**RELATÓRIO DE PROJETO**

Douglas Felizardo de L. Lima - 2016003123

Professora Thaís Gaudencio – Introdução à Inteligência Artificial

João Pessoa,14/06/2018

**1. Resumo**

Usar dos algoritmos de machine learning, tanto de aprendizado supervisionado, quanto não supervisionado, para entender a estrutura do nível de dificuldade dos monstros, em uma base de dados coletada manualmente.

**3. Introdução**

Encontrar na versão 3.5 do sistema de RPG Dungeons and Dragons, onde, ainda hoje se mantém como o mais popular e completo, uma estrutura ou padrão que nos permita encontrar um padrão no nível de desafio dos monstros por inteligência artificial, facilitando o trabalho do mestre, diminuindo a quantidade de tabelas que o mesmo precisa olhar.

A maioria dos atributos numéricos presentes no sistema seguem fórmulas estabelecidas nos vários livros do sistema, como o livro do mestre, livro do jogador ou livro dos monstro, como no caso do bônus de vida, que é determinado pela fórmula:

Entretanto, em relação ao nível de desafio, a métrica, algoritmo ou fórmula para sua determinação é apenas a checagem de uma aproximação em várias tabelas espalhadas pelos livros do sistema, sendo assim, ele é o único, que caso o mestre crie um monstro personalizado, demandaria muito tempo.

**4.1 Ferramentas**

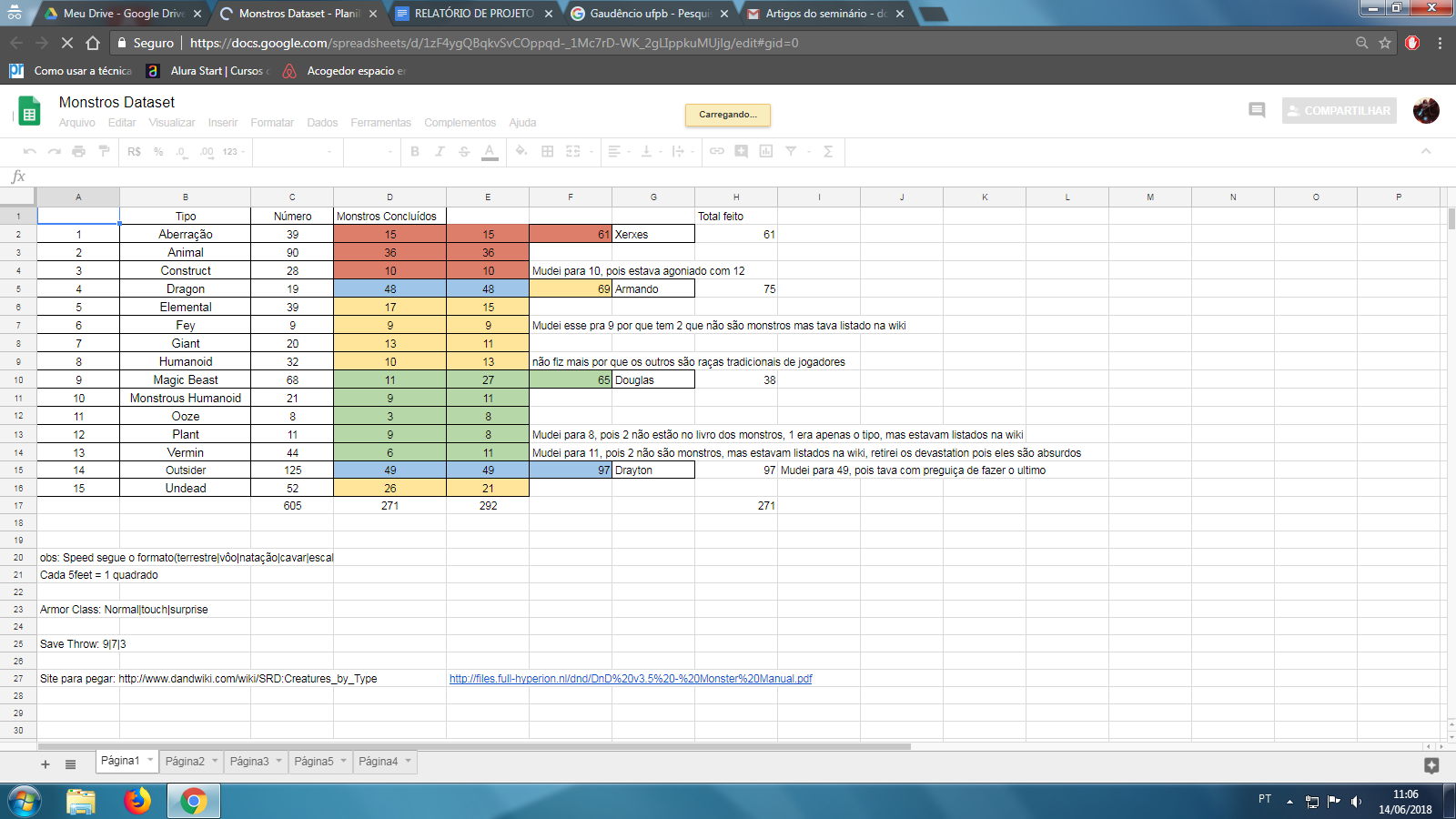
* Usada a linguagem python devido a sua facilidade de escrita e extensa biblioteca para uso científico;
* Algoritmos de Machine Learning foram executados com auxílio da biblioteca Scikit Learning, escolhida por ser eficiente, fácil, muito difundida pela comunidade e boa documentação.
* Para carregar as bases de dados no formato csv, se utilizou do Pandas, por ser necessário em alguns métodos do Scikit, abstração ao se tratar de dicionários e facilidade na manipulação de arquivos
* Para os métodos não supervisionados, se usou o numpy para criar objetos array estruturados na mesma ordem que o ancidataframe carregado pelo Pandas, de forma que retirassem os rótulos de nosso dataset.
* De modo que fosse possível ver a distribuição das instâncias nos grupos formados, se usou o matplotlib para melhor visualização.

**4.2 Metodologia**

Como uma base de dados do sistema não se encontrava disponível, se retirou os dados manualmente de livros do sistema até um total de 270 instâncias.

Durante a coleta de dados, se optou por uma porcentagem de monstros de um estrutura para cada tipo, de forma que a base tivesse uma proporção de sua presença nos livros representados na base. Monstros cujo tipo eram muito raros na base, se optou por colocar todos devido a escassez de amostras.

Do total de 1322 monstros, foram considerados apenas os 605 dos livros bases, no manual dos monstros 1, 2, 3, 4 e 5. Desse total, 270 monstros foram coletados seguindo a proporção estabelecida acima:



Durante a extração dos monstros, se preferiu pegar todos os detalhes referentes a eles para que a base possa ser usada em trabalhos futuros, por outras pessoas e com outros objetivos, de maneira que a mesma possa ser divulgada em sites como Kraggle, crowdAI ou DrivenData. Os valores coletados manualmente de cada monstro foram passados para a base em inglês, visando seu uso global. O atributo que vamos procurar determinar está denotado pela cor vermelha:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Type | Sub-Type | Syze | Hit dice | Dice Type | Life Bonus | Initiative |
| Speed | Armor Class | Base Attack | Grapple | Attack | Full Attack | Space|Reach | Special Attacks | Special Qualities |
| Fortitude | Reflexes | Will | Str | Dex | Con | Int | Wis | Cha |
| Skills | Feats | Environments | Challenge Rating | Treasure | Alignment | LvL Adjustment | Organization | Advancement |

Para diminuir a quantidade de dimensões em nosso DataSet, alguns atributos foram cortados por serem muito raros em criaturas, tal qual o caso da resistência ou imunidade mágica que pouquíssimos seres possuíam essa característica.

Com isso chegamos a esse total de 36 atributos que poderiam ser usados em diversas aplicações computacionais, entretanto, no ponto de vista dos algoritmos apresentados em inteligência artificial, esse número elevado de atributos, simboliza uma quantidade muito grande de dimensões que deveriam ser consideradas em nossos problemas, dificultando as generalizações de nossa IA, além do que, alguns desses atributos são frutos de proporções de outros atributos, como no caso do Life Bonus.

Alguns desses atributos também não oferecem muita capacidade de aprendizado para a nossa IA, como é o caso do Treasure, que a grande maioria das criaturas possuem o valor standard para ele, gerando nenhuma capacidade de aprendizado através dele.

Fazendo um estudo a respeito de cada uma dessas características dos monstros, consultando mestres e jogadores para auxiliar na tomada de decisão e juntando com o devido conhecimento dos algoritmos de inteligência artificial, diminuímos de 36 atributos, para um total de 16:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Syze | Hit dice | Dice Type | Armor Class | Base Attack |
| Fortitude | Reflexes | Will | Str | Con | Int |
| Wis | Cha | Environments | LvL Adjustment | Alignment | Challenge Rating |

Como os métodos do sklearn usam apenas números para seus métodos e devido a falta de uma melhor ferramenta nativa em sua classe de pré-processamento, foi confeccionado uma função para transformar os atributos textuais qualitativos em números discretos que fossem referentes a cada uma dessas qualidades, chamada de “discretiza”. Essa função criava uma estrutura de dados de dicionário, que para cada novo conjunto textual encontrado, era passado um valor numérico começando do 0 e subindo progressivamente à medida que novos conjuntos eram encontrados, depois disso todas as instâncias tinham seus valores traduzidos de acordo com esse dicionário.

Outra função criada para se fazer o pré-processamento veio a ser necessário graças ao valor neutro, comum em alguns atributos. Ele simboliza a inexistência daquele atributo para a instância. Por exemplo o atributo de constituição, que representa a saúde, o vigor e condição física de um personagem comum que caso tenha o seu valor levado a 0, morre. Entretanto. Seres construtos ou mortos-vivos, que são apenas uma massa de matéria animada graças ao sistema mágico do jogo, não possuem uma energia vital em seus corpos, por isso seu valor de constituição é inexistente e consequentemente, inalterável. Outro exemplo são os monstros do tipo planta, que mesmo sendo um ser vivo, não possui nenhuma inteligência, logo a ela é atribuída o valor neutro, quanto que para um ser vivo comum, caso sua inteligência seja levada a 0, ele é considerado em estado de demência ou coma.

Como se foi configurado que qualquer conjunto textual recebesse um valor numérico começando do 0 e aumentando para cada novo conjunto, nos casos desses atributos inexistentes apenas se precisou alterar a marcação ‘-‘, que foi reconfigurada para o valor inteiro -1, pois assim não se atrapalha a predição em métodos de regressão. A procura desses valores inexistentes foi feito em todos os elementos do DataSet.

Agora que os atributos estavam todos com dados numéricos e discretos, ao se utilizar alguns algoritmos se viu que a Inteligência artificial apenas predizia valores baixos de Challenge Rate e quando o monstro deveria ter níveis de Challenge Rate alto ou médio, a distância entre o valor predito e o real era grande, denotando sinais de super-ajustamento da base, entretanto, a acurácia da mesma ficava alta(86%~88%)

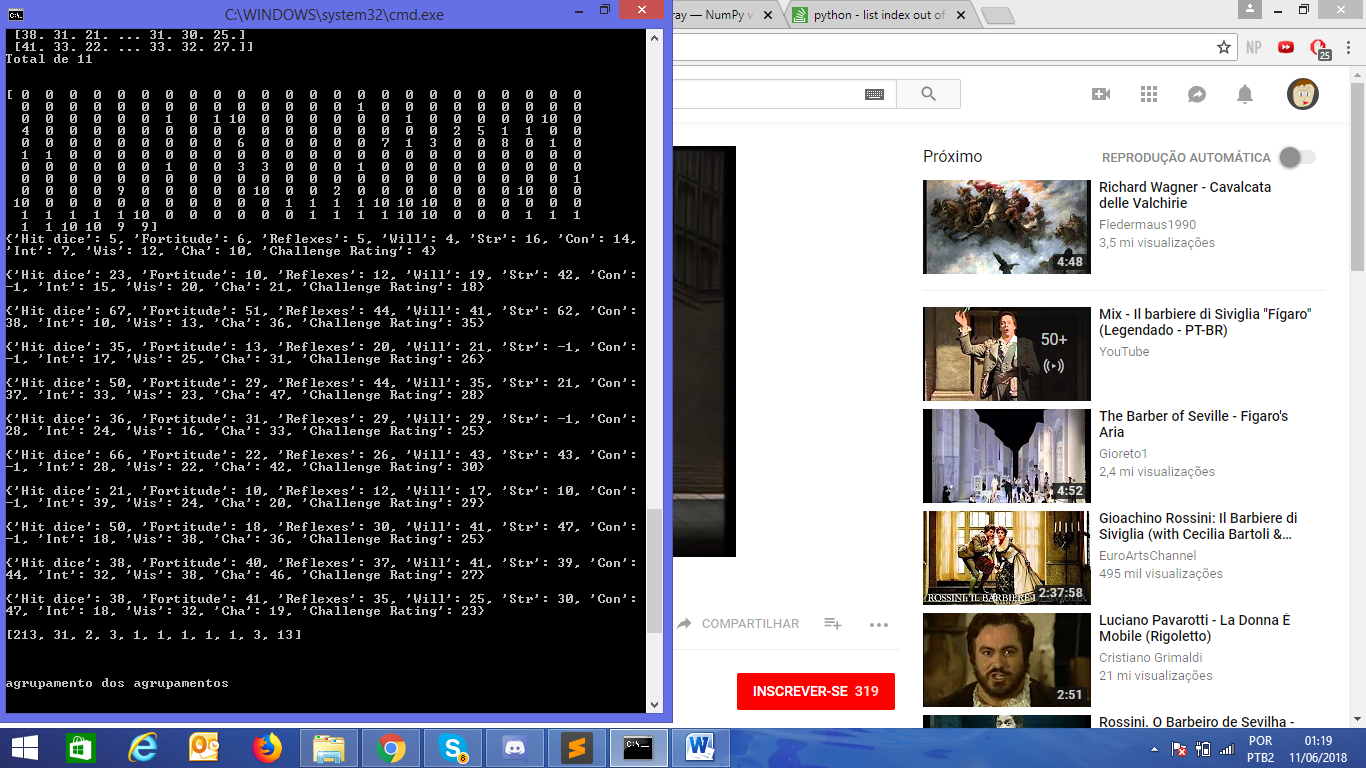
Como estávamos trabalhando com um problema de 16 dimensões, era impossível representar graficamente como estava a distribuição de nossos dados para que pudessemos o entender e ajustar nosso pré-processamento na base, entretanto, havia um método de aprendizado que poderia nos dar estruturas para podermos ir entendendo como estava a distribuição de nossos dados nesse espaço, os métodos de aprendizado não supervisionado.

Na aprendizagem não supervisionada nós temos duas alternativas, métodos em que você diz quantos clusters ele deve formar, como o K-means e métodos que vão determinar grupos baseados nos dados que passarmos a ele, e o kernel especificado, como o Mean-Shift.

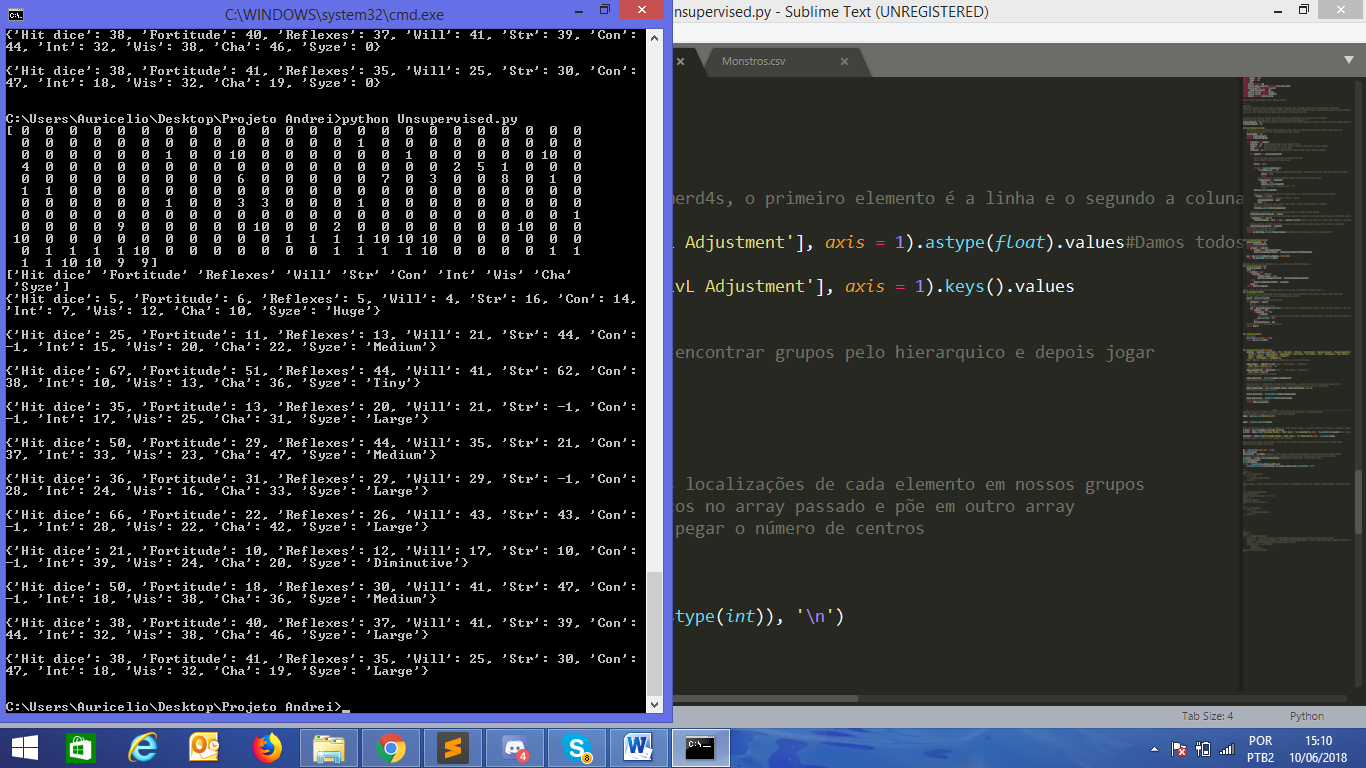
Já que estamos procurando alguma estrutura desconhecida em nossa base de dados e não conseguimos os visualizar os dados por estarem dispersos em muitas dimensões, dentre as duas técnicas apresentadas, foi escolhido o Mean-Shift por ela se encaixar justamente no que foi demandado.

Usou-se então esse método várias vezes, colocando e retirando atributos, até que se chegou em algumas relações.

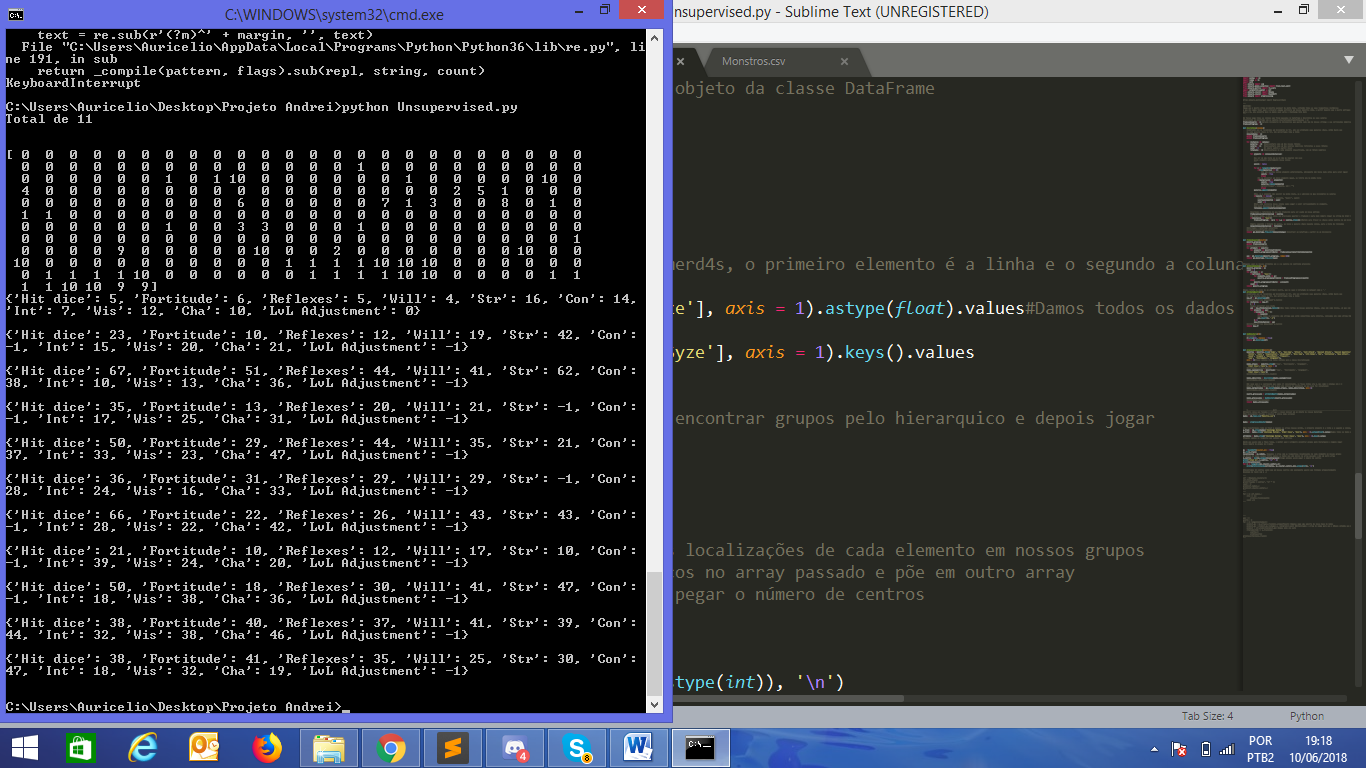
Conseguimos alguns atributos que seguiam um padrão de agrupamento com centróides que faziam sentido e possuíam correlação com o challenge rate:



Entretanto, em relação a outros atributos, situações extremamente adversas aconteceram nos valores dos desses centróide, como no caso do tamanho, quando relacionado com o respectivo grupo, situações plenamente anormais ocorriam:



Outro exemplo de problema encontrado está no LvL adjustment que quando se foi coletado para a base de dados, se percebeu que muitos monstros tinham essa característica com o valor neutro, consequentemente com pouquíssimos exemplos numéricos, além do -1 que é configurado para esses valores no pré-processamento, sendo então plenamente inutilizado pela IA:



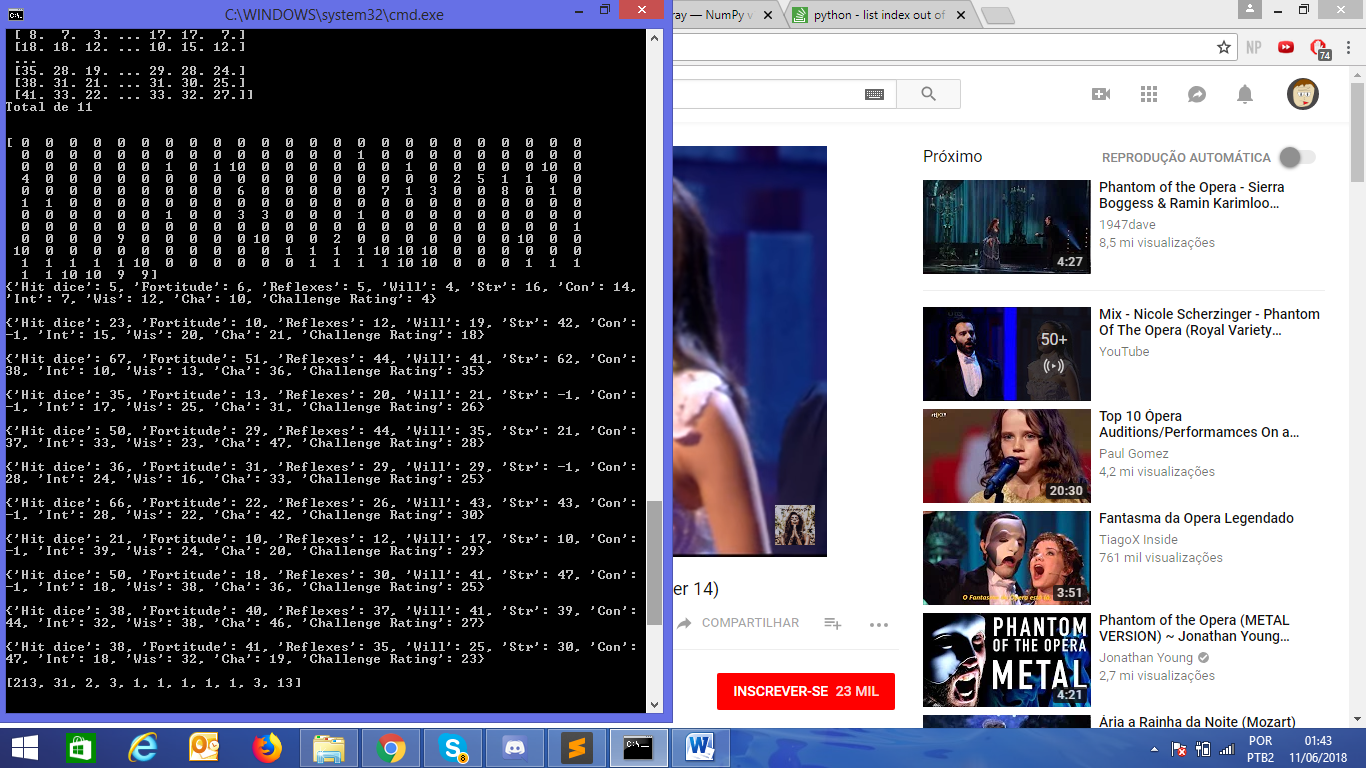
E assim se foi verificando para cada um dos atributos e vendo se os mesmos são correlacionados, até chegar a esses atributos finais:

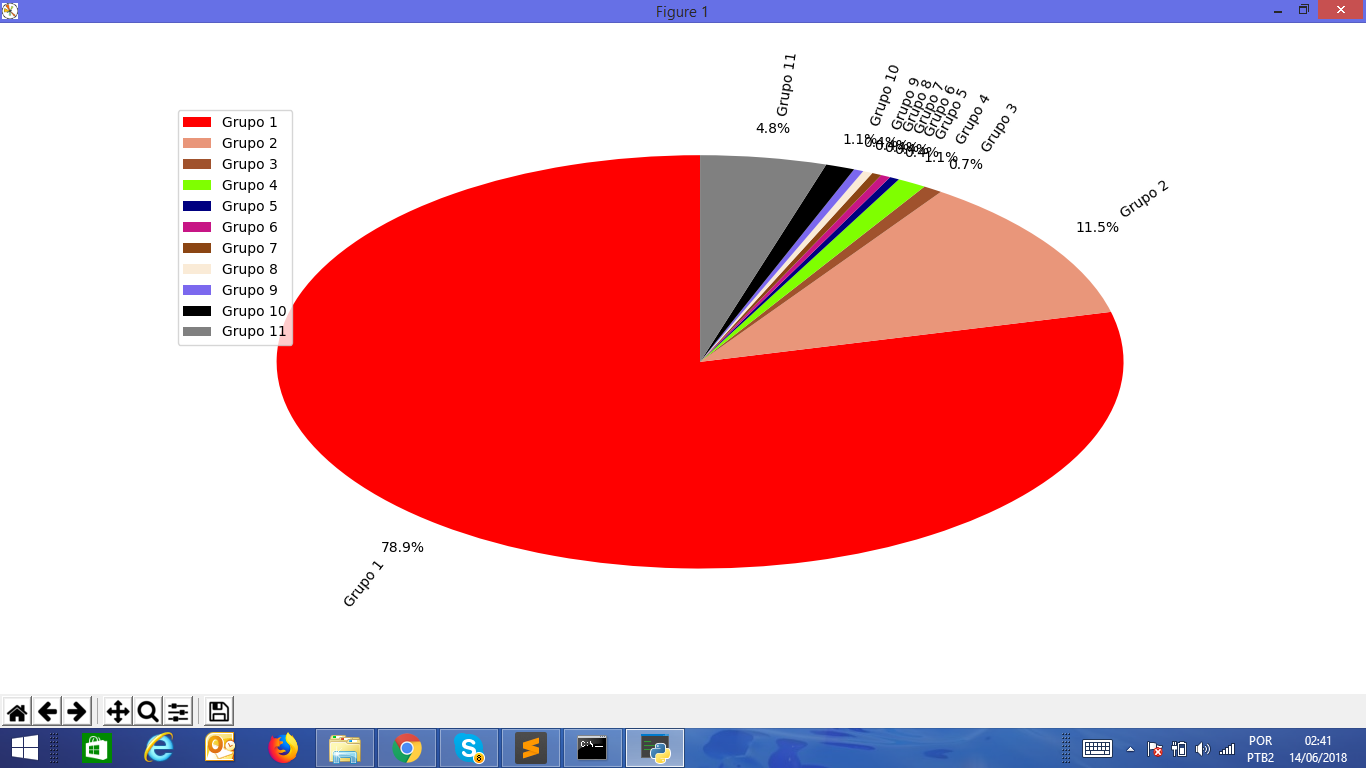
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fortitude | Reflexes | Will | Str | Con |
| Int | Wis | Cha | Challenge Rating |  |

O próximo passo foi começar a olhar as instâncias, já que tínhamos reduzido ao máximo as dimensões de nosso problema.

Vamos agora fazer um estudo mais aprofundado de nossas instâncias, utilizando-se da concepção que no Mean Shift todos as instâncias são inicialmente consideradas como possíveis conjuntos, então se é feita uma verificação das distâncias, que no nosso caso estamos usando a euclidiana, com todos as instâncias em volta até se chegar a um grupo que todos convertam.

Essa noção é muito útil para encontrar outliers, pois a mesma estará muito longe de qualquer outro elemento para formar um grupo, listamos então quais instâncias pertenciam a cada agrupamento e assim descobrimos algumas instâncias que ficaram isoladas na base de dados no gráfico abaixo, eles seguem as mesmas posições respectivas dos grupos encontrados anteriomente:



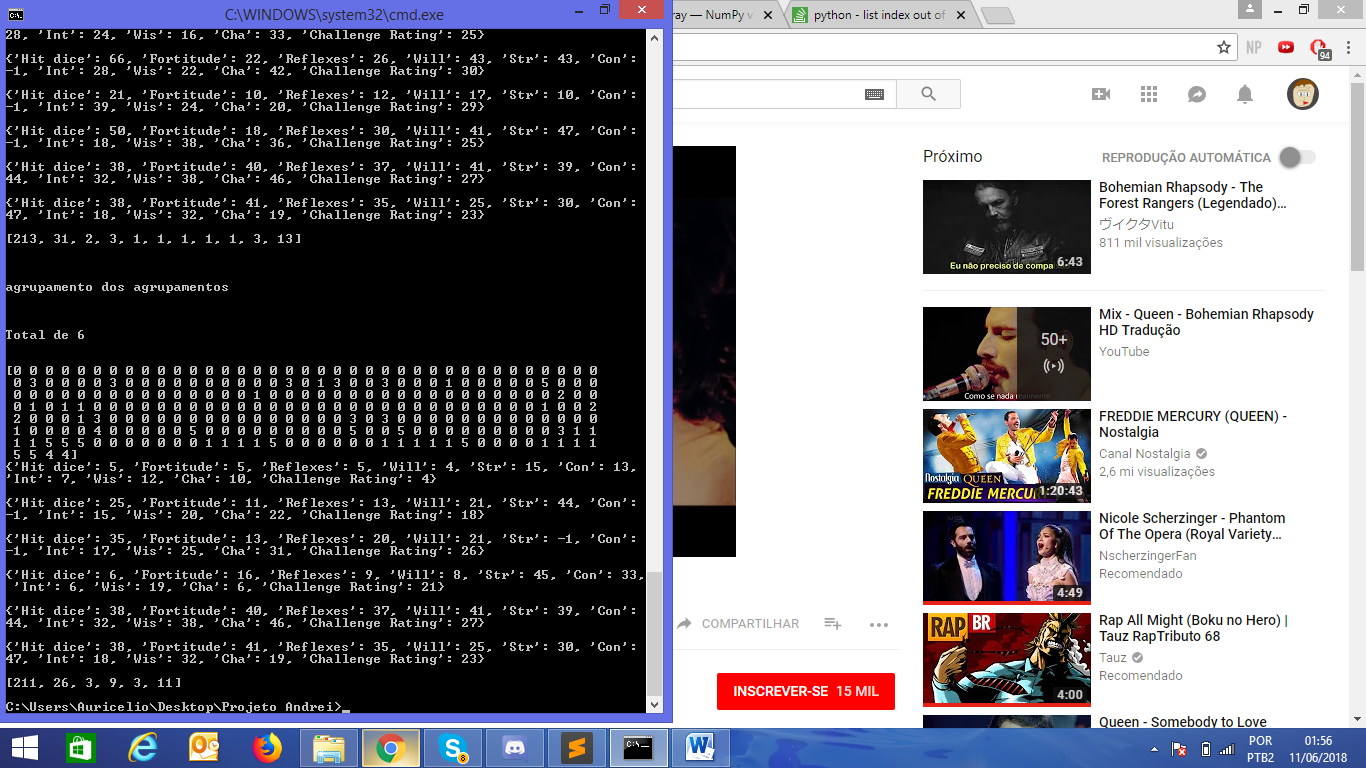


Apenas analisando esses grupos já entendemos o porquê inicialmente tínhamos acurácias tão boas, basicamente a nossa inteligência artificial estava considerando apenas o grupo 1 e 2 em suas predições, tendo em base apenas as porcentagens que o gráfico nos dá.

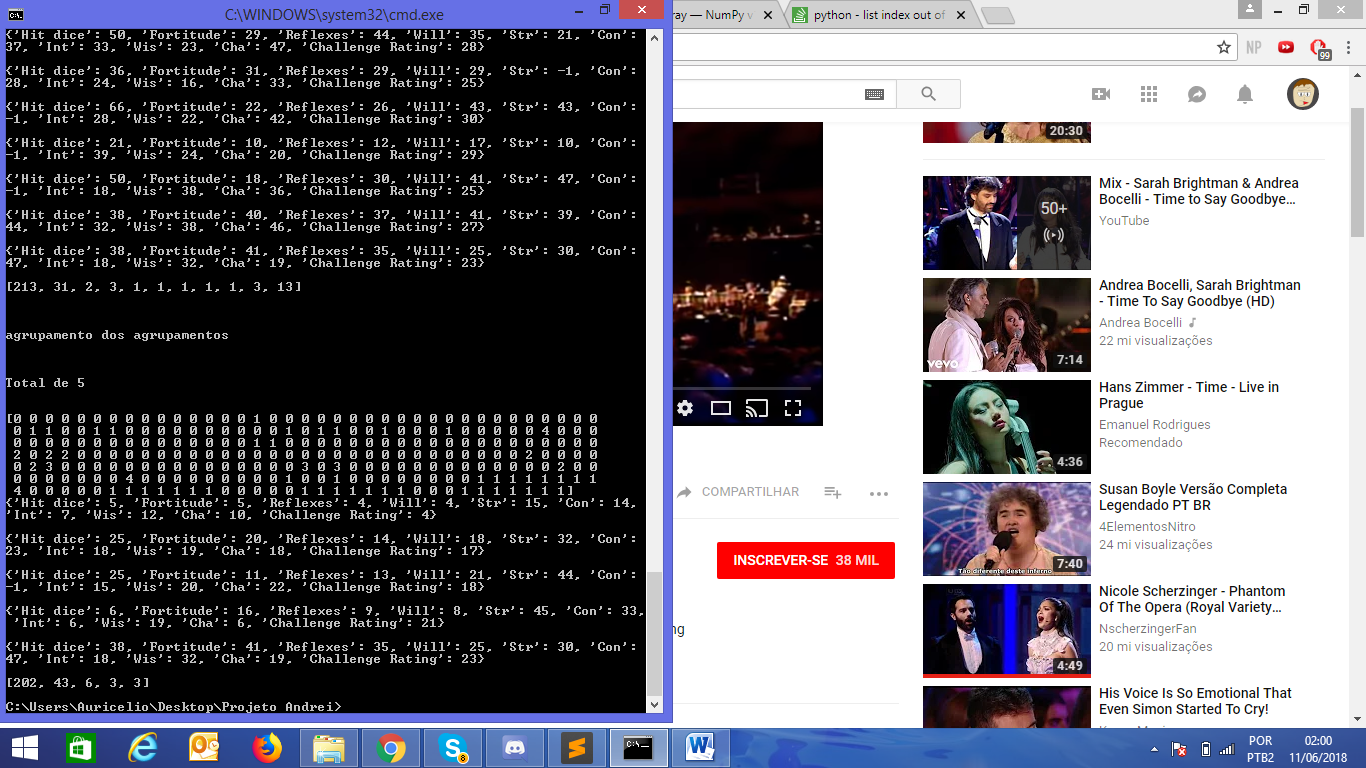
Vemos aqui também que 5 grupos são formados basicamente para abrigar outliers, contendo apenas 1 elemento. Já os outros 3 grupos que comportam poucos elementos foi necessária uma análise para saber se era melhor retirar ou não eles de nossa base, visto que temos poucos elementos.

Para isso vamos usar o Mean Shift mais duas vezes, criando dois conjuntos de agrupamentos, um com os grupos contendo três ou mais elementos e outro apenas com os resultados maiores que três e ver em qual das duas temos os grupos melhor estruturado melhor.

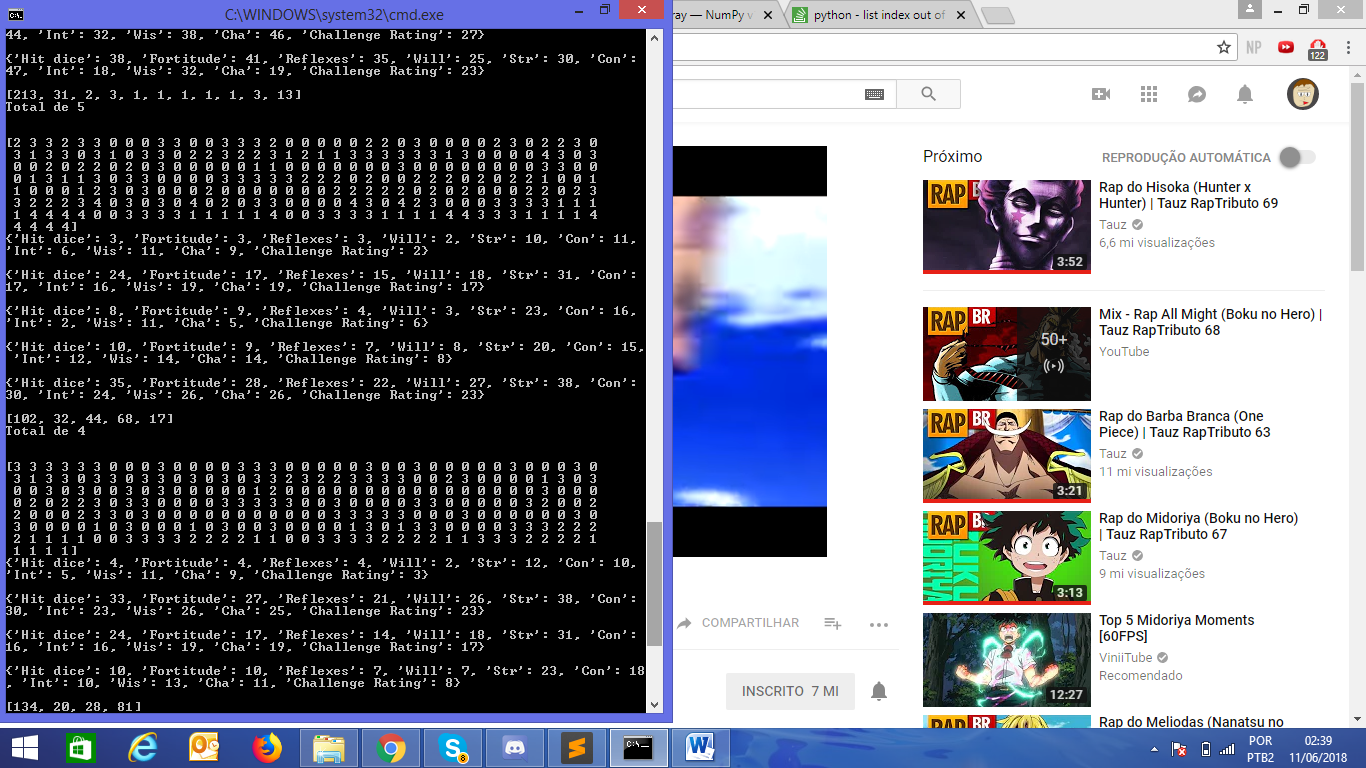
1º conjunto de agrupamentos(>=3):



2º conjunto de agrupamentos(>3):



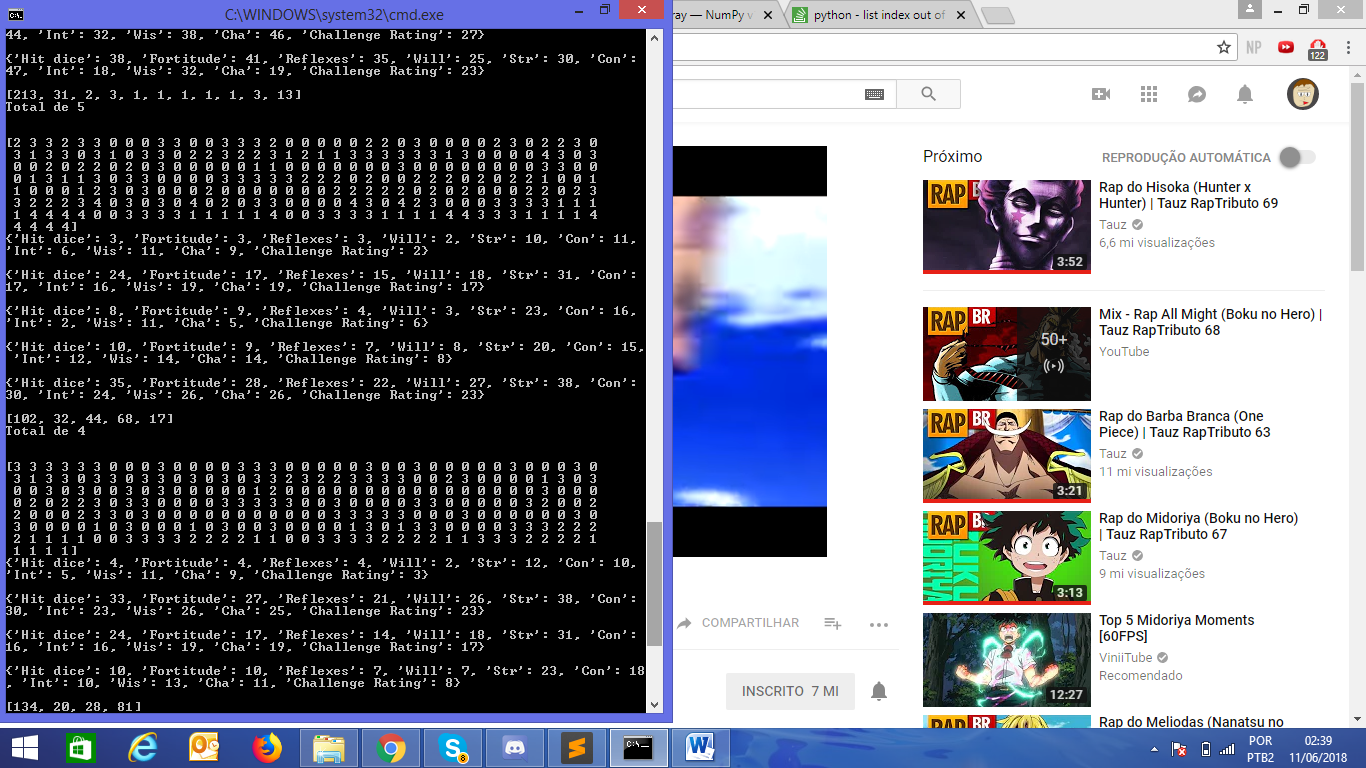
Vemos que o primeiro conjunto acaba tendo uma melhor distribuição de dados por ter mais amostras, mas no segundo a distribuição dos níveis de desafio fica bem suave e bem definida com os 5 grupos definidos. Com isso em mente vamos mandar a base de dados do conjunto um para treinar 5 grupos por meio do algoritmo K-Means:



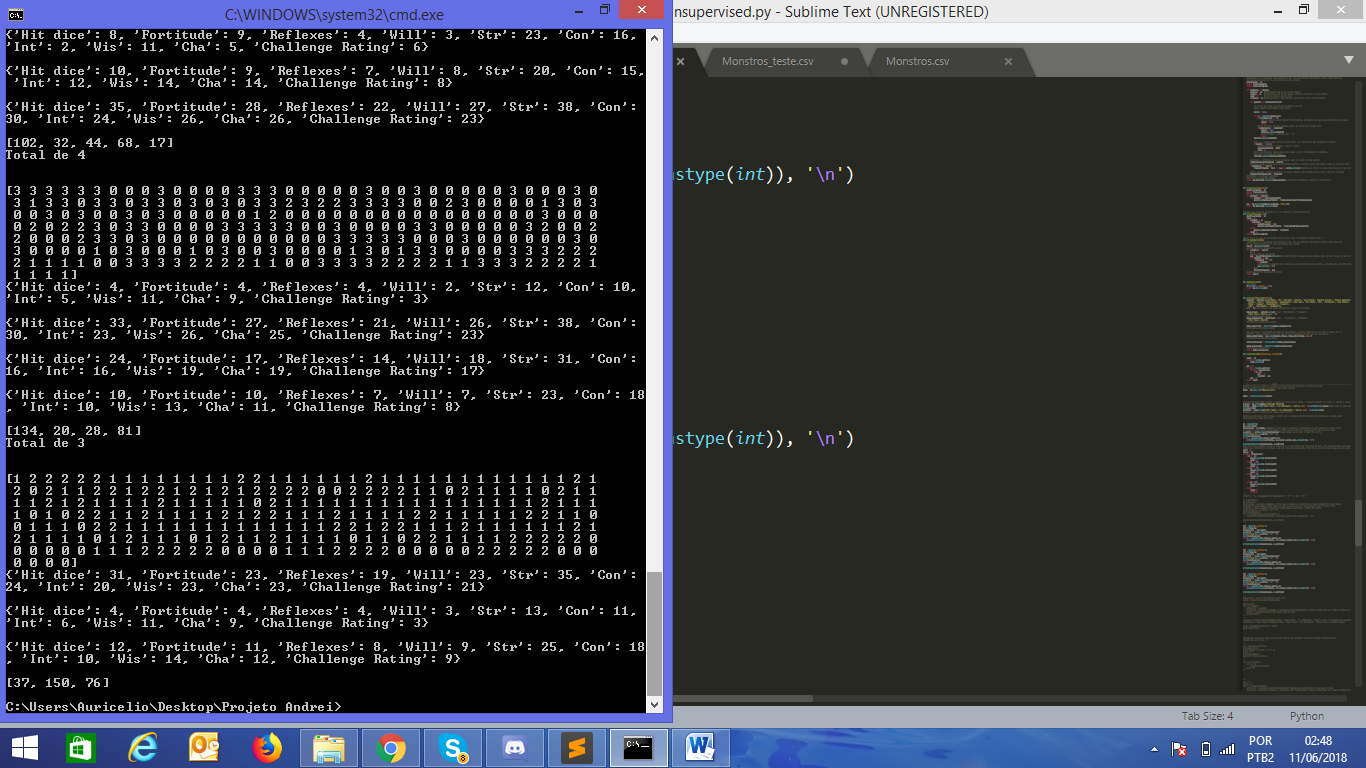
Veja que aqui os grupos e as instâncias ficam melhores distribuídos que no Mean Shift, os atributos estão com valores que fazem sentido de maneira geral e com um Challenge Rating que gradualmente aumenta em nossa base de dados, já que há esse aumento gradual, talvez se diminuíssemos os grupos, poderíamos continuar tendo essa evolução gradual, entretanto, em grupos melhor definido.

Como o Challenge Rating aumenta inicialmente de maneira bem gradual, vamos fazer outros dois conjuntos, um com 4 e outro com 3 grupos, respectivamente:

1º Conjunto:



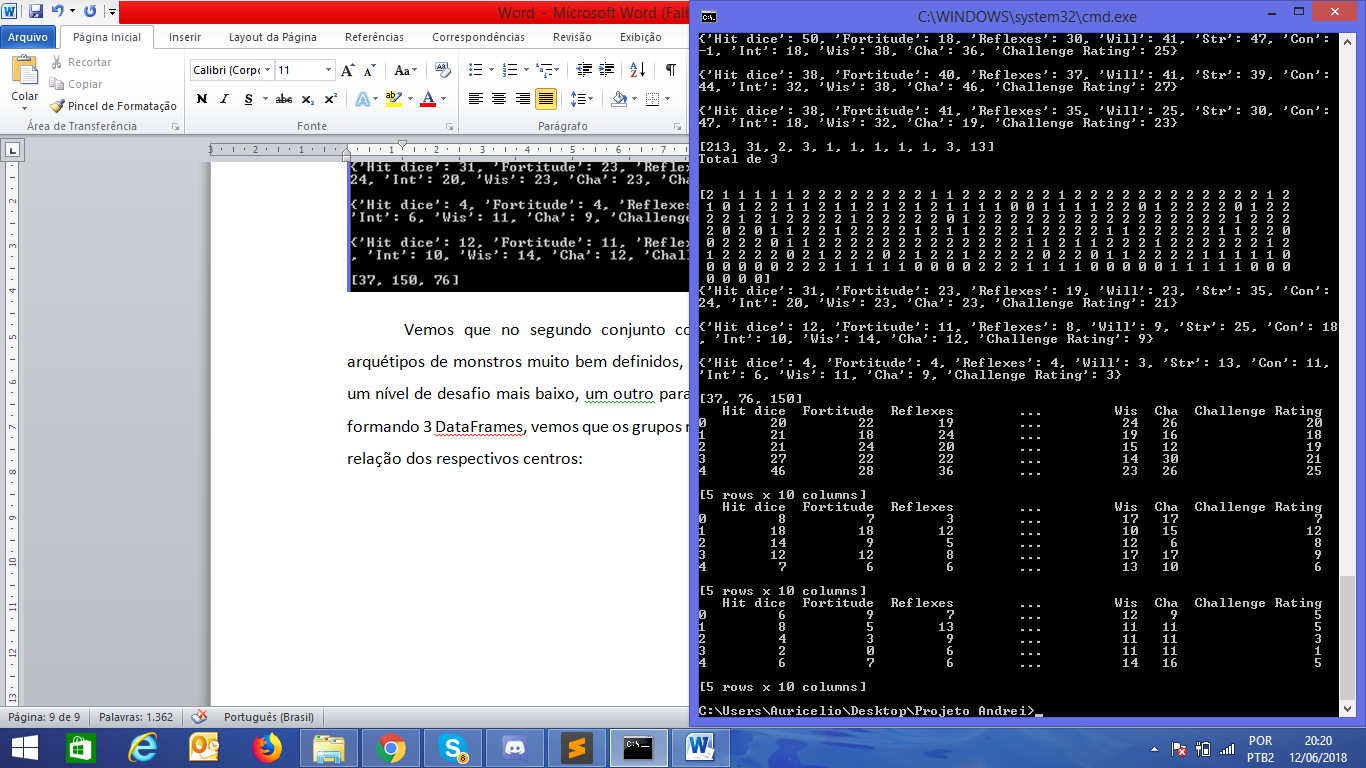
2º Conjunto:



Vemos que no segundo conjunto conseguimos três grupos com centróides em arquétipos de monstros muito bem definidos, um primeiro grupo para monstros que tenham um challange rate mais baixo, um para médios e o último para níveis mais elevados, formando esses 3 DataFrames.

**5. Resultados**

Como vemos os grupos realmente estão nas proporções corretas com a relação aos seus respectivos centros:



Utilizando então dos métodos supervisionados, visando procurar treinar uma IA que fique especialista em cada um dos grupos, podendo então aproveitar características de vários métodos, temos como os melhores resultados:

* 10 Cross-Validation
* Grupo mais numeroso com menor Challenge Rate, melhor método Linear Regression - 74%
* Grupo com elementos com challange rate médio, melhor método Knn 64%
* Grupo com menos elementos e challange rate alto, melhor método Random Forest - 72%

**6. Conclusão**

Através dessa abordagem, consegui ver as vantagens da capacidade de generalização e visão computacional garantida pelos métodos não supervisionados que com sua ajuda puderam proporcionar resultados mais precisos nos respectivos métodos supervisionados.

Em relação a base de dados se viu que há uma grande falta de monstros fortes para a compor, não estando bem balanceada em relação ao seu número de elementos.

Na parte supervisionada, o único método que retornou resultados positivos em relação aos monstros com alto challenge rating foi o Random Forest, já os outros tiveram resultados bem medíocres, denotando quase nenhuma aprendizagem dos dados. Suponho que isso se deve que quanto mais alto o nível do monstro, menos seu poder tem relação com esses atributos base e mais com suas habilidades, magias ou talentos. Sendo assim é preciso levar eles em conta para melhorar a acurácia para esses monstros.