**qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm**

|  |
| --- |
| Relatório Machine learnig  Trabalho da disciplina de IA  21/09/2018  Thiago Outeiro e Joyce Brum |

Analisando o dataset: Iris Plants Database

# Objetivo:

Tendo como entrada um conjunto de dados sobre flores e sua classificação entre os tipos “Iris-setosa”, “Iris-versicolor” e “Iris-virginica”, o programa tem como objetivo receber dados de outro conjunto de rosas e tentar classificar este novo conjunto como um dos três tipos acima listados.

# Metodologia:

A metodologia utilizada foi baseada no PLA (perceptron learning algorithm), que visa traçar retas que separem da melhor maneira possível os três tipos de flor. A cada iteração, o algoritmo percorre os vetores de entrada procurando pelo primeiro que gera uma classificação errada. Então, o algoritmo reajusta a “reta” que havia sido traçada, repetindo o loop até o critério de parada ser atingido (que no nosso caso foi um número máximo de passos ou nenhuma classificação errada ser encontrada).

Para separar os vetores com as informações de cada flor entre dados e caso de teste, utilizamos um algoritmo que lia o arquivo com todas as flores e randomicamente selecionava 20% dos dados e escrevia em um arquivo. Os demais eram escritos em outro arquivo.

Foi necessário gerar duas funções de separação, a primeira separava a flor “Iris-setosa” das demais e a segunda função separava a flor “Iris-versicolor” das demais. Dessa forma, as duas retas tentavam linearmente separar os 3 grupos de rosas dentro das características apresentadas.

A função de teste lia o arquivo com os dados de teste e verifica se o ponto estava acima ou abaixo da primeira função. Se estivesse acima (o sinal da multiplicação escalar entre a matriz de transformação da função e a matriz com o vetor de teste desse positivo), então a flor era classificada como Iris-setosa, caso contrário era feito o mesmo teste com a segunda reta. Se o teste desse sinal positivo, ela seria classificada como Iris-versicolor, e se desse negativo, lhe sobraria Iris-virginica como classificação.

# Resultados:

Do teste (com valores separados randomicamente) rodado no algoritmo, houve um acerto de 100% dos valores dados, superando, e muito, as nossas expectativas.

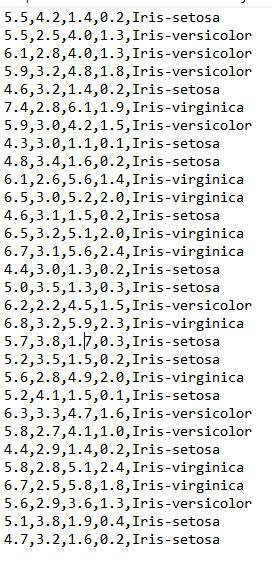


# Conclusão:

Apesar de ser um algoritmo simples, percebemos que o mesmo se mostrou muito mais poderoso do que esperávamos.

Além disso, a justificativa geométrica do funcionamento do algorítimo ficou ainda mais clara quando tivemos que implementá-lo, ampliando ainda mais o nosso conhecimento sobre o mesmo.

Estes foram os resultados obtidos pelo nosso algoritmo de PLA, usando estes valores para teste:

Os dados utilizados como banco de dados do algoritmo se encontram na lista a seguir:

5.1,3.5,1.4,0.2,Iris-setosa

4.9,3.0,1.4,0.2,Iris-setosa

4.7,3.2,1.3,0.2,Iris-setosa

5.0,3.6,1.4,0.2,Iris-setosa

5.4,3.9,1.7,0.4,Iris-setosa

4.6,3.4,1.4,0.3,Iris-setosa

5.0,3.4,1.5,0.2,Iris-setosa

4.9,3.1,1.5,0.1,Iris-setosa

5.4,3.7,1.5,0.2,Iris-setosa

4.8,3.0,1.4,0.1,Iris-setosa

5.8,4.0,1.2,0.2,Iris-setosa

5.7,4.4,1.5,0.4,Iris-setosa

5.4,3.9,1.3,0.4,Iris-setosa

5.1,3.5,1.4,0.3,Iris-setosa

5.1,3.8,1.5,0.3,Iris-setosa

5.4,3.4,1.7,0.2,Iris-setosa

5.1,3.7,1.5,0.4,Iris-setosa

4.6,3.6,1.0,0.2,Iris-setosa

5.1,3.3,1.7,0.5,Iris-setosa

4.8,3.4,1.9,0.2,Iris-setosa

5.0,3.0,1.6,0.2,Iris-setosa

5.0,3.4,1.6,0.4,Iris-setosa

5.2,3.4,1.4,0.2,Iris-setosa

4.8,3.1,1.6,0.2,Iris-setosa

5.4,3.4,1.5,0.4,Iris-setosa

4.9,3.1,1.5,0.2,Iris-setosa

5.0,3.2,1.2,0.2,Iris-setosa

5.5,3.5,1.3,0.2,Iris-setosa

4.9,3.6,1.4,0.1,Iris-setosa

5.1,3.4,1.5,0.2,Iris-setosa

4.5,2.3,1.3,0.3,Iris-setosa

4.4,3.2,1.3,0.2,Iris-setosa

5.0,3.5,1.6,0.6,Iris-setosa

4.8,3.0,1.4,0.3,Iris-setosa

5.1,3.8,1.6,0.2,Iris-setosa

5.3,3.7,1.5,0.2,Iris-setosa

5.0,3.3,1.4,0.2,Iris-setosa

7.0,3.2,4.7,1.4,Iris-versicolor

6.4,3.2,4.5,1.5,Iris-versicolor

6.9,3.1,4.9,1.5,Iris-versicolor

5.5,2.3,4.0,1.3,Iris-versicolor

6.5,2.8,4.6,1.5,Iris-versicolor

5.7,2.8,4.5,1.3,Iris-versicolor

4.9,2.4,3.3,1.0,Iris-versicolor

6.6,2.9,4.6,1.3,Iris-versicolor

5.2,2.7,3.9,1.4,Iris-versicolor

5.0,2.0,3.5,1.0,Iris-versicolor

6.0,2.2,4.0,1.0,Iris-versicolor

6.1,2.9,4.7,1.4,Iris-versicolor

6.7,3.1,4.4,1.4,Iris-versicolor

5.6,3.0,4.5,1.5,Iris-versicolor

5.6,2.5,3.9,1.1,Iris-versicolor

6.3,2.5,4.9,1.5,Iris-versicolor

6.1,2.8,4.7,1.2,Iris-versicolor

6.4,2.9,4.3,1.3,Iris-versicolor

6.6,3.0,4.4,1.4,Iris-versicolor

6.8,2.8,4.8,1.4,Iris-versicolor

6.7,3.0,5.0,1.7,Iris-versicolor

6.0,2.9,4.5,1.5,Iris-versicolor

5.7,2.6,3.5,1.0,Iris-versicolor

5.5,2.4,3.8,1.1,Iris-versicolor

5.5,2.4,3.7,1.0,Iris-versicolor

5.8,2.7,3.9,1.2,Iris-versicolor

6.0,2.7,5.1,1.6,Iris-versicolor

5.4,3.0,4.5,1.5,Iris-versicolor

6.0,3.4,4.5,1.6,Iris-versicolor

6.7,3.1,4.7,1.5,Iris-versicolor

6.3,2.3,4.4,1.3,Iris-versicolor

5.6,3.0,4.1,1.3,Iris-versicolor

5.5,2.6,4.4,1.2,Iris-versicolor

6.1,3.0,4.6,1.4,Iris-versicolor

5.8,2.6,4.0,1.2,Iris-versicolor

5.0,2.3,3.3,1.0,Iris-versicolor

5.6,2.7,4.2,1.3,Iris-versicolor

5.7,3.0,4.2,1.2,Iris-versicolor

5.7,2.9,4.2,1.3,Iris-versicolor

6.2,2.9,4.3,1.3,Iris-versicolor

5.1,2.5,3.0,1.1,Iris-versicolor

5.7,2.8,4.1,1.3,Iris-versicolor

6.3,3.3,6.0,2.5,Iris-virginica

5.8,2.7,5.1,1.9,Iris-virginica

7.1,3.0,5.9,2.1,Iris-virginica

6.3,2.9,5.6,1.8,Iris-virginica

6.5,3.0,5.8,2.2,Iris-virginica

7.6,3.0,6.6,2.1,Iris-virginica

4.9,2.5,4.5,1.7,Iris-virginica

7.3,2.9,6.3,1.8,Iris-virginica

7.2,3.6,6.1,2.5,Iris-virginica

6.4,2.7,5.3,1.9,Iris-virginica

6.8,3.0,5.5,2.1,Iris-virginica

5.7,2.5,5.0,2.0,Iris-virginica

6.4,3.2,5.3,2.3,Iris-virginica

6.5,3.0,5.5,1.8,Iris-virginica

7.7,3.8,6.7,2.2,Iris-virginica

7.7,2.6,6.9,2.3,Iris-virginica

6.0,2.2,5.0,1.5,Iris-virginica

6.9,3.2,5.7,2.3,Iris-virginica

7.7,2.8,6.7,2.0,Iris-virginica

6.3,2.7,4.9,1.8,Iris-virginica

6.7,3.3,5.7,2.1,Iris-virginica

7.2,3.2,6.0,1.8,Iris-virginica

6.2,2.8,4.8,1.8,Iris-virginica

6.1,3.0,4.9,1.8,Iris-virginica

6.4,2.8,5.6,2.1,Iris-virginica

7.2,3.0,5.8,1.6,Iris-virginica

7.9,3.8,6.4,2.0,Iris-virginica

6.4,2.8,5.6,2.2,Iris-virginica

6.3,2.8,5.1,1.5,Iris-virginica

7.7,3.0,6.1,2.3,Iris-virginica

6.3,3.4,5.6,2.4,Iris-virginica

6.4,3.1,5.5,1.8,Iris-virginica

6.0,3.0,4.8,1.8,Iris-virginica

6.9,3.1,5.4,2.1,Iris-virginica

6.9,3.1,5.1,2.3,Iris-virginica

5.8,2.7,5.1,1.9,Iris-virginica

6.7,3.3,5.7,2.5,Iris-virginica

6.7,3.0,5.2,2.3,Iris-virginica

6.3,2.5,5.0,1.9,Iris-virginica

6.2,3.4,5.4,2.3,Iris-virginica

5.9,3.0,5.1,1.8,Iris-virginica

# Conclusão:

O resultado obtido foi muito além das nossas expectativas, nos mostrando que, pelo menos para o caso que testamos, o algoritmo era extremamente poderoso. Além disso, seu funcionamento e explicação geométrica ficaram extremamente mais claro após a implementação do mesmo.

Analisando o dataset: Discrimination in Salaries

# Objetivo:

Tendo como entrada um conjunto de dados sobre a distribuição de salários dos funcionários de uma empresa, o programa tem como objetivo receber dados de outro funcionário e tentar prever corretamente seu possível salário.

Diferente do caso anterior, fica mais complicado calcular o erro deste algoritmo, pois não existem mais apenas dois resultados possíveis: acertar e errar, agora é importante avaliar o distanciamento entre o valor encontrado pelo algoritmo ao valor real que corresponde ao salário do funcionário.

# Metodologia:

A metodologia utilizada foi baseada no método de Regressão Linear, que tenta calcular o valor da função de Rn → R que mapeia cada dado de funcionário em seu respectivo salário, minimizando o erro do somatório de (h(xi) – yi)2/n onde 0 <= i <= n.

Dessa forma, através de uma manipulação algébrica foi possível achar que w = pseudoInversa(X) x Y, onde X é a matriz onde cada linha é um vetor contendo os dados de um funcionário e Y é um vetor contendo os salários de casa funcionário.

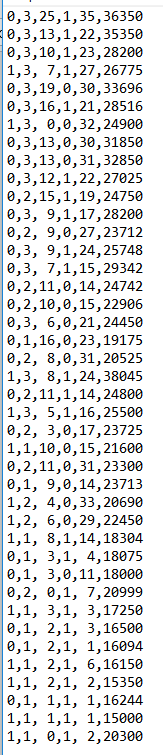
Como o numpy possui um método para o cálculo da pseudo-inversa, o algoritmo se tornou muito mais fácil de implementar.

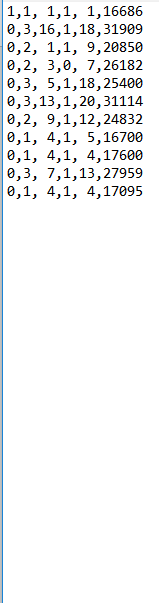
A função testeRegressao() roda automaticamente o algoritmo de Regressão e já a testa, imprimindo os valores encontrados e esperados e o erro calculado. Implementamos também um método calculaErro() que calcula o erro usando os valores iniciais em vez dos valores de teste.

Os dados foram divididos da seguinte forma: foram escolhidos 20% aleatórios dentre os existentes e divididos em dois arquivos.

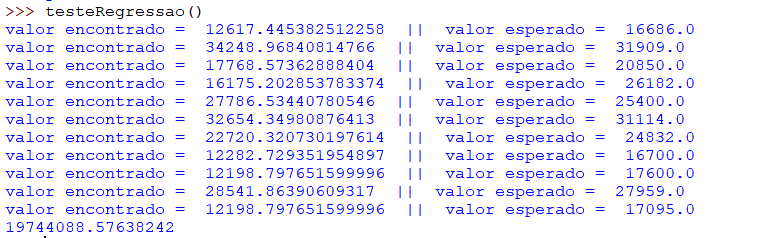
# Resultados:

Segue respectivamente os dados de utilizados como dados iniciais e os dados de teste:

  
Figura 1: dados iniciais

  
Figura 2: dados de teste

Os resultados obtidos para esses testes seguem abaixo:

O valor especificado na última linha representa o erro calculado utilizando os valores de entrada dos casos de teste. O valor retornado pelo método calculaErro() utilizando os valores de dados iniciais para o cálculo do teste foi:



# Conclusão:

O erro foi muito maior do que estávamos esperando para este algorítmo, entretanto, o resultado da previsão de salário foi interessante, pois o valor retornado sempre se mantinha de certa forma próximo ao valor desejado.