FIFA - Previsao Salarial com Machine Learning

Marco Rodrigues

Bibliotecas

```
library(tidyverse)
library(stringr)
library(tidymodels)
library(leaps)
library(corrplot)
library(car)
library(ggplot2)
library(stringr)
library(readr)
library(knitr)
```

1° Manipulação, ajuste e limpeza dos dados

–>Carregando Dados

```
# Definir a URL do arquivo
file_url <- "https://drive.google.com/uc?export=download&id=1jiWcGsl_tbqK5F0ryUTq48kcDTKWTTuk"

# Ler o arquivo CSV e converter para tibble
df_origin <- read.csv(file_url) %>% as_tibble
```

```
df_origin %>% arrange(desc(Overall)) %>% select(c(Name, Age, Overall, Club,Nationality)) %>%  head(6)
```

```
## # A tibble: 6 x 5
##   Name                Age Overall Club                Nationality
##   <chr>             <int>   <int> <chr>               <chr>
## 1 Cristiano Ronaldo    32      94 Real Madrid CF      Portugal
## 2 L. Messi             30      93 FC Barcelona        Argentina
## 3 Neymar               25      92 Paris Saint-Germain Brazil
## 4 L. Suárez            30      92 FC Barcelona        Uruguay
## 5 M. Neuer             31      92 FC Bayern Munich    Germany
## 6 R. Lewandowski       28      91 FC Bayern Munich    Poland
```

```
str(df_origin)
```

```
## tibble [17,981 x 75] (S3: tbl_df/tbl/data.frame)
##  $ X                   : int [1:17981] 0 1 2 3 4 5 6 7 8 9 ...
##  $ Name                : chr [1:17981] "Cristiano Ronaldo" "L. Messi" "Neymar" "L. Suárez" ...
##  $ Age                 : int [1:17981] 32 30 25 30 31 28 26 26 27 29 ...
```

```
##  $ Photo             : chr [1:17981] "https://cdn.sofifa.org/48/18/players/20801.png" "https://cdn.s
##  $ Nationality       : chr [1:17981] "Portugal" "Argentina" "Brazil" "Uruguay" ...
##  $ Flag              : chr [1:17981] "https://cdn.sofifa.org/flags/38.png" "https://cdn.sofifa.org/f
##  $ Overall           : int [1:17981] 94 93 92 92 92 91 90 90 90 90 ...
##  $ Potential         : int [1:17981] 94 93 94 92 92 91 92 91 90 90 ...
##  $ Club              : chr [1:17981] "Real Madrid CF" "FC Barcelona" "Paris Saint-Germain" "FC Barce
##  $ Club.Logo         : chr [1:17981] "https://cdn.sofifa.org/24/18/teams/243.png" "https://cdn.sofif
##  $ Value             : chr [1:17981] "€95.5M" "€105M" "€123M" "€97M" ...
##  $ Wage              : chr [1:17981] "€565K" "€565K" "€280K" "€510K" ...
##  $ Special           : int [1:17981] 2228 2154 2100 2291 1493 2143 1458 2096 2165 1961 ...
##  $ Acceleration      : chr [1:17981] "89" "92" "94" "88" ...
##  $ Aggression        : chr [1:17981] "63" "48" "56" "78" ...
##  $ Agility           : chr [1:17981] "89" "90" "96" "86" ...
##  $ Balance           : chr [1:17981] "63" "95" "82" "60" ...
##  $ Ball.control      : chr [1:17981] "93" "95" "95" "91" ...
##  $ Composure         : chr [1:17981] "95" "96" "92" "83" ...
##  $ Crossing          : chr [1:17981] "85" "77" "75" "77" ...
##  $ Curve             : chr [1:17981] "81" "89" "81" "86" ...
##  $ Dribbling         : chr [1:17981] "91" "97" "96" "86" ...
##  $ Finishing         : chr [1:17981] "94" "95" "89" "94" ...
##  $ Free.kick.accuracy: chr [1:17981] "76" "90" "84" "84" ...
##  $ GK.diving         : chr [1:17981] "7" "6" "9" "27" ...
##  $ GK.handling       : chr [1:17981] "11" "11" "9" "25" ...
##  $ GK.kicking        : chr [1:17981] "15" "15" "15" "31" ...
##  $ GK.positioning    : chr [1:17981] "14" "14" "15" "33" ...
##  $ GK.reflexes       : chr [1:17981] "11" "8" "11" "37" ...
##  $ Heading.accuracy  : chr [1:17981] "88" "71" "62" "77" ...
##  $ Interceptions     : chr [1:17981] "29" "22" "36" "41" ...
##  $ Jumping           : chr [1:17981] "95" "68" "61" "69" ...
##  $ Long.passing      : chr [1:17981] "77" "87" "75" "64" ...
##  $ Long.shots        : chr [1:17981] "92" "88" "77" "86" ...
##  $ Marking           : chr [1:17981] "22" "13" "21" "30" ...
##  $ Penalties         : chr [1:17981] "85" "74" "81" "85" ...
##  $ Positioning       : chr [1:17981] "95" "93" "90" "92" ...
##  $ Reactions         : chr [1:17981] "96" "95" "88" "93" ...
##  $ Short.passing     : chr [1:17981] "83" "88" "81" "83" ...
##  $ Shot.power        : chr [1:17981] "94" "85" "80" "87" ...
##  $ Sliding.tackle    : chr [1:17981] "23" "26" "33" "38" ...
##  $ Sprint.speed      : chr [1:17981] "91" "87" "90" "77" ...
##  $ Stamina           : chr [1:17981] "92" "73" "78" "89" ...
##  $ Standing.tackle   : chr [1:17981] "31" "28" "24" "45" ...
##  $ Strength          : chr [1:17981] "80" "59" "53" "80" ...
##  $ Vision            : chr [1:17981] "85" "90" "80" "84" ...
##  $ Volleys           : chr [1:17981] "88" "85" "83" "88" ...
##  $ CAM               : num [1:17981] 89 92 88 87 NA 84 NA 88 83 81 ...
##  $ CB                : num [1:17981] 53 45 46 58 NA 57 NA 47 72 46 ...
##  $ CDM               : num [1:17981] 62 59 59 65 NA 62 NA 61 82 52 ...
##  $ CF                : num [1:17981] 91 92 88 88 NA 87 NA 87 81 84 ...
##  $ CM                : num [1:17981] 82 84 79 80 NA 78 NA 81 87 71 ...
##  $ ID                : int [1:17981] 20801 158023 190871 176580 167495 188545 193080 183277 182521 1
##  $ LAM               : num [1:17981] 89 92 88 87 NA 84 NA 88 83 81 ...
##  $ LB                : num [1:17981] 61 57 59 64 NA 58 NA 59 76 51 ...
##  $ LCB               : num [1:17981] 53 45 46 58 NA 57 NA 47 72 46 ...
##  $ LCM               : num [1:17981] 82 84 79 80 NA 78 NA 81 87 71 ...
```

```
##  $ LDM                : num [1:17981] 62 59 59 65 NA 62 NA 61 82 52 ...
##  $ LF                 : num [1:17981] 91 92 88 88 NA 87 NA 87 81 84 ...
##  $ LM                 : num [1:17981] 89 90 87 85 NA 82 NA 87 81 79 ...
##  $ LS                 : num [1:17981] 92 88 84 88 NA 88 NA 82 77 87 ...
##  $ LW                 : num [1:17981] 91 91 89 87 NA 84 NA 88 80 82 ...
##  $ LWB                : num [1:17981] 66 62 64 68 NA 61 NA 64 78 55 ...
##  $ Preferred.Positions: chr [1:17981] "ST LW " "RW " "LW " "ST " ...
##  $ RAM                : num [1:17981] 89 92 88 87 NA 84 NA 88 83 81 ...
##  $ RB                 : num [1:17981] 61 57 59 64 NA 58 NA 59 76 51 ...
##  $ RCB                : num [1:17981] 53 45 46 58 NA 57 NA 47 72 46 ...
##  $ RCM                : num [1:17981] 82 84 79 80 NA 78 NA 81 87 71 ...
##  $ RDM                : num [1:17981] 62 59 59 65 NA 62 NA 61 82 52 ...
##  $ RF                 : num [1:17981] 91 92 88 88 NA 87 NA 87 81 84 ...
##  $ RM                 : num [1:17981] 89 90 87 85 NA 82 NA 87 81 79 ...
##  $ RS                 : num [1:17981] 92 88 84 88 NA 88 NA 82 77 87 ...
##  $ RW                 : num [1:17981] 91 91 89 87 NA 84 NA 88 80 82 ...
##  $ RWB                : num [1:17981] 66 62 64 68 NA 61 NA 64 78 55 ...
##  $ ST                 : num [1:17981] 92 88 84 88 NA 88 NA 82 77 87 ...
```

—> Ajuste dos dados para analise

A partir da análise de variáveis e dos dados gerais, podemos ver que os dados não estão otimizados para analise, assim sendo, precisamos fazer algumas manipulações de modo a propiciar e evitar erro nos códigos e gráficos

```
#Testando os valores das variaveis para Value
fatores_textuais_value <- str_extract(df_origin$Value, "[A-Za-z]+") %>% as.factor()
cat(nlevels(fatores_textuais_value),"- ")
```

```
## 2 -
```

```
cat(levels( fatores_textuais_value),"\n")
```

```
## K M
```

```
#Testando para Wage
fatores_textuais_wage <- str_extract(df_origin$Wage, "[A-Za-z]+") %>% as.factor()
cat(nlevels(fatores_textuais_wage),"- ")
```

```
## 1 -
```

```
cat(levels( fatores_textuais_wage),"\n")
```

```
## K
```

Podemos ver que as variáveis Wage (Sálario do jogador) e Value (valor do jogar), não estão em sua forma numérica e sim em entidades monetárias, assim sendo, possuem componentes textuais como o K(10^3) e M(10^6) além do símbolo da moeda EURO, então surge a necessidade de conversão

```r
#retira lacunas sem informação dos dados
df <- df_origin

#função de conversão da notação textual para numeral
parse.valor <- function(x) {
  x <- substring(x, 2)
  if (grepl("M", x) | grepl("K", x)) {

    notacao <- substring(x, nchar(x))
    x <- as.integer(sub(notacao, "", x))
    if (notacao == "K") {
      x <- x * 1000
    } else {
      x <- x * 1000000
    }
  }
  return(x)
}

#aplica a função para todos os itens da coluna Wage e Value
df <- df %>%
  mutate(Value = sapply(Value, parse.valor)) %>%
  mutate(Wage = sapply(Wage, parse.valor)) %>% na.omit()
```

```r
#altera o tipo de dados da variaveis CHR par INT exceto para os valores da var-> colunas
colunas <- c("Name", "Flag", "Photo", "Club.Logo", "Club", "Preferred.Positions", "Nationality")
df <- df %>% mutate_at(vars(-one_of(colunas)), ~as.integer(as.character(.)))
```

2 Análise Exploratória

```r
paleta_cores <- c("#8179AF","#38A3A5","#255056","#2F4858","#A3586D")
#scale_fill_manual(values = paleta_cores)

# Stats Gerais
cat(c("Quantidade de Jogadores:",nrow(df),"\n"))
```

```
## Quantidade de Jogadores: 17981
```

```r
cat(c("Quantidade Características:", ncol(df)))
```

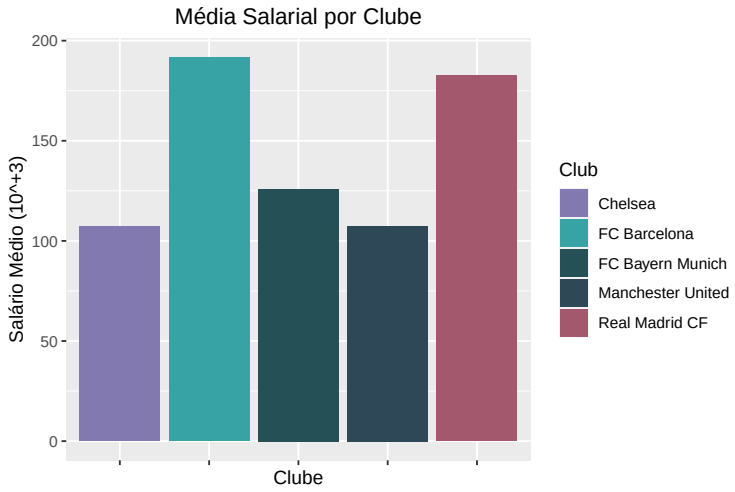
```
## Quantidade Características: 75
```

```r
# Cálculo da média dos valores inteiros
mean_values <- df %>%
  select(where(is.integer)) %>%
  summarise(across(everything(), ~mean(., na.rm = TRUE))) %>%
  pivot_longer(everything(), names_to = "Caracteristica", values_to = "Media")

# Formatação dos números em apenas 2 casas decimais
mean_values$Media <- sprintf("%.2f", as.numeric(mean_values$Media))
```

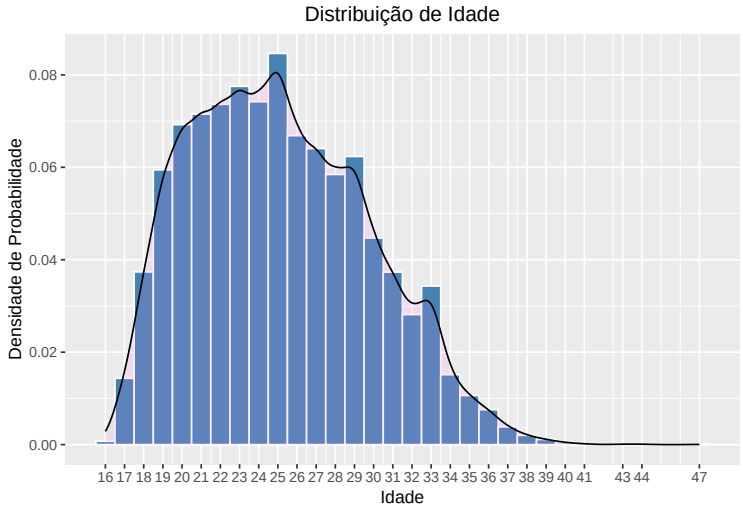
```r
# Quebra da coluna em duas partes
n_linhas <- nrow(mean_values)
mean_values_1 <- mean_values %>%slice(1:round(n_linhas/2))
mean_values_2 <- mean_values %>% slice((round(n_linhas/2)+1):nrow(mean_values))

# Combinando as tibbles em uma única com uma quebra de coluna
combined_mean_values <- cbind(mean_values_1, mean_values_2)

# Exibindo a tibble combinada com kable
combined_mean_values %>% head(10) %>% kable()
```

Caracteristica	Media	Caracteristica	Media
X	8990.00	Sliding.tackle	45.50
Age	25.14	Sprint.speed	64.80
Overall	66.25	Stamina	63.23
Potential	71.19	Standing.tackle	47.35
Value	2259629.05	Strength	65.28
Wage	11546.97	Vision	52.97
Special	1594.10	Volleys	43.20
Acceleration	64.58	CAM	59.25
Aggression	55.79	CB	55.55
Agility	63.33	CDM	56.87

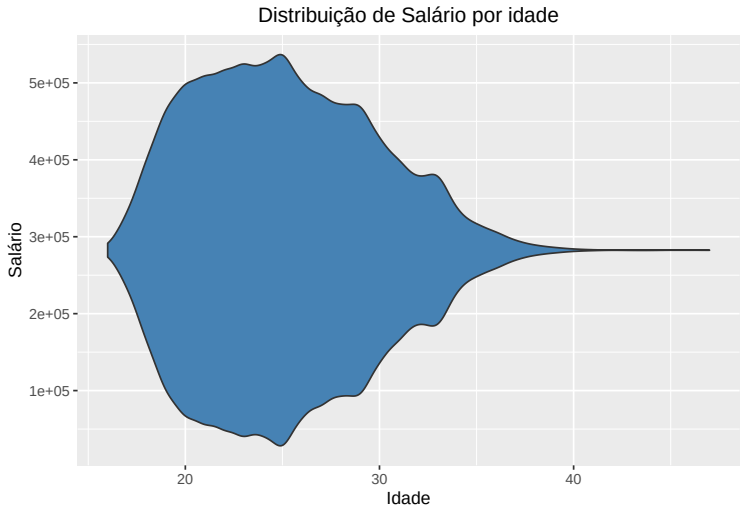
```r
#Verifica a média salarial por Club
df %>% mutate(Club = as.factor(Club)) %>%
  group_by(Club) %>%
  summarise(Salario = sum(Wage),
            Media = mean(Wage)) %>%
  arrange(desc(Salario))%>%
  head(5) %>%
  ggplot(aes(x = Club, y = Media/1000, fill =Club)) +
    geom_bar(stat = "identity") +
    labs(x = "Clube", y = "Salário Médio (10^+3)", title = "Média Salarial por Clube")+
    theme(plot.title = element_text(hjust = 0.5),
        axis.text.x = element_blank()) +
    scale_fill_manual(values = paleta_cores)
```




```r
# Distribuição de Idade
df %>% select(Age) %>%
ggplot(aes(x = Age)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, fill = "steelblue", color = "white") +
  geom_density(alpha = 0.15, fill = "violet") +
  labs(title = "Distribuição de Idade", x = "Idade", y = "Densidade de Probabilidade") +
  scale_x_continuous(breaks = unique(df$Age), labels = unique(df$Age))+
  theme(plot.title = element_text(hjust = 0.5))
```




```r
df %>% select(Age,Wage) %>%
  ggplot(aes(x=Wage, y= Age))+
    geom_violin(fill="steelblue")+
    labs(title = "Distribuição de Salário por idade", x ="Salário", y="Idade")+
    coord_flip()+
    theme(plot.title = element_text(hjust = 0.5))
```



3 Modelos Preditivos

-Seleção e ajuste variáveis escolhidas (desconsiderando fatores fora as características físicas do jogador)

```
# Selecionar as colunas desejadas
df <- df %>%
  select(Age, Overall, Potential, Wage, Special, Acceleration, Aggression, Agility, Balance,
         Ball.control, Composure, Crossing, Curve, Dribbling, Finishing, Positioning, Stamina,
         Interceptions,Strength, Vision, Volleys, Jumping, Penalties, Shot.power, Sprint.speed,
         Heading.accuracy, Long.passing, Short.passing) %>% mutate_if(is.character,as.integer) %>%
         na.omit()

# Conjuntos para validação cruzada
cv_folds <- vfold_cv(df, v = 10)
```

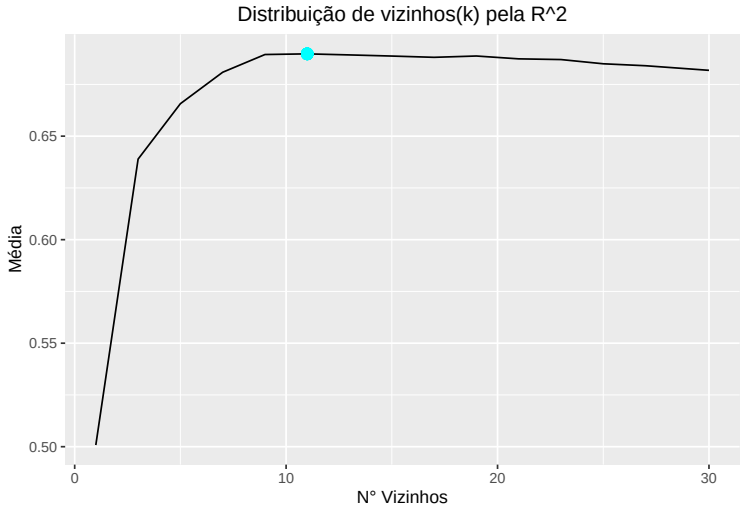
- knn - K-nearest neighbors

```
# Implementa modelo knn modo de regressão
knn.model <- nearest_neighbor(neighbors = tune(),
  weight_func = "rectangular",
  dist_power = 2) %>%
  set_engine("kknn") %>%
  set_mode("regression")

# Define a grade de busca para o hiperparâmetro K
knn_grid <- grid_regular(neighbors(range = c(1,30)), levels = 15)

# Crie uma receita de treinamento
knn_recipe <- recipe(Wage ~ ., data = df) %>%
  step_normalize(all_predictors())

# Busca do hiperparâmetro
knn_values <- tune_grid(knn.model, knn_recipe, resamples = cv_folds, grid = knn_grid)
```

```r
# Armazena o melhor parâmtro
best_knn_model <- select_best(knn_values, metric = "rsq" )

#conjunto de melhores valores para k
knn_values %>% show_best(metric = "rsq") %>% arrange(desc(mean)) %>% kable()
```

neighbors	.metric	.estimator	mean	n	std_err	.config
11	rsq	standard	0.6897528	10	0.0110842	Preprocessor1_Model06
9	rsq	standard	0.6894315	10	0.0107403	Preprocessor1_Model05
13	rsq	standard	0.6892399	10	0.0101372	Preprocessor1_Model07
19	rsq	standard	0.6887158	10	0.0097973	Preprocessor1_Model10
15	rsq	standard	0.6886893	10	0.0104715	Preprocessor1_Model08

```r
knn_values %>%
  collect_metrics() %>%
  filter(.metric == "rsq") %>%
  arrange(desc(mean)) %>%
  ggplot(aes(x= neighbors, y = mean))+
  geom_line()+
  geom_point(aes(x=neighbors[1], mean[1]),
             color = "cyan",
             size = 3)+
  labs(x = "N° Vizinhos", y = "Média", title = "Distribuição de vizinhos(k) pela R^2")+
  theme(plot.title = element_text(hjust = 0.5))
```




```r
knn_wflow <-
  workflow() %>%
  add_model(knn.model) %>%
  add_recipe(knn_recipe)

final_mlp_wflow <-
  knn_wflow %>%
  finalize_workflow(best_knn_model)
print(final_mlp_wflow)
```

```
## == Workflow ====================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ----------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------
## K-Nearest Neighbor Model Specification (regression)
##
## Main Arguments:
##   neighbors = 13
##   weight_func = rectangular
##   dist_power = 2
##
## Computational engine: kknn
```

- Regressão Linear

```
modelo.linear <- lm(Wage ~ ., data = df)
summario <- summary(modelo.linear) %>%
  tidy() %>%
  filter(p.value < 0.001, term != "(Intercept)") %>%
  arrange(p.value)

summario %>% kable()
```

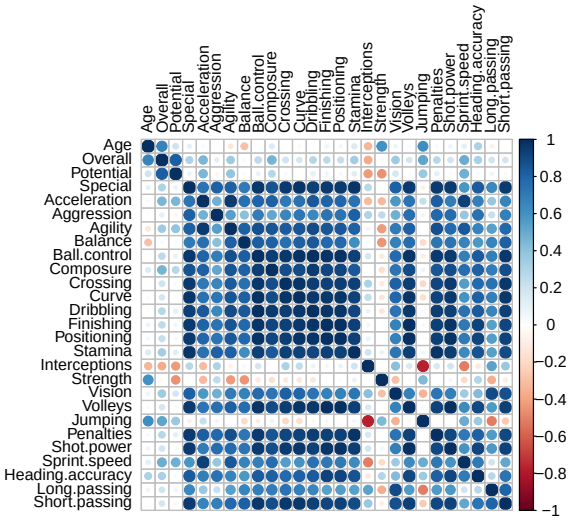
term	estimate	std.error	statistic	p.value
Overall	1720.87260	64.15965	26.821728	0.0000000
Potential	580.95980	56.21955	10.333767	0.0000000
Age	-361.31056	64.33485	-5.616094	0.0000000
Volleys	106.84254	20.23221	5.280813	0.0000001
Composure	88.32764	20.84418	4.237520	0.0000227
Jumping	57.13361	15.33750	3.725093	0.0001959
Penalties	67.26675	18.97058	3.545845	0.0003924

```
#Variance inflation factor
coef.inf <- vif(modelo.linear)

tibbel_vif <- tibble(caracteristicas = names(coef.inf), vif = coef.inf) %>%
  arrange(desc(vif))
high_corr <- sum(head(tibbel_vif$caracteristicas,10) %in%  summario$term)
cat(high_corr)
```

```
## 1
```

Efetuando uma regressão simples sem parâmetros de penalização, seleção de variáveis e outros métodos, podemos ver que pelos menos 3 variáveis com alta relação à resposta possuem alta colinearidade.

```r
matrix_corr <- df %>% select(all_of(names(coef.inf))) %>% head(10) %>% cor()

# Plotar a matriz de colinearidade
corrplot(matrix_corr, type = "full", method = "circle",
         tl.col = "black", tl.srt = 90, tl.cex = 0.8,
         addrect = 3, rect.col = "white", rect.lwd = 2, rect.lty = "dashed")
```



Devido a alta colinearidade devemos utilizar de algum método para reduzir tal interferência nos dados, para este estudo foi utilizado o método elastic net, tal qual está entre a penalização mais abrangente as características RIDGE e o LASSO uma mais restritiva.

```r
# Implementa modelo regressão
elastic_net_model <- linear_reg(penalty = tune(),
                        mixture = tune()) %>%
  set_engine("glmnet")

# Define a grade de busca para o hiperparâmetro a penalidade (lambda) e mixture (seleção de variáveis)
elastic_grid <- grid_regular(penalty(), mixture(range=c(0.1,0.9)), levels = 10)

# Crie uma receita de treinamento
elastic_recipe <- recipe(Wage ~ ., data = df) %>%
  step_normalize(all_predictors())

# Busca do hiperparâmetro
elastic_values <- tune_grid(elastic_net_model, elastic_recipe, resamples = cv_folds, grid = elastic_gri

print(show_best(elastic_values,metric = "rsq"))
```

```
## # A tibble: 5 x 8
##         penalty mixture .metric .estimator  mean     n std_err .config
##           <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.0000000001      0.1 rsq     standard   0.388    10 0.00796 Prepro~
## 2 0.00000000129     0.1 rsq     standard   0.388    10 0.00796 Prepro~
## 3 0.0000000167      0.1 rsq     standard   0.388    10 0.00796 Prepro~
## 4 0.000000215       0.1 rsq     standard   0.388    10 0.00796 Prepro~
## 5 0.00000278        0.1 rsq     standard   0.388    10 0.00796 Prepro~
```

```r
# Armazena o melhor parâmtro
best_ln_model <- select_best(elastic_values, metric = "rsq" )

#conjunto de melhores valores para a penalidade e mixture
elastic_values %>% show_best(metric = "rsq") %>% arrange(desc(mean)) %>% kable()
```

penalty	mixture	.metric	.estimator	mean	n	std_err	.config
0.0e+00	0.1	rsq	standard	0.3875497	10	0.0079612	Preprocessor1_Model001
0.0e+00	0.1	rsq	standard	0.3875497	10	0.0079612	Preprocessor1_Model002
0.0e+00	0.1	rsq	standard	0.3875497	10	0.0079612	Preprocessor1_Model003
2.0e-07	0.1	rsq	standard	0.3875497	10	0.0079612	Preprocessor1_Model004
2.8e-06	0.1	rsq	standard	0.3875497	10	0.0079612	Preprocessor1_Model005

Podemos ver agora quais os coeficientes betas que foram escolhidos pelo modelo (fator de interferência na resposta por caracteristica)

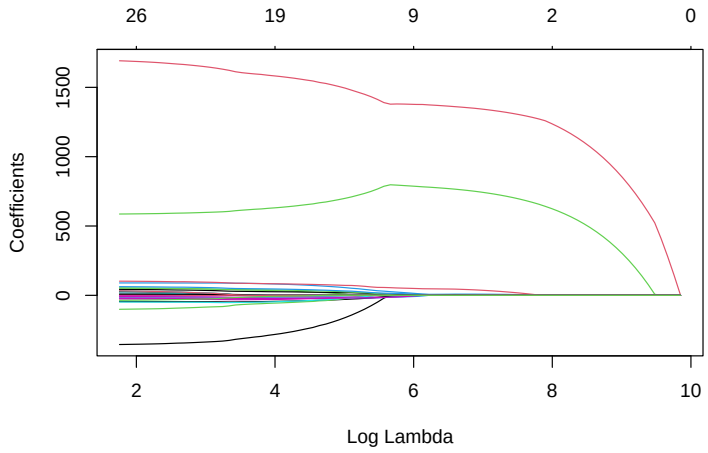
```r
melhor_penalty = best_ln_model$penalty
melhor_mixture = best_ln_model$mixture

lin.fit <- linear_reg( penalty = melhor_penalty, mixture = melhor_mixture) %>%
  set_engine("glmnet") %>%
  fit( Wage ~ . , data = df )

# Extraindo os coeficientes beta do ajuste
lin.fit %>%
  pluck("fit") %>%
  coef(s = melhor_penalty)
```

```
## 28 x 1 sparse Matrix of class "dgCMatrix"
##                               s1
## (Intercept)      -1.301304e+05
## Age              -3.464922e+02
## Overall           1.680175e+03
## Potential         5.940222e+02
## Special          -7.587615e+00
## Acceleration     -1.682427e+01
## Aggression       -9.869392e+00
## Agility          -4.525876e+01
## Balance           1.883799e+01
## Ball.control     -9.889392e+01
## Composure         8.977934e+01
## Crossing          1.613559e+01
## Curve             .
## Dribbling         2.850634e+01
## Finishing         3.720726e+00
## Positioning       5.312834e+01
## Stamina          -2.159185e+01
## Interceptions     3.126703e+01
## Strength         -2.590686e+01
## Vision            4.375860e+01
## Volleys           1.025712e+02
```

```
## Jumping            5.227084e+01
## Penalties          6.270707e+01
## Shot.power        -4.855109e+01
## Sprint.speed      -2.036393e+01
## Heading.accuracy   9.169809e+00
## Long.passing       3.375234e+01
## Short.passing     -3.871830e+01
```

E o gráfico de decaimento dos coeficiente perante o aumento da penalização (> penalty <caracteristicas importam para resposta)

```
# gráfico do decaimento dos coeficientes betas
plot( lin.fit %>% pluck("fit"), xvar = "lambda")
```




```
ln_wflow <-
  workflow() %>%
  add_model(elastic_net_model) %>%
  add_recipe(elastic_recipe)

final_ln_model <-
  ln_wflow %>%
  finalize_workflow(best_ln_model)
print(final_ln_model)
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 1 Recipe Step
##
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
```

```
##
## Main Arguments:
##   penalty = 1e-10
##   mixture = 0.722222222222222
##
## Computational engine: glmnet
```