```
#Exportando as tabelas
matches = read.csv("matches.csv", header = TRUE)
champs = read.csv("champs.csv", header = TRUE)
participants = read.csv("participants.csv", header = TRUE)
stats1 = read.csv("stats1.csv", header = TRUE)
stats2 = read.csv("stats2.csv", header = TRUE)

#Tratamento das tabelas
participants = subset(participants, matchid!=127703 & matchid!=127704 & matchid!=127715
& matchid!=127721 & matchid!=127716)
stats2 = subset(stats2, id!=1272008 & id!=1272009 & id!=1272010 & id!=1272011 &
id!=1272012 & id!=1272014 & id!=1272015 & id!=1272016 & id!=1272017 & id!=1272018)
stats2 = subset(stats2, id!=1272115 & id!=1272116 & id!=1272117 & id!=1272118 &
id!=1272119 & id!=1272120 & id!=1272121 & id!=1272122 & id!=1272123)
stats2 = subset(stats2, id!=1272174)
stats2 = subset(stats2, id!=1272125 & id!=1272126 & id!=1272127 & id!=1272128 &
id!=1272129 & id!=1272130 & id!=1272131 & id!=1272132 & id!=1272133)

#Gerar tabelas
#Tabelas ded partidas
partidas = sqldf::sqldf("SELECT matchid, championid from participants")

#Tabela de Status
stats = rbind(stats1, stats2)

#Tabela de resultados finais, onde 1 é o vencedor e 0 é o perdedor
resultados = sqldf::sqldf("SELECT win from stats")

#Tabela mista entre partida e resultados
geral = cbind(partidas, resultados)

#Desocupar espaço em memória
rm(stats1, stats2, participants)
rm(partidas, stats, resultados)

#Function para separar os jogos pelo ID
coletarDados = function (data, matchid){
  coletar = sqldf::sqldf(paste("SELECT matchid, championid, win from geral where matchid =
", matchid))
  time_azul <- subset(coletar,
as.numeric(rownames(coletar))<=(length(coletar$championid)/2))
  time_verm <- subset(coletar,
as.numeric(rownames(coletar))>(length(coletar$championid)/2))
  final <- data.frame(cbind(bChampionid=time_azul[,2],
rChampionid=time_verm[,2],win=time_azul[,3]))
  return(final)
```

```
}

#Function para verificar o vencedor
verificarVencedor = function(data){
  if(data[1,3]==1){
    valor = 0
    return(valor)
  }else{
    valor = 1
    return(valor)
  }
}

#Function principal que chama todas as outras functions,
#o arquivo vai salvar matchid_ladovencedor
#ladovencedor = 0 -> time azul
#ladovencedor = 1 -> time vermelho
porPartida = function(data){
  for(i in 10:187588){
    coletar = coletarDados(data, i)
    if(is.na(coletar[1,1])){
      print("Sem MatchID")
    }else{
      vencedor = verificarVencedor(coletar)
      coletar = coletar[,-3]
      texto = c(i,"_",vencedor,".csv")
      write.csv(coletar, paste(texto, collapse = ''), row.names = FALSE)
    }
  }
}

#Function para separar os jogos pelo ID
vencedoresPartida = function(data){
  tabela = 1
  for(i in 10:10010){
    coletar = coletarDados(data, i)
    if(is.na(coletar[1,1])){
      print("Sem MatchID")
    }else{
      vencedor = verificarVencedor(coletar)
      tabela = rbind(tabela, vencedor,deparse.level = 0)
    }
  }
  tabela = tabela[-1,]
  return(tabela)
}
```

```
#ffasasasasas
milPartidas = function(data){
  tabela = 1
  for(i in 10:10010){
    coletar = coletarDados(data, i)
    if(is.na(coletar[1,1])){
      print("Sem MatchID")
    }else{
      tabela = rbind(tabela, coletar)
    }
  }
  tabela = tabela[-1,]
  tabela = tabela[,-3]
  return(tabela)
}

#Executar
porPartida(geral)
vencedores = vencedoresPartida(geral)
vencedores2 = matrix(vencedores, ncol = 1)
vencedores = milPartidas(geral)
write.csv(vencedores, "resultadosFinal.csv", row.names = FALSE)
vencedores2 = vencedoresPartida(geral)
```

**Código no Google Colab**

```
import random as rd # numbers random
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import requests
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Convolution2D
from keras.layers.convolutional import MaxPooling2D
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from keras import backend as K
from subprocess import check_output
from numpy.random import seed
seed(15)
```

```
champs = pd.read_csv("champs.csv").values
train = pd.read_csv("partidas_1000.csv").values
y_train = pd.read_csv("resultados.csv").values
characters = pd.read_csv("personagens.csv").values

characters = characters.flatten()

y = np.zeros([1000,2],dtype=np.uint8)

print(y[0])

for i in range(0, len(y_train)):
  if(y_train[i] == 0):
    y[i] = [0,1]
  else:
    y[i] = [1,0]


#y = y_train.flatten()


#print("Y com Flatten =",y)
#print("Y Shape =",y.shape)

def ladoAzul(champs):
  champ = input("Personagem Azul: ")
  champ = idChamp(champs, champ)
  return champ


def ladoVerm(champs):
  champ = input("Personagem Vermelho: ")
  champ = idChamp(champs, champ)
  return champ

def idChamp(champs, champ):
  cont = 0
  for v in [x[0] for x in champs]:
    if v.lower() == champ.lower():
      return champs[cont][1]
    else:
      cont = cont + 1

def montarTime(champs, characters, time, pos, pers):
  time.insert(pos, pers)
  pos = pos + 1
```

```
  while len(time) != 5:
    valor = rd.choices(characters)
    valor = sum(valor)
    if valor not in time:
      time.insert(pos,valor)
      pos = pos + 1

  return time


def alteracaoTime(champs, characters, time, pos):
  pos = pos + 1
  while pos != 5:
    valor = rd.choices(characters)
    valor = sum(valor)
    if valor not in time:
      time[pos] = valor
      pos = pos + 1

  return time

print("Qual lado você está jogando? 1 - Azul / 2 - Vermelho")

numero = int(input("Digite um número: "))

timeAzul = []
timeVerm = []
sorteio = []

sorteio1 = []
sorteio2 = []
sorteio3 = []

contador = 1


timeFinalAzul = []
timeFinalVerm = []

while contador != 4:
  if contador == 1:
    pers1 = ladoAzul(champs)
    c = 0
    while c != 1000:
      timeAzul = []
      timeAzul = montarTime(champs, characters, timeAzul, 0, pers1)
```

```
    c = c + 1
    sorteio1.append(timeAzul)

  print("Sorteio Azul 1 =",sorteio1)
  contador = contador + 1
  timeFinalAzul.insert(0, pers1)

  '''
elif contador == 4:
  sorteio = []
  pers = ladoAzul(champs)
  timeAzul[1] = pers
  c = 0
  while c != 10:
    timeAzul = alteracaoTime(champs, characters, timeAzul, 1)
    c = c + 1
    sorteio.append(timeAzul)

  print("Sorteio Azul 2 =",sorteio)
  contador = contador + 1
  timeFinalAzul.insert(1, pers)

elif contador == 5:
  sorteio = []
  pers = ladoAzul(champs)
  timeAzul[2] = pers
  c = 0
  while c != 10:
    timeAzul = alteracaoTime(champs, characters, timeAzul, 2)
    c = c + 1
    sorteio.append(timeAzul)

  print("Sorteio Azul 3 =",sorteio)
  contador = contador + 1
  timeFinalAzul.insert(2, pers)

elif contador == 8:
  sorteio = []
  pers = ladoAzul(champs)
  timeAzul[3] = pers
  c = 0
  while c != 10:
    timeAzul = alteracaoTime(champs, characters, timeAzul, 3)
    c = c + 1
    sorteio.append(timeAzul)
```

```
    print("Sorteio Azul 4 =",sorteio)
    contador = contador + 1
    timeFinalAzul.insert(3, pers)

  elif contador == 9:
    sorteio = []
    pers = ladoAzul(champs)
    timeAzul[4] = pers
    sorteio.append(timeAzul)

    print("Sorteio Azul 5 =",sorteio)
    contador = contador + 1
    timeFinalAzul.insert(4, pers)
'''

  elif contador == 2:
    #sorteio = []
    pers2 = ladoVerm(champs)
    c = 0
    while c != 1000:
      timeVerm = []
      timeVerm = montarTime(champs, characters, timeVerm, 0, pers2)
      c = c + 1
      sorteio2.append(timeVerm)

    print("Sorteio Verm 1 =",sorteio2)
    contador = contador + 1
    timeFinalVerm.insert(0, pers2)

  elif contador == 3:
    #sorteio = []
    pers3 = ladoVerm(champs)
    c = 0
    while c != 1000:
      timeVerm = [pers2,pers3,0,0,0]
      #timeVerm.insert(0, pers2)
      #timeVerm.insert(1, pers3)
      timeVerm = alteracaoTime(champs, characters, timeVerm, 1)
      c = c + 1
      sorteio3.append(timeVerm)

    print("Sorteio Verm 2 =",sorteio3)
    contador = contador + 1
    timeFinalVerm.insert(1, pers3)

    '''
```

```
  elif contador == 6:
    sorteio = []
    pers = ladoVerm(champs)
    timeVerm[2] = pers
    c = 0
    while c != 10:
      timeVerm = alteracaoTime(champs, characters, timeVerm, 2)
      c = c + 1
      sorteio.append(timeVerm)

    print("Sorteio Verm 3 =",sorteio)
    contador = contador + 1
    timeFinalVerm.insert(2, pers)

  elif contador == 7:
    sorteio = []
    pers = ladoVerm(champs)
    timeVerm[3] = pers
    c = 0
    while c != 10:
      timeVerm = alteracaoTime(champs, characters, timeVerm, 3)
      c = c + 1
      sorteio.append(timeVerm)

    print("Sorteio Verm 4 =",sorteio)
    contador = contador + 1
    timeFinalVerm.insert(3, pers)

  elif contador == 10:
    sorteio = []
    pers = ladoVerm(champs)
    timeVerm[4] = pers
    sorteio.append(timeVerm)

    print("Sorteio Verm 5 =",sorteio)
    contador = contador + 1
    timeFinalVerm.insert(4, pers)


if numero == 1:
  print("Final:", timeFinalAzul)
else:
  print("Final:", timeFinalVerm)
'''
x = 0
tamanho = len(sorteio3)
```

```
vetor = []

while(x < tamanho):
    sorteio2 = sorteio1[x] + sorteio3[x]
    vetor.append(sorteio2)
    x+=1


#print("Sorteio sem Flatten =",sorteio)
print(vetor)
sorteio = np.array(vetor)
sorteio.reshape([1000,10])

print(sorteio)
#sorteio = np.array(sorteio, dtype=np.uint8)
#sorteio = sorteio.flatten()

#print("Sorteio com Flatten =",sorteio)

partida = np.zeros([5,2],dtype=np.uint8)
lista_partidas = []

j = 0
for i in range(0, 5001):
    partida[j][0]= train[i][0]
    partida[j][1]= train[i][1]
    j+=1
    if (j == 5):
        a = partida.flatten()
        lista_partidas.append(a)
        j = 0


#print("lista_partidas =",lista_partidas)


print(timeFinalAzul)
print(timeFinalVerm)
partidas_tensor = np.array(lista_partidas, dtype=np.uint8)

#print("Sorteio Shape =",sorteio.shape)

#print("Partidas_tensor =",partidas_tensor)
#print("Partidas_tensor Shape =",partidas_tensor.shape)

model = Sequential()
```

```
model.add(Dense(256, input_dim=10 , activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
#model.add(Dense(1, activation='sigmoid'))
model.add(Dense(2, activation='sigmoid'))
#model.add(Dense(2, activation='softmax'))


model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(partidas_tensor, y, epochs=60, batch_size=10)

scores = model.evaluate(partidas_tensor, y)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))

#X = np.zeros([5,2],dtype=np.uint8)
#sorteio = np.reshape(sorteio, (10))

predictions = model.predict(sorteio)
print("Predictions Shape", predictions.shape)
#for i in range(0, len(predictions)):
 # print("Sorteio = ", sorteio[i])
  #print(predictions[i])

# round predictions
#rounded = [round(x[0]) for x in predictions]
#print(rounded)
recom = []
resultados = []

predictions.flatten()

t = len(sorteio)
z = 0
ind = 0
while z < t:
  if numero == 1:
    for v in [x[0] for x in predictions]:
      if v == 1.0:
        r = sorteio[ind]
        #print(r)
        recom.append(r)
        #print(v)
      ind = ind + 1
      if ind != 11:
```

```
      z = t

  z = z + 1
 else:
  for v in [x[1] for x in predictions]:
   if v == 1.0:
    r = sorteio[ind]
    #print(r)
    recom.append(r)
    #print(v)
   ind = ind + 1
   if ind != 11:
    z = t

  z = z + 1


print(recom[0],"\n",recom[1],"\n",recom[2],"\n",recom[3],"\n",recom[4],"\n",recom[5],"\n",recom
[6],"\n",recom[7],"\n",recom[8],"\n",recom[9],"\n",recom[10])

pers4 = ladoAzul(champs)
pers5 = ladoAzul(champs)
pers6 = ladoVerm(champs)
pers7 = ladoVerm(champs)
pers8 = ladoAzul(champs)
pers9 = ladoAzul(champs)
pers10 = ladoVerm(champs)

timeFinalAzul.insert(1, pers4)
timeFinalAzul.insert(2, pers5)
timeFinalVerm.insert(2, pers6)
timeFinalVerm.insert(3, pers7)
timeFinalAzul.insert(3, pers8)
timeFinalAzul.insert(4, pers9)
timeFinalVerm.insert(4, pers10)

print("Time Azul final:",timeFinalAzul)

print("Time Verm final:",timeFinalVerm)

times = timeFinalAzul + timeFinalVerm
print("Final",times)
```