TRABALHO DE ESTATÍSTICA APLICADA II (IAA005)

EQUIPE:

Gabriel Rulka Tardoski

Jordhan Emmanuel Marciano da Silva

Raul Cordeiro Chiarella

Zaira Mendonça Amorim

1. Sobre o trabalho

"O enunciado pedia que fizéssemos as regressões Ridge, Lasso e ElasticNet com a variável dependente 'lwage' (salário-hora da esposa em logaritmo neperiano) e todas as demais variáveis explicativas na base de dados (todas aquelas que tentam explicar o salário-hora da esposa). Com esse intuito, fizemos o seguinte processo:"

Pré-processamento dos Dados:

```
load("trabalhosalarios.RData")
library(plyr)
library(readr)
library(dplyr)
library(caret)
library(ggplot2)
library(repr)
library(glmnet)

# Padronizar os resultados
set.seed(75)

```

Importamos a biblioteca glmnet (utilizada para as regressões Ridge, Lasso e Elastic Net) e configuramos a seed para o número fixo de '75', para garantir a replicabilidade e consistência da amostragem dos dados entre cada modelo. Ainda neste passo, a coluna earns também é removida do dataset, pois não será utilizada na análise.

```
# Passando o dataset em uma variavel para ficar mais facil a chamada
dat <- trabalhosalarios

#Tirando a coluna earns pois não será utilizada
dat$earns <- NULL
```

2. Rotina Genérica

Existem variáveis e dados em comum, acessados e utilizados pelos modelos em questão, que, portanto, devem ser normalizados (no caso dos valores não binários) e indexados em um dataset:

```
# Criar um indice e particionar o dataset em 80:20
index = sample(1:nrow(dat),0.8*nrow(dat))
train = dat[index,]
```

```r
test = dat[-index,]

# Dimensão das bases, ambas com 17 colunas
dim(train) # 2059 amostras
dim(test) # 515 amostras

# Colunas que necessitam de padronização e não são binárias exceto lwage
cols = c('husage', 'husearns', 'huseduc', 'hushrs',
         'age', 'educ', 'exper', 'lwage')

# Padronizando a base de treinamento e teste
pre_proc_val <- preProcess(train[,cols],
                           method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

```

```
> dim(train)  # 2059 amostras
[1] 5059  16
> dim(test)   # 515 amostras
1] 515   16
```

Também se instancia uma matriz, que logo adiante será usada para predição (com base nos valores do trabalho e aplicando logaritmo neperiano):

```r
# Criando o dataframe para predição no futuro
husage_norm <- (40-pre_proc_val[["mean"]][["husage"]])/
  pre_proc_val[["std"]][["husage"]]
husearns_norm <- (600-pre_proc_val[["mean"]][["husearns"]])/
  pre_proc_val[["std"]][["husearns"]]
huseduc_norm <- (13-pre_proc_val[["mean"]][["huseduc"]])/
  pre_proc_val[["std"]][["huseduc"]]
hushrs_norm <- (40-pre_proc_val[["mean"]][["hushrs"]])/
  pre_proc_val[["std"]][["hushrs"]]
age_norm <- (38-pre_proc_val[["mean"]][["age"]])/
  pre_proc_val[["std"]][["age"]]
educ_norm <- (13-pre_proc_val[["mean"]][["educ"]])/
  pre_proc_val[["std"]][["educ"]]
exper_norm <- (18-pre_proc_val[["mean"]][["exper"]])/
  pre_proc_val[["std"]][["exper"]]

our_pred <- as.matrix(data.frame(husage=husage_norm,
                                 husunion=0,
                                 husearns=husearns_norm,
                                 huseduc=huseduc_norm,
                                 husblck=1,
                                 hushisp=0,
                                 hushrs=hushrs_norm,
                                 kidge6=1,
                                 age=age_norm,
                                 black=0,
                                 educ=educ_norm,
                                 hispanic=1,
                                 union=0,
```

```
                                  exper=exper_norm,
                                  kidlt6=1))
```

	husage	husunion	husearns	huseduc	husblck	hushisp	hushrs	kidge6	age	black	educ	hispanic	union	exper	kidlt6
1	[illegible]	0	[illegible]	[illegible]	1	0	[illegible]	1	[illegible]	0	[illegible]	1	0	[illegible]	1

Devido à baixa legibilidade da imagem fornecida, a tabela foi manualmente transcrita para garantir precisão e clareza dos dados. Abaixo está a tabela transcrita:

husage	husunion	husearns	huseduc	husblck	hushisp	hushrs	kidpres	age	black	educ	hispanic	union	exper	kidslt6
3.582579	0	3.940915	3.243491	0	0	3.252873	1	3.378238	0	3.243491	1	0	2.930072	1

Essa transcrição foi feita para assegurar que todas as informações da imagem original sejam facilmente acessíveis e compreensíveis.

```
# Objeto com os valores do modelo
cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'husblck',
             'hushisp', 'hushrs', 'kidge6', 'age', 'black', 'educ',
             'hispanic', 'union', 'exper', 'kidlt6', 'lwage')

# Estamos interessado em estimar o
# salário-hora da esposa em logaritmo neperiano (lwage)
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+
                       husblck+hushisp+hushrs+kidge6+age+
                       black+educ+hispanic+union+exper+kidlt6,
                     data = dat[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))
```

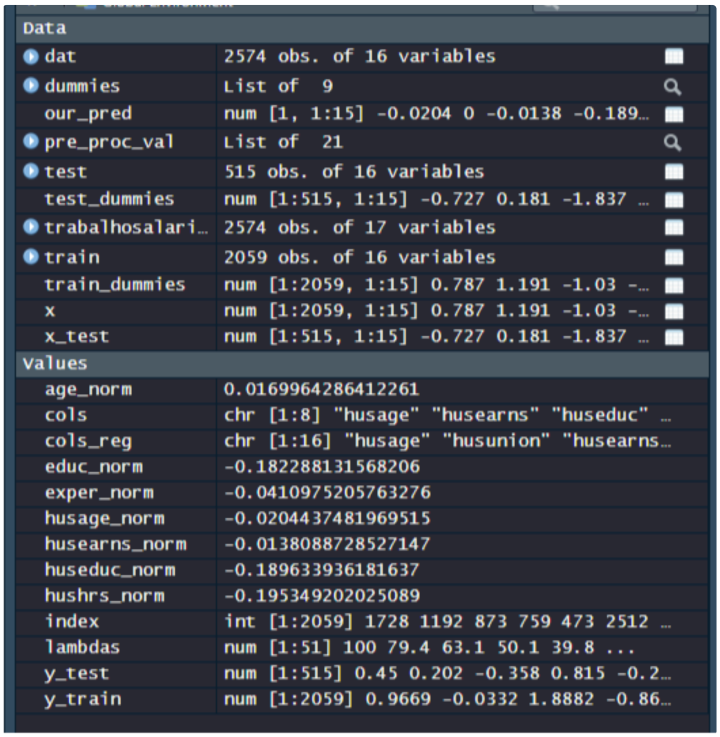
```
> print(dim(train_dummies)); print(dim(test_dummies))

1] 2059    15

[1] 515     15
```

Feito isso, são gravados os valores para aplicação posterior em todos os modelos aqui sendo aplicados, com exceção de 'lambdas', que só é utilizada em Ridge e Lasso.

```
# Vamos guardar a matriz de dados de treinamento das
# variaveis explicativas para o modelo em um objeto
# chamado "x"
x = as.matrix(train_dummies)

# Vamos guardar o vetor de dados de treinamento da
# variavel dependente para o modelo em um objeto
# chamado "y_train"
y_train = train$lwage

# Vamos guardar a matriz de dados de teste das variaveis
```

```
# explicativas para o modelo em um objeto chamado
# "x_test"
x_test = as.matrix(test_dummies)

# Vamos guardar o vetor de dados de teste da variavel
# dependente para o modelo em um objeto chamado "y_test"
y_test = test$lwage

# Lambda padrao para Ridge e Lasso
lambdas <- 10^seq(2, -3, by = -.1)
```

Preview de como ficou os nossos dados genéricos:



Data	
dat	2574 obs. of 16 variables
dummies	List of 9
our_pred	num [1, 1:15] -0.0204 0 -0.0138 -0.189…
pre_proc_val	List of 21
test	515 obs. of 16 variables
test_dummies	num [1:515, 1:15] -0.727 0.181 -1.837 …
trabalhosalari…	2574 obs. of 17 variables
train	2059 obs. of 16 variables
train_dummies	num [1:2059, 1:15] 0.787 1.191 -1.03 -…
x	num [1:2059, 1:15] 0.787 1.191 -1.03 -…
x_test	num [1:515, 1:15] -0.727 0.181 -1.837 …
Values	
age_norm	0.0169964286412261
cols	chr [1:8] "husage" "husearns" "huseduc" …
cols_reg	chr [1:16] "husage" "husunion" "husearns…
educ_norm	-0.182288131568206
exper_norm	-0.0410975205763276
husage_norm	-0.0204437481969515
husearns_norm	-0.0138088728527147
huseduc_norm	-0.189633936181637
hushrs_norm	-0.195349202025089
index	int [1:2059] 1728 1192 873 759 473 2512 …
lambdas	num [1:51] 100 79.4 63.1 50.1 39.8 ...
y_test	num [1:515] 0.45 0.202 -0.358 0.815 -0.2…
y_train	num [1:2059] 0.9669 -0.0332 1.8882 -0.86…

3. Regressão Ridge

Calculando o lambda:

```
# Calculando o lambda:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
                        lambda = lambdas)
# Vamos ver qual o lambda otimo
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge
```

```
> best_lambda_ridge

[1] 0.06309573
```

Estimativa de coeficientes:

```
# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
```

```
                   family = 'gaussian',
                   lambda = best_lambda_ridge)

# Vamos ver o resultado (valores) da estimativa
# (coeficientes)
ridge_reg[["beta"]]
```

```
> ridge_reg[["beta"]]

15 x 1 sparse Matrix of class "dgCMatrix"

                          s0

husage           0.01545592

husunion        -0.01943655

husearns         0.10500981

huseduc          0.03877441

husblck          0.02475497

hushisp          0.04068344

hushrs          -0.03665247

kid6ge          -0.07602030

age              0.01699643

black           -0.02524068

educ             0.15938866

educ             0.15938866

hispanic        -0.02511423

union            0.21152823

exper           -0.01710070

kidlt6          -0.02114730
```

Calculando o RMSE e R² do modelo em Ridge:

```
# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
ridge_eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

  # As metricas de performace do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

# Predicao e avaliacao nos dados de treinamento:
```

```
ridge_predictions_train <- predict(ridge_reg,
                                   s = best_lambda_ridge,
                                   newx = x)

# As metricas da base de treinamento sao:
ridge_eval_results(y_train, ridge_predictions_train, train)

# Predicao e avaliacao nos dados de teste:
ridge_predictions_test <- predict(ridge_reg,
                                  s = best_lambda_ridge,
                                  newx = x_test)
```

```
> ridge_eval_results(y_train, ridge_predictions_train, train)
      RMSE  Rsquare
1 0.8459544 0.2840134

> # Predicao e avaliacao nos dados de teste:
> ridge_predictions_test <- predict(ridge_reg,
+                                   s = best_lambda_ridge,
+                                   newx = x_test)
> # As metricas da base de teste sao:
      RMSE  Rsquare
1 0.8692358 0.2836009
```

Fazendo a predição em Ridge:

```
# Fazendo a predicao:
predict_our_ridge <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = our_pred)
# Normalizando o resultado:
lwage_pred_ridge=(predict_our_ridge*
                    pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]

# Antilog no resultado:
ridge_lwage <- exp(lwage_pred_ridge)
```

```
> predict_our_ridge
          s1
[1,] -0.2389061
> # O resultado normalizado é:
> lwage_pred_ridge
         s1
[1,] 2.073597
```

```
> # O salário hora da esposa é (antilog logaritmo neperiano)

> ridge_lwage

          s1

[1,] 7.953378
```

Valor original predito foi -0.238, após a normalização, voltando a ficar em logaritmo neperiano, foi de $3.24, por fim, após a aplicação do antilog, se tornou $7.95.

O intervalo de confiança para nosso modelo ficou em 7.93 o inferior e 7.98 o superior.

```
# O intervalo de confianca para o nosso exemplo é:
n_ridge <- nrow(train) # tamanho da amostra
m_ridge <- ridge_lwage # valor medio predito
s_ridge <- pre_proc_val[["std"]][["lwage"]] # desvio padrao
dam_ridge <- s_ridge/sqrt(n_ridge) # distribuicao da amostragem da media
CIlwr_ridge <- m_ridge + (qnorm(0.025))*dam_ridge # intervalo inferior
CIupr_ridge <- m_ridge - (qnorm(0.025))*dam_ridge # intervalo superior
```

```
> CIlwr_ridge

            s1

[1,] 7.930951

> CIupr_ridge

            s1

[1,] 7.975806
```

O que significa que o salário pode variar entre 7.93 e 7.97.

4. Regressão Lasso

Como feito na Regressão Ridge, não será necessário recriar variáveis para dataset ou para predição, pois são os mesmos valores, além do próprio lambda, porém será feito um lambda otimizado para essa regressão.

```
# Vamos atribuir alpha = 1 para implementar a regressao
# lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
                        lambda = lambdas,
                        standardize = TRUE, nfolds = 5)

# Vamos guardar o lambda "otimo" em um objeto chamado
# best_lambda_lasso
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso

# Vamos estimar o modelo Lasso
lasso_model <- glmnet(x, y_train, alpha = 1,
                      lambda = best_lambda_lasso,
                      standardize = TRUE)

# Vamos visualizar os coeficientes estimados
```

```
lasso_model[["beta"]]
```

```
> best_lambda_lasso
[1] 0.01258925
> # Vamos estimar o modelo Lasso
> lasso_model <- glmnet(x, y_train, alpha = 1,
+                       lambda = best_lambda_lasso,
+                       standardize = TRUE)
> # Vamos visualizar os coeficientes estimados
> lasso_model[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
                       s0
husage        0.022316956
husunion     -0.008868825
husearns      0.205639157
huseduc       0.052309556
husblck       .
hushisp       .
hushrs       -0.064254431
kid6ge       -0.113650167
age           0.016579810
black         .
educ          0.332126998
hispanic      .
union         0.388032081
exper         .
kidlt6       -0.003836047
```

Agora será realizado a predição do modelo Lasso e encontrar o RMSE e R²:

```
# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
lasso_predictions_train <- predict(lasso_model,
                                   s = best_lambda_lasso,
                                   newx = x)

# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
lasso_eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
```

```
  R_square <- 1 - SSE / SST
  RMSE <- sqrt(SSE / nrow(df))
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

# As metricas da base de treinamento sao:
lasso_eval_results(y_train, lasso_predictions_train, train)
```

```
# As metricas da base de treinamento sao:

> lasso_eval_results(y_train, lasso_predictions_train, train)

       RMSE   Rsquare

1 0.8464903 0.2831059

# Vamos fazer as predicoes na base de teste

lasso_predictions_test <- predict(lasso_model,

                                  s = best_lambda_lasso,

                                  newx = x_test)


# As metricas da base de teste sao:

lasso_eval_results(y_test, lasso_predictions_test, test)


# As metricas da base de teste sao:

> lasso_eval_results(y_test, lasso_predictions_test, test)

       RMSE   Rsquare

1 0.870381 0.281217
```

Em comparação a Regressão Ridge, o RMSE e R² também apresentou resultados baixos, explicando que o modelo não está tão bem otimizado.

Vamos realizar a predição com os dados montados:

```
# Vamos para a predicao
predict_our_lasso <- predict(lasso_model,
                             s = best_lambda_lasso,
                             newx = our_pred)

# Novamente, o resultado esta padronizado, nos temos de
# converte-lo para valor compativel com o dataset original
lwage_pred_lasso=(predict_our_lasso*
                   pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]

# Antilog no resultado:
lasso_lwage <- exp(lwage_pred_lasso)
```

```
> # O resultado da predicao é:
> predict_our_lasso
          s1
[1,] -0.1947567
> # O salário hora da esposa é (antilog logaritmo neperiano)
> lasso_lwage
          s1
[1,] 8.137804
```

Valor original predito foi -0.19, após a normalização, voltando a ficar em logaritmo neperiano, foi de $2.10, por fim, após a aplicação do antilog, se tornou $8.14.

Agora calculando o intervalo de confiança:

```
# Vamos criar o intervalo de confianca para o nosso
# exemplo
n_lasso <- nrow(train)
m_lasso <- lasso_lwage
s_lasso <- pre_proc_val[["std"]][["lwage"]]
dam_lasso <- s_lasso/sqrt(n_lasso)
CIlwr_lasso <- m_lasso + (qnorm(0.025))*dam_lasso
CIupr_lasso <- m_lasso - (qnorm(0.025))*dam_lasso
```

```
> CIlwr_lasso
            s1
[1,] 8.115377
> CIupr_lasso
            s1
[1,] 8.160231
```

O que significa que o salário pode variar entre 8.12 e 8.16 na Regressão Lasso.

5. Regressão ElasticNet

```
#############################################################
#                     REGRESSAO ELASTICNET                  #
#############################################################

train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)

# Vamos treinar o modelo
elastic_reg <- train(lwage~husage+husunion+husearns+huseduc+
```

```
                       husblck+hushisp+hushrs+kidge6+age+
                       black+educ+hispanic+union+exper+kidlt6,
                     data = train,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)

# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

# E os parametros sao:
elastic_reg[["finalModel"]][["beta"]]
```

Melhor alpha escolhido:

```
> # O melhor parametro alpha escolhido eh:

> elastic_reg$bestTune

      alpha     lambda

6 0.5594109 0.02112781

> # E os parametros sao:

> elastic_reg[["finalModel"]][["beta"]]

15 x 75 sparse Matrix of class "dgCMatrix"

[ suppressing 75 column names ‘s0’, ‘s1’, ‘s2’, ‘s3’, ‘s4’, ‘s5’, ‘s6’, ‘s7’, ‘s8’, ‘s9’, ‘s10’, ‘s11’, ‘s12’,
‘s13’, ‘s14’, ‘s15’, ‘s16’, ‘s17’, ‘s18’, ‘s19’, ‘s20’, ‘s21’, ‘s22’, ‘s23’, ‘s24’,

‘s25’, ‘s26’, ‘s27’, ‘s28’, ‘s29’, ‘s30’, ‘s31’, ‘s32’, ‘s33’, ‘s34’, ‘s35’, ‘s36’, ‘s37’, ‘s38’, ‘s39’,
‘s40’, ‘s41’, ‘s42’, ‘s43’, ‘s44’, ‘s45’, ‘s46’, ‘s47’, ‘s48’, ‘s49’, ‘s50’, ‘s51’, ‘s52’,

‘s53’, ‘s54’, ‘s55’, ‘s56’, ‘s57’, ‘s58’, ‘s59’, ‘s60’, ‘s61’, ‘s62’, ‘s63’, ‘s64’, ‘s65’, ‘s66’, ‘s67’,
‘s68’, ‘s69’, ‘s70’, ‘s71’, ‘s72’, ‘s73’, ‘s74’]
```

Fazendo a predição e adquirindo os valores de RMSE e R^2:

```
> en_eval_results(y_train, en_predictions_train, train)

       RMSE  Rsquare

1 0.8464912 0.2831044

> # Vamos fazer as predicoes na base de teste

> en_predictions_test <- predict(elastic_reg, x_test)

> # As metricas de performance na base de teste sao:

> en_eval_results(y_test, en_predictions_test, test)

       RMSE  Rsquare

1 0.8704495 0.281599
```

Novamente não saíram tão bons.

```
# Vamos fazer a predicao com base nos parametros que
# selecionamos
predict_our_elastic <- predict(elastic_reg,our_pred)

# Novamente, o resultado eh padronizado, nos temos que
# reverte-lo para o nivel dos valores originais do
# dataset, vamos fazer isso:
lwage_pred_elastic=(predict_our_elastic*
                     pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]

elastic_lwage <- exp(lwage_pred_elastic)

# O resultado da predicao eh:
predict_our_elastic

# O resultado normalizado eh:
lwage_pred_elastic

# O salário hora da esposa é (antilog logaritmo neperiano)
elastic_lwage
```

Valor original predito foi -0.19, após a normalização, voltando a ficar em logaritmo neperiano, foi de $2.10, por fim, após a aplicação do antilog, se tornou $8.13, bem semelhante ao Lasso.

Para os intervalos de confiança:

```
# Vamos criar o intervalo de confianca para o nosso
# exemplo
n_elastic <- nrow(train)
m_elastic <- elastic_lwage
s_elastic <- pre_proc_val[["std"]][["lwage"]]
dam_elastic <- s_elastic/sqrt(n_elastic)
CIlwr_elastic <- m_elastic + (qnorm(0.025))*dam_elastic
CIupr_elastic <- m_elastic - (qnorm(0.025))*dam_elastic
```

```
> CIlwr_elastic
[1] 8.107046
> CIupr_elastic
[1] 8.151901
```

```
# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

# E os parametros sao:
elastic_reg[["finalModel"]][["beta"]]
```

```
> # O melhor parametro alpha escolhido eh:
> elastic_reg$bestTune
    alpha    lambda
```

```
6 0.5594109 0.02112781

> # E os parametros sao:

> elastic_reg[["finalModel"]][["beta"]]

15 x 75 sparse Matrix of class "dgCMatrix"

[ suppressing 75 column names ‘s0’, ‘s1’, ‘s2’, ‘s3’, ‘s4’, ‘s5’, ‘s6’, ‘s7’, ‘s8’, ‘s9’, ‘s10’, ‘s11’, ‘s12’,
‘s13’, ‘s14’, ‘s15’, ‘s16’, ‘s17’, ‘s18’, ‘s19’, ‘s20’, ‘s21’, ‘s22’, ‘s23’, ‘s24’,

‘s25’, ‘s26’, ‘s27’, ‘s28’, ‘s29’, ‘s30’, ‘s31’, ‘s32’, ‘s33’, ‘s34’, ‘s35’, ‘s36’, ‘s37’, ‘s38’, ‘s39’,
‘s40’, ‘s41’, ‘s42’, ‘s43’, ‘s44’, ‘s45’, ‘s46’, ‘s47’, ‘s48’, ‘s49’, ‘s50’, ‘s51’, ‘s52’,

‘s53’, ‘s54’, ‘s55’, ‘s56’, ‘s57’, ‘s58’, ‘s59’, ‘s60’, ‘s61’, ‘s62’, ‘s63’, ‘s64’, ‘s65’, ‘s66’, ‘s67’,
‘s68’, ‘s69’, ‘s70’, ‘s71’, ‘s72’, ‘s73’, ‘s74’]
```

Fazendo a predição e adquirindo os valores de RMSE e R^2:

```
> en_eval_results(y_train, en_predictions_train, train)

       RMSE    Rsquare

1 0.8464912
```

Valores entre 8.11 e 8.15.

6. Conclusão

Em resumo, os modelos ficaram da seguinte forma:

Modelo	RMSE Treino	R^2 Treino	RMSE Teste	R^2 Teste
Ridge Regression	0.8459544	0.2840134	0.8692358	0.2836009
Lasso Regression	0.8464903	0.2831059	0.870381	0.281712
ElasticNet	0.8464912	0.2831044	0.8704495	0.281599

De forma geral, os modelos sem o atributo ‘earn’ performaram bem mal, porém entre eles, o melhor foi o que usou regressão Ridge.

Outra coisa é que ele também foi o que mostrou o menor salário por hora ($7.94/h) , dando a ideia que talvez o salário por hora correto deve ser muito menor do que o apresentado pelos modelos.