e+00
countryUnited States*h(log(pageviews_sum + 1)-2.19722)                                             1.201729e-01
countryUnited States*h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722)                         2.412107e-02
h(log(pageviews_sum + 1)-2.19722)*h(log(pageviews_mean + 1)-2.37158)                              -4.146197e-02
h(log(pageviews_sum + 1)-2.19722)*h(2.37158-log(pageviews_mean + 1))                              -4.546623e+00
deviceCategorymobile*h(log(pageviews_sum + 1)-2.19722)                                            -4.207226e-01
h(as.numeric(last_ses_from_the_period_end)-355)*countryUnited States*h(log(pageviews_sum + 1)-2.19722)  -8.974155e-02
h(355-as.numeric(last_ses_from_the_period_end))*countryUnited States*h(log(pageviews_sum + 1)-2.19722)   3.020874e-04
countryUnited States*sourcegoogle*h(log(pageviews_sum + 1)-2.19722)                               -4.302315e-02
h(pageviews_sum-88)*h(log(pageviews_sum + 1)-2.19722)                                             -2.715253e-04
h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722)                                             -3.737463e-03
operatingSystemMacintosh*h(log(pageviews_sum + 1)-2.19722)                                         1.819713e-01
h(log(pageviews_sum + 1)-2.19722)*h(2.37158-log(pageviews_mean + 1))*h(pageviews_min-6)           -4.367677e+00
h(log(pageviews_sum + 1)-2.19722)*h(2.37158-log(pageviews_mean + 1))*h(6-pageviews_min)            7.111586e-01
h(as.numeric(first_ses_from_the_period_start)-280)*h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722) -4.916194e-05
h(280-as.numeric(first_ses_from_the_period_start))*h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722) -3.766199e-05
sourcegoogle*h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722)                                -7.408962e-03
sourceyoutube.com*h(88-pageviews_sum)*h(log(pageviews_sum + 1)-2.19722)                           -1.869610e-02
```

The above are the coefficients of Mars model. For the best tuning of nprune as 23 and degree as 3

```
tuned_mars$resample
> tuned_mars$resample
       RMSE  Rsquared       MAE Resample
1 0.6890163 0.7161871 0.2425003    Fold2
2 0.7020092 0.7154595 0.2387663    Fold3
3 0.6936648 0.7242198 0.2377096    Fold1
```

As we can see all the RSquared are around 0.71 which has to more. The more it is the better the model. And we can also say there is not model overfitting. Since there is no huge difference in the values for both RSquared and RMSE. We took degree range 1-3 in out cross validation. If we take more test may be model will perform very well.

ii. **(35 points) Submit your model predictions to the Kaggle.com competition website and outperform your peers in high quality predictions on the test data. You can submit multiple times each day to get feedback on the "public leaderboard". The final competition placement will be based on the "private leaderboard" standings. See the competition website for more details.**

Submitted predicted revenue using **MARS** Model results on to Kaggle. Please find our team name – **(C) AS 13.**