



Kaggle's Shelter Animal

Mineração de dados

Universidade Federal do Rio de Janeiro
Mineração de dados

Leonardo Neves da Silva
Ricardo Denílson

Rio de Janeiro
2016

SUMÁRIO

| | |
|--------------------------------------|--------------|
| 1 O DESAFIO | |
| 2 DADOS COLETADOS | |
| 3 DADOS FILTRADOS | |
| 4 SOLUÇÃO ESCOLHIDA | |
| 5 PROBLEMAS ENCONTRADOS | |
| 6 APLICAÇÃO | |
| 7 RESULTADOS | |
| 8 CONSIDERAÇÕES FINAIS | |

O DESAFIO

Nosso desafio envolvia o destino dos animais de um abrigo.

Este abrigo, protege gatos e cachorros que chegaram ali em diversas situações e vindo de várias origens.

Existem 5 destinos possíveis para os cães e gatos deste abrigo: eutanásia, morte natural, adoção, transferência ou retornar ao seu dono.

Tomando em consideração os dados históricos que o abrigo já têm de milhares de animais que por ali já passaram, precisamos prever o percentual para cada um dos 5 destinos finais que cada um dos animais do abrigo.

A partir destes dados, os responsáveis pelo abrigo poderão dar um “empurrãozinho” nos cachorros e gatos que têm a maior probabilidade de ter um fim triste, como a eutanásia.

Tomando em consideração dados como idade, raça, cor, se castrado ou não, precisamos prever qual será o destino destes pequenos animais.

DADOS COLETADOS

O conjunto de dados fornecido pelo abrigo é extenso e representa um total de mais de 26 mil animais que já passaram por este abrigo.

Dentro do **arquivo de treino** para cada um dos animais, tínhamos as seguintes características:

- ID do animal
- Nome (quando houver)
- Data do evento
- **Destino (o que queremos prever)**
- *Subtipo de destino*
- Tipo de animal
- Se é castrado (ou não)
- Idade
- Raça
- Cor

É importante notar que nem sempre todos estes dados estão preenchidos.

Um exemplo é que para alguns animais temos diversas informações a respeito do mesmo, e não temos uma informação básica, como o nome. Julgamos a informação do nome importante, pois ela diz bastante coisa sobre o histórico de um animal.

Um outro exemplo é a raça dos animais. Há uma variação muito grande nas raças informadas. O que muitas das vezes pode gerar uma diversidade excessiva que pode ser melhor organizada, para uma melhor análise.

DADOS FILTRADOS

Com a quantidade de dados que recebemos se tornou de fundamental importância segmentar os dados em características (features) para que pudéssemos utilizar alguma das inúmeras bibliotecas de análise de dados existentes. Além disso aprendemos que transformar os dados de linguagem natural em dados numéricos facilita a taxa de acerto da maioria dos algoritmos.

Logo para analisar o arquivo de **treino** destacamos as seguintes *features*:

Logo de início descartamos a primeira coluna **ID** pois podemos perceber que ela é apenas um identificador único e não tem ligação com a classificação.

Em relação ao **nome** a única feature que pudemos criar foi se o animal tinha um nome ou não. Logo temos uma feature binária

A **data** listada contém o dia, mês, ano, hora, minuto e segundo em que o animal foi classificado.
Ex. No dia 25/12/2013 às 15:01:00 o animal foi adotado.

Desta forma separamos a data nas quatro features que julgamos realmente fazer diferença.

A **hora** do dia.

Percebemos que em determinadas horas ocorrem eventos específicos, como a transferência.

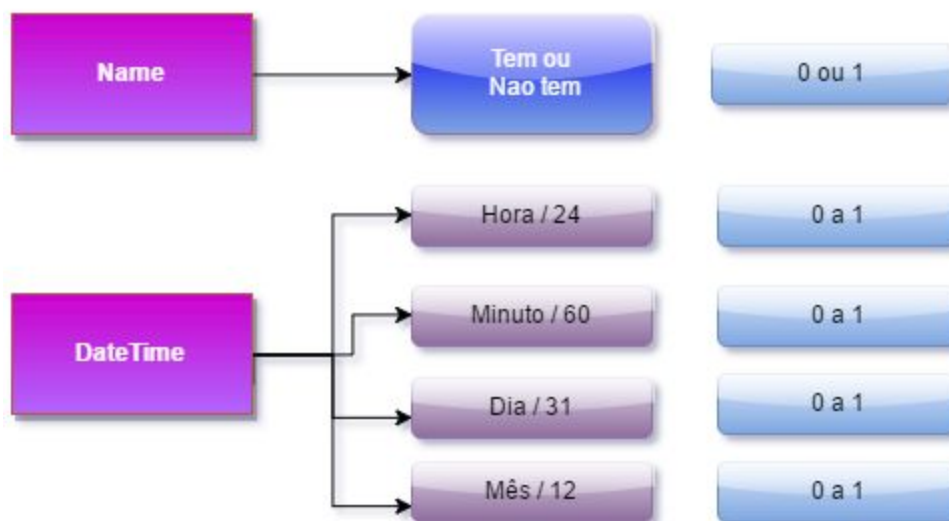
Desta forma separamos as horas em uma faixa de 0 a 1. O valor era criado pela divisão $\text{hora}/24$

Da mesma forma criamos a coluna minuto com o valor de divisão $\text{minuto}/60$.

A característica marcante para incluirmos os minutos foi justamente a mesma da hora, uma vez que se tivéssemos um evento exatamente às 9:00h a chance dele ser uma transferência era bem grande. Enquanto se o evento ocorresse às 09:10 as chances caíam drasticamente.

Transformamos **dia** e **mês** em duas colunas individuais apenas para podermos aproximar o evento do início ou fim do mês ou início ou fim do ano de acordo com a normalização.

$\text{Dia}/31$ normalizado entre 0 e 1 e $\text{mês}/12$ normalizado entre 0 e 1.

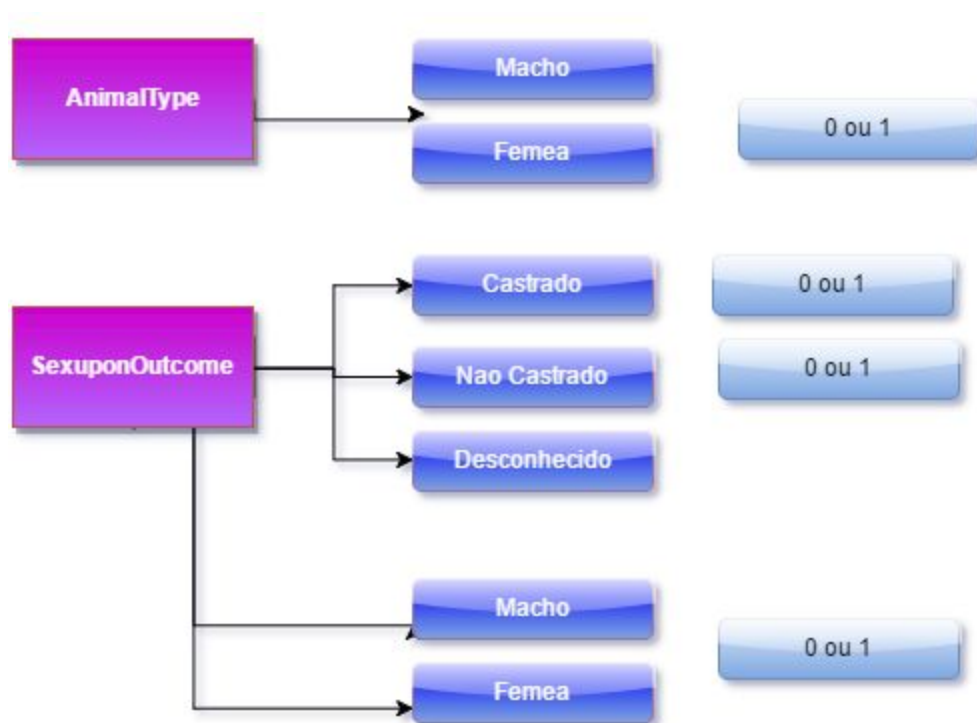


Como **Destino** era o que queríamos avaliar, apenas transformamos os dados em dados numéricos sem alterações.

Porém não vimos utilidade para a informação **subtipo de destino**, uma vez que apenas o destino era nosso objetivo.

Tipo de animal, definimos uma coluna classificadora binária que define se o animal era cachorro ou gato

A informação da coluna SexOutcome continha informações misturadas, definia se o animal era do sexo masculino ou feminino ou se era castrado ou não. Desta forma separamos em duas colunas. **Sexo** e **castração** ambas binárias.



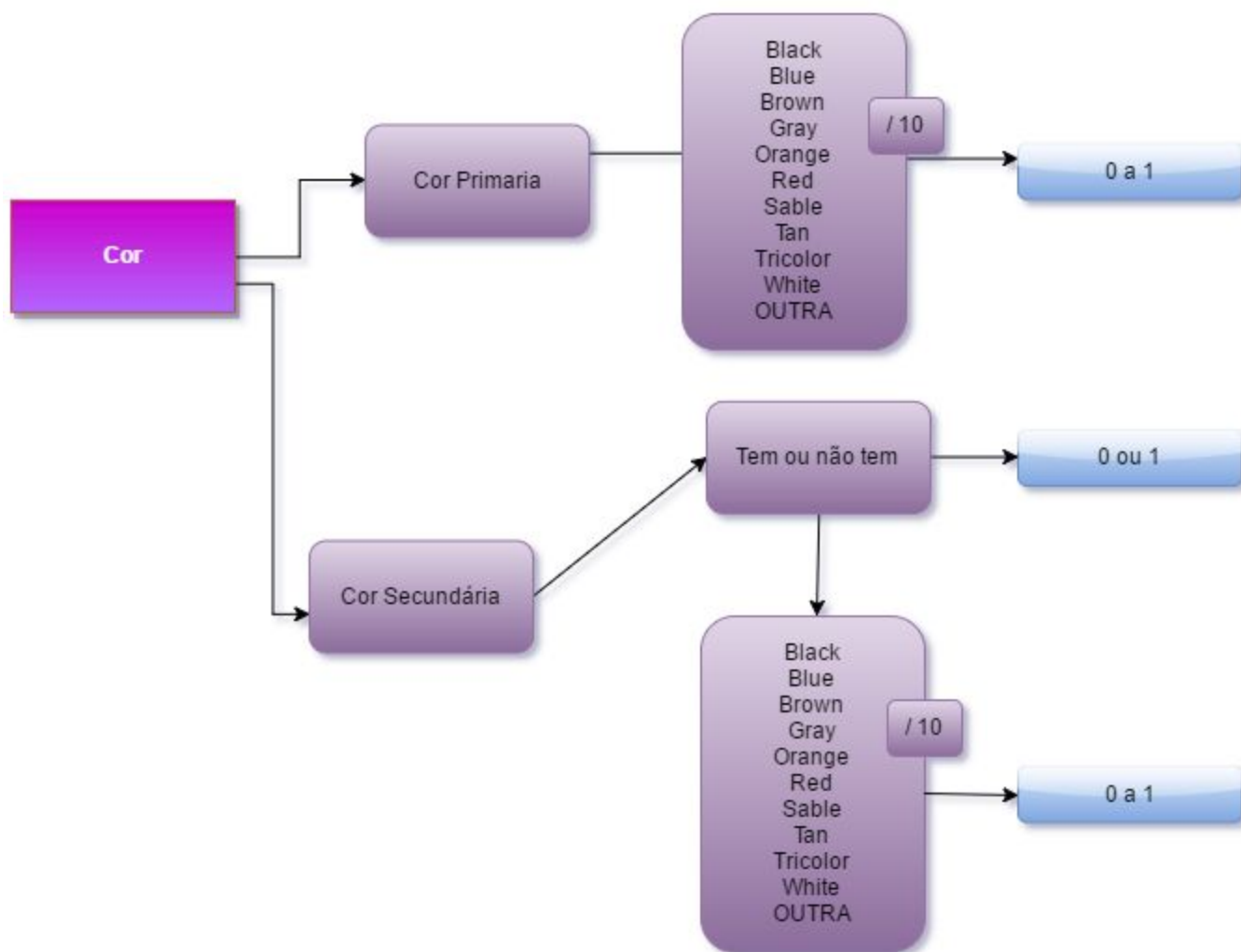
Em **idade** o trabalho que tivemos foi para organizar a classificação, pois os dados vinham no formato de meses, semanas, dias ou anos.

O que fizemos foi converter todos os dados para dias e em seguida classificar em relação ao animal mais velho dos dados. Ficamos com a normalização de 0 a 1 onde zero era o animal recém nascido e 1 o animal mais velho do grupo.

Nas colunas raça e cor, antes de classificarmos analisamos o nosso arquivo e separamos as raças e cores mais populares.

Em **cor** criamos uma coluna em que classificamos o quão popular é aquela cor em relação a demais, criando valores entre 0 e 1, onde 0 são valores pouco populares.

Em seguida criamos uma segunda coluna em que avaliamos se o animal tinha mais de uma cor. caso não tivesse a coluna seria valorada com (0) e caso tivesse classificariamos de acordo com as 10 cores populares que sugerimos, assim como feito na coluna anterior.teríamos um valor entre zero e 1.



Em relação a raça criamos várias colunas.

A primeira definia se o animal era de raça pura ou não (raça/raça ou raça Mix)

Em seguida criamos classificadores binários para testar se determinado animal pertencia a alguma das raças mais populares que foram as seguintes:

Domestic, Longhair, Miniature, Shorthair e Wirehair



Recapitulando:

Dadas 7 características para análise e duas com as saída esperadas:

- Nome (quando houver)
- Data do evento
- **Destino (o que queremos prever)**
- ***Subtipo de destino***
- Tipo de animal
- Se é castrado (ou não)
- Idade
- Raça
- Cor

Transformamos os dados em 19 colunas/features:

- **Nome** (0 ou 1)
- **Hora** (0 a 1)
- **Minuto** (0 a 1)
- **Dia** (0 a 1)
- **Mês** (0 a 1)
- **É dia comemorativo?** (0 ou 1)
- **Tipo (cachorro ou gato)** (0 ou 1)
- **Sexo** (0 ou 1)
- **Castração** (0 ou 1)
- **Idade em dias / idade do animal mais velho** (0 a 1)
- **Raça pura ou não** (0 ou 1)
- **Raça Domestic?** (0 ou 1)
- **Raça Longhair?** (0 ou 1)
- **Raça Sorthair?** (0 ou 1)
- **Raça Wirehair?** (0 ou 1)
- **Raça Miniature?** (0 ou 1)
- **Popularidade da cor1** (0 a 1)
- **Tem uma segunda cor?** (0 ou 1)
- **Popularidade da segunda cor** (0 a 1)

SOLUÇÃO ESCOLHIDA

Chegamos a cogitar a utilização de vários perceptrons. o que não se mostrou uma ideia eficiente. Optamos por construir uma rede neural pois criando a quantidade correta de camadas devemos receber como resultado a matriz de pesos.

Aplicando esta matriz em um exemplo obtemos a sua classificação, ou seja, o quanto um exemplo está próximo de cada classe.

Dentre as implementações de redes neurais que analisamos o módulo **pybrain** se mostrou o que com uma boa documentação, api de fácil manuseio e alguns bons exemplos que seguimos como base.

APLICAÇÃO

Primeiro, separamos as colunas dos dados de treino de acordo com as classificações que determinamos. Com os dados filtrados prontos para uso, criamos o nosso dataset de treino com a função `ClassificationDataset`.

```
TrainData = ClassificationDataSet(len(_x[0]), 1, nb_classes=5)
```

Ao utilizar a função definimos os seguintes parâmetros:

O tamanho de cada linha (quantidade de features do dataset), a quantidade e quantidade de de classes que queremos classificar.

Com o dataset criado o trabalho devemos agrupar ao menos um neurônio por classe. Fazemos isso utilizando a função

```
traindata._convertToOneOfMany( )
```

Com o dataset pronto para uso.

Antes de criar a rede podemos verificar se já foi feito algum treinamento e se já há alguma matriz de pesos criada.

Para carregar uma matriz pré existente utilizamos a função:

```
if os.path.exists('rede_animal.xml'):  
fnn = NetworkReader.readFrom('rede_animal.xml')
```

Caso contrário podemos criar uma nova rede:

```
fnn = buildNetwork(traindata.indim , 5, traindata.outdim, outclass=SoftmaxLayer)
```

Esta nova rede precisa das dimensões do dataset como a quantidade de features e quantidade de exemplos (`traindata.indim`), resultado dos exemplos (`traindata.outdim`), a quantidade de classes/camadas (5) e o método de redução que será utilizado (`softmaxlayer`).

Importante dizer que ao utilizar 5 camadas não estamos utilizando nenhuma camada oculta, utilizando algum número de camadas maior do que 5 significaria estar inserindo camadas ocultas que facilitariam a amortização dos valores da matriz.

Utilizando o dataset e a rede neural podemos fazer a rede treinar, com a função `backprop` que mede o `logloss` a cada época e exibe na tela (`verbose`)

A taxa de correção é o `weightdecay`.

```
trainer = BackpropTrainer( fnn, dataset=traindata, momentum=0.1, verbose=True,  
weightdecay=0.01)
```

```

for i in range(epochs):
    print("Treinando época ", i)
    trainer.trainEpochs( steps )
    NetworkWriter.writeToFile(fnn, 'rede_animal.xml')
    print(" Rede salva em rede_animal.xml")

```

Com a ultima linha de código acima salvamos os valores da rede a cada iteração, o que torna possível interromper a análise a qualquer momento mantendo salvo o treinamento já feito.

Com a rede criada e projetada basta analisar cada exemplo desejado com a função `rede.activate(exemplo)`.

A função retorna um vetor em que cada posição corresponde a probabilidade de exemplo pertencer a cada uma das classes.

```

print("Gerando resultados em animal_output.csv .....")

output = open('animal_output.csv', 'wb')
i=1
for line in open(test_file, 'r'):
    x = ast.literal_eval(line)
    output.write( "{},{},{},{},{},{}\n".
        format(i,fnn.activate( x )[0],fnn.activate( x )[1],
            fnn.activate( x )[2],fnn.activate( x )[3],fnn.activate( x )[4]) )
    i=i+1

```

PROBLEMAS ENCONTRADOS

Após filtrar, definimos e escolhermos quais colunas iríamos utilizar, começamos a encontrar as primeiras dificuldades.

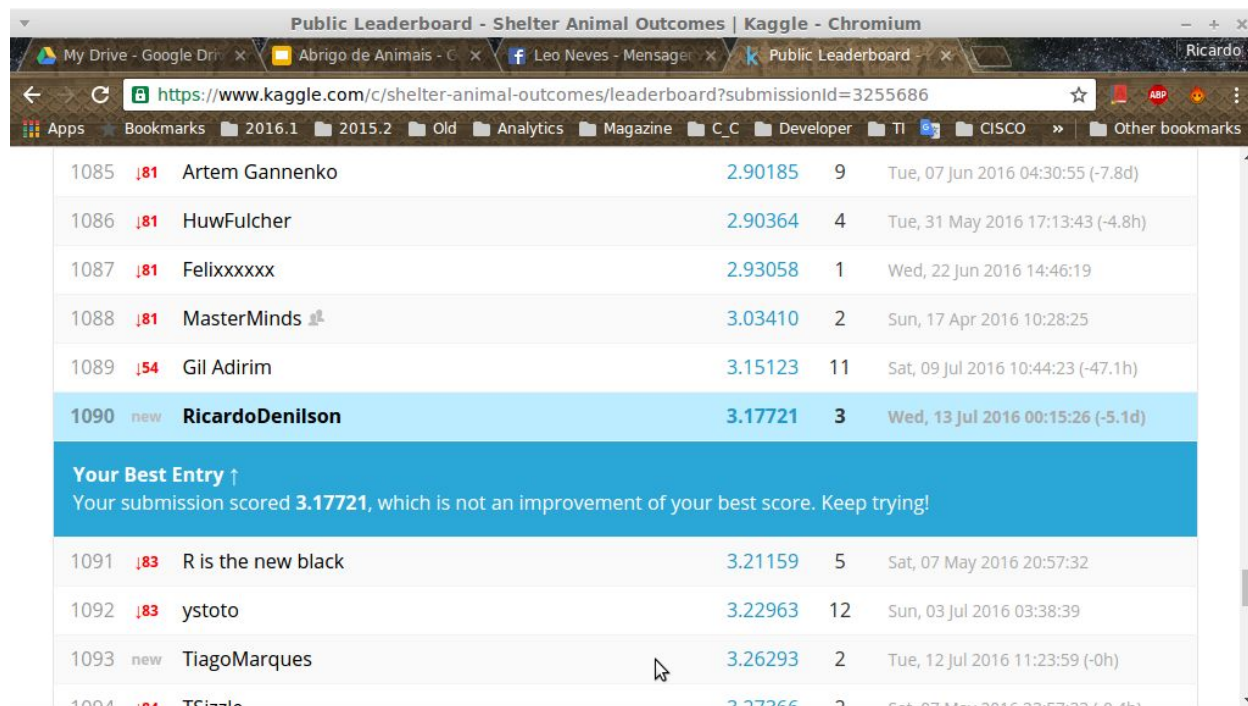
Procuramos uma biblioteca que pudesse nos ajudar em algumas tarefas. Após uma busca exaustiva, encontramos a PyBrain. Que se mostrou perfeita para nossa necessidade.

Pybrain é uma biblioteca que tem uma integração perfeita com o Python. Por ter uma característica modular, o Pybrain é uma biblioteca destinada a implementações para aprendizado de máquina. Por ser flexível e com algoritmos muito eficientes, nosso trabalho avançou muito com o Pybrain.

Uma outra dificuldade que tivemos foi na hora de codificar. Apesar de termos em nossa mente, o que esperávamos, foi pesquisando e descobrindo as inúmeras funcionalidades do Python que nos fez chegar ao resultado final. Tivemos uma evolução nas nossas habilidades com Python ao longo do trabalho.

RESULTADOS

Inicialmente nós obtemos um resultado bem aquém de nossas previsões. Conseguimos nos posicionar na posição **1090**.



| | | | | | |
|--|-----|------------------------|----------------|----------|--|
| 1085 | ↓81 | Artem Gannenکو | 2.90185 | 9 | Tue, 07 Jun 2016 04:30:55 (-7.8d) |
| 1086 | ↓81 | HuwFulcher | 2.90364 | 4 | Tue, 31 May 2016 17:13:43 (-4.8h) |
| 1087 | ↓81 | Felixxxxxx | 2.93058 | 1 | Wed, 22 Jun 2016 14:46:19 |
| 1088 | ↓81 | MasterMinds 1 | 3.03410 | 2 | Sun, 17 Apr 2016 10:28:25 |
| 1089 | ↓54 | Gil Adirim | 3.15123 | 11 | Sat, 09 Jul 2016 10:44:23 (-47.1h) |
| 1090 | new | RicardoDenilson | 3.17721 | 3 | Wed, 13 Jul 2016 00:15:26 (-5.1d) |
| Your Best Entry ↑ Your submission scored 3.17721, which is not an improvement of your best score. Keep trying! | | | | | |
| 1091 | ↓83 | R is the new black | 3.21159 | 5 | Sat, 07 May 2016 20:57:32 |
| 1092 | ↓83 | ystoto | 3.22963 | 12 | Sun, 03 Jul 2016 03:38:39 |
| 1093 | new | TiagoMarques | 3.26293 | 2 | Tue, 12 Jul 2016 11:23:59 (-0h) |
| 1094 | ↓84 | TSizzle | 3.27366 | 7 | Sat, 07 May 2016 22:57:23 (-0.4h) |

Para alcançarmos esta posição inicial, definimos poucas épocas, **5**, e **3** ciclos. O resultado nos surpreendeu. Partimos para uma outra abordagem com mais épocas e mais ciclos.

Desta vez fizemos **15** épocas e **3** ciclos. Nós tivemos uma melhora considerável. Dando um salto para a posição **881**.

| | | | | | |
|--|-----|-----------------------|---------|----|------------------------------------|
| 870 | .53 | Osprey | 1.03623 | 3 | Tue, 14 Jun 2016 13:29:55 (-4.7d) |
| 871 | .53 | SylvesterdeKoning | 1.03623 | 1 | Tue, 28 Jun 2016 15:05:42 |
| 872 | .53 | RichardStanton | 1.03623 | 1 | Sun, 03 Apr 2016 16:52:43 |
| 873 | .53 | Didac Cortiada Rovira | 1.03751 | 1 | Thu, 02 Jun 2016 12:56:08 |
| 874 | .53 | deefers | 1.03799 | 4 | Tue, 19 Apr 2016 21:54:23 |
| 875 | .53 | Climbatize | 1.04147 | 3 | Wed, 18 May 2016 22:29:19 |
| 876 | new | DDerek | 1.04233 | 3 | Thu, 07 Jul 2016 15:52:20 |
| 877 | .54 | Bao baos | 1.05315 | 2 | Mon, 27 Jun 2016 22:36:34 (-0.7h) |
| 878 | .54 | Nicolas Pielawski | 1.05448 | 3 | Sat, 14 May 2016 17:52:14 (-1.8h) |
| 879 | new | Nanun Tr | 1.05859 | 3 | Tue, 05 Jul 2016 20:08:36 (-28.4h) |
| 880 | .21 | DataScience_Bonn | 1.05930 | 8 | Fri, 01 Jul 2016 22:49:04 |
| 881 | .57 | LeoNeves | 1.06108 | 6 | Thu, 07 Jul 2016 23:21:11 |
| Your Best Entry ↑ You improved on your best score by 0.00828. You just moved up 2 positions on the leaderboard. Tweet this! | | | | | |
| 882 | .57 | tommy | 1.06693 | 4 | Mon, 02 May 2016 09:14:44 |
| 883 | new | BrunoAlves | 1.06718 | 3 | Wed, 06 Jul 2016 23:32:36 (-0.4h) |
| 884 | .58 | Amy W | 1.07004 | 1 | Sat, 26 Mar 2016 01:35:39 |
| 885 | .57 | KarolSzmurlo | 1.07304 | 1 | Wed, 15 Jun 2016 12:48:58 |
| 886 | .57 | kmacunl | 1.07584 | 2 | Thu, 05 May 2016 01:50:25 |
| 887 | .57 | soleaf | 1.07768 | 1 | Mon, 02 May 2016 08:29:56 |
| 888 | .52 | noke | 1.08020 | 14 | Sun, 03 Jul 2016 13:28:20 (-46.2h) |
| 889 | .56 | GOGUMA | 1.08461 | 1 | Wed, 22 Jun 2016 13:21:52 |

| | | | | | |
|--|-----|-------------------|---------|----|------------------------------------|
| 875 | .55 | KarolSzmurlo | 1.07304 | 1 | Wed, 15 Jun 2016 12:48:58 |
| 876 | .55 | kmacunl | 1.07584 | 2 | Thu, 05 May 2016 01:50:25 |
| 877 | .55 | soleaf | 1.07768 | 1 | Mon, 02 May 2016 08:29:56 |
| 878 | .50 | noke | 1.08020 | 14 | Sun, 03 Jul 2016 13:28:20 (-46.2h) |
| 879 | .56 | GOGUMA | 1.08461 | 1 | Wed, 22 Jun 2016 13:21:52 |
| 880 | new | LeoNeves | 1.08535 | 3 | Wed, 06 Jul 2016 20:57:54 |
| Your Best Entry ↑ You improved on your best score by 0.76289. You just moved up 115 positions on the leaderboard. Tweet this! | | | | | |
| 881 | .57 | rb1310 | 1.08929 | 23 | Mon, 04 Jul 2016 18:35:51 (-12.8d) |
| 882 | .57 | Ivar | 1.09060 | 14 | Mon, 27 Jun 2016 18:31:23 (-0.1h) |
| 883 | .57 | MatthewChandler | 1.09089 | 1 | Wed, 04 May 2016 10:13:12 |
| 884 | .57 | OlexanderYermakov | 1.09775 | 6 | Mon, 04 Apr 2016 09:15:19 |
| 885 | .56 | Js | 1.09974 | 1 | Wed, 27 Apr 2016 20:12:13 |
| 886 | .56 | weedjy | 1.10076 | 34 | Sat, 14 May 2016 16:34:27 (-0.6h) |
| 887 | .56 | rjwo | 1.10194 | 7 | Fri, 01 Apr 2016 18:15:29 (-3d) |
| 888 | .56 | odo54 | 1.10336 | 4 | Sat, 18 Jun 2016 02:33:36 |
| 889 | .56 | 13oblige | 1.10336 | 2 | Sat, 18 Jun 2016 02:33:36 |
| 890 | .56 | takusamar | 1.10336 | 2 | Sat, 18 Jun 2016 02:33:59 |
| 891 | .56 | alpakazuma | 1.10336 | 2 | Sat, 18 Jun 2016 02:34:01 |
| 892 | .56 | LucyK | 1.10413 | 22 | Tue, 07 Jun 2016 10:03:50 (-10.8h) |
| 893 | .56 | PierreDesbrieres | 1.10708 | 6 | Sun, 19 Jun 2016 12:44:56 |

Diante desta melhora considerável, decidimos treinar um pouco mais e a melhora foi bem pequena.

Apesar de nosso esforço, a estratégia não resultou em uma melhora significativa. Mantivemos o número de épocas, **15**, e aumentamos o número de ciclos para **5**.

Conseguimos a melhora de apenas 1 posição. Passando para a posição **880**.