# Parte 1 – Métodos de Classificação

## Escolha das *features*

Com o objetivo de se obter um erro dentro dos limites de tolerância (≤ 30%), tendo em conta os exemplos de classificação (*True* ou *False*) de palavras dados no enunciado do projeto e as palavras para avaliação incluídas no ficheiro de texto, optou por se escolher as seguintes *features*:

🡪 Tamanho da palavra;

🡪 Paridade do número de vogais;

🡪 Primeira letra da palavra ser uma vogal;

🡪 Palavra tem acentuação;

🡪 Palavra tem letras repetidas;

## 1.2 Análise dos métodos de aprendizagem

Com estas *features* foram testados diferentes métodos de aprendizagem através dos seguintes algoritmos, de onde se obtiveram as seguintes percentagens de sucesso de classificação:

|  |  |  |
| --- | --- | --- |
| Algoritmo | Teste 1 | Teste 2 |
| *Neighbors* | 74% | 72% |
| *Decision Tree* | 84% | 84% |
| *Linear* | 73% | 79% |

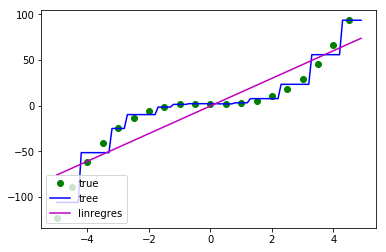


Gráfico 2: Teste de Regressão 2 - FAILED

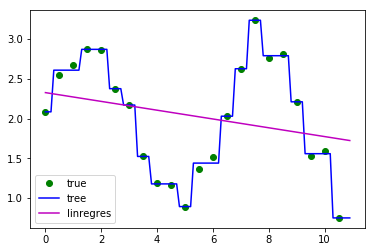


Gráfico 1: Teste de Regressão 1 - FAILED

Antes de uma decisão final, foi repetida a análise dos métodos de aprendizagem com as seguintes *features*:

🡪 Número ímpar de letras;

🡪 Começar e terminal em vogal;

🡪 O sufixo da palavra ser uma palavra de 3 letras já classificada;

# 1.3 Conclusão

Após a revisão dos resultados obtidos, verificou-se que nenhuma destas últimas *features* alcançava uma percentagem maior do que as que já tinham sido obtidas anteriormente, continuando assim o ***Decision Tree*** a ter a maior percentagem de sucesso.

Apesar da utilização dos parâmetros ***max\_depth*** (profundidade máxima da árvore de decisão) e ***min\_samples\_split***, as percentagens de sucesso nunca foram superiores relativamente às anteriores.



Figura

Em suma, a partir da função acima apresentada e substituindo f(x) pelo vetor de features citado anteriormente e tendo em conta y (variável de saída) e as percentagens obtidas e utilizando utilizando as features definidas para classificar as palavras, conclui-se que o melhor método de aprendizagem é o ***Decision Tree***.

# Parte 2 – Métodos de Regressão

## Análise dos métodos de aprendizagem

Figura

Para o estudo do método de regressão utilizando a

função da figura 2, foram implementados vários tipos de métodos de regressão (Linear, Tree e Kernel Ridge), sendo que a avaliação comparativa foi realizada entre os métodos Tree e Kernel Ridge , pois os métodos de regressão Linear e Tree têm ambos o mesmo resultado de output (y da figura 2), comparação esta que foi maioritariamente realizada observando os gráficos obtidos através da implementação dos algoritmos:

### Tree Regression:

O método de regressão Tree Regression tem a tendência a ficar overfitted (criação de árvores de decisão demasiado complexas não generalizando assim a informação dada como input) em relação aos dados existentes.

Na tentativa de que isso não acontecesse, foi necessário ajustar o parâmetro “Max\_depth” consoante o número exemplos a serem testados (parâmetro “min\_sample\_split”), sem sucesso, neste caso podemos observar nos gráficos 1 e 2 que o teste tem como output “FAILED” não confirmando assim a validação cruzada.

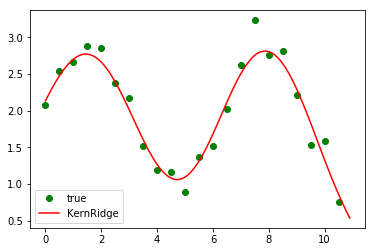
Gráfico 1: Teste de Regressão 1 - FAILED

### Kernel Ridge:

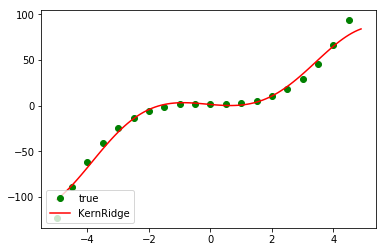
Tipicamente o método de regressão Kernel Ridge é utilizado para conjuntos de informação de tamanho médio, utilizando “squared error loss” (função de perda), tendo assim uma execução consideravelmente rápida.

Neste caso ajustando os parâmetros de validação cruzada (parâmetro “cv”) e da própria função de Kernel Ridge (parâmetro “kernel”, “alfa” e “gama”).

Para o caso em que kernel=’rbf’ (Radial Basis Function) a variância de valores é testada no parâmetro “gama” da função Kernel Ridge, sendo que apenas o primeiro teste de regressão confirmou a validação cruzada.

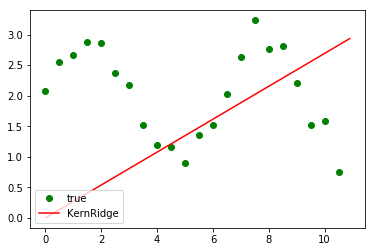


Teste de Regressão 1 (kernel = rbf) - OK



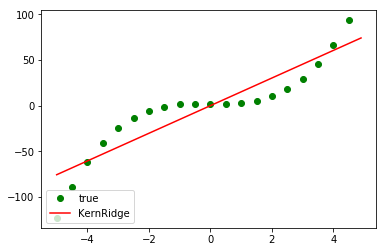
Teste de Regressão 2 (kernel = rbf) - FAILED

Para o caso em que kernel=’linear’ a variância de valores é testada no parâmetro “alpha” da função Kernel Ridge, sendo que apenas o segundo teste de regressão confirmou a validação cruzada.



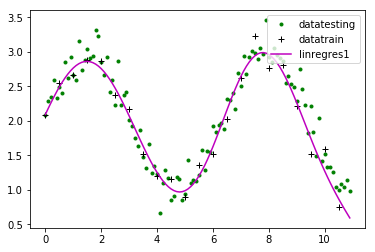
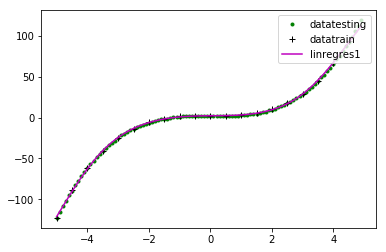
Teste de Regressão 1 (kernel = linear) - FAILED

Teste de Regressão 2 (kernel = linear) - OK



### GridSearchCV with Kernel Ridge:

Com o intuito de obter validação em ambos os casos de teste, através de uma exploração complementar, foi executado o método de aprendizagem *GridSearchCV* baseado no método *Kernel Ridge*, método este que executa uma procura exaustiva:



Teste de Regressão 1 (kernel = rbf) - OK

Teste de Regressão 2 (kernel = rbf) - OK

# Parte 3 – Aprendizagem por Esforço

Qual é o valor de cada ação em cada estado do mundo?

Qual é a politica a seguir em cada estado?

A politica a seguir em cada estado é dada pela função Q2pol(Q,eta=5) em que Q é um valor da composição estado-ação neste mundo e eta é a variável relativa à exploração

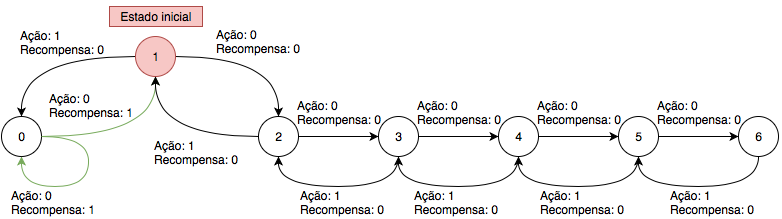
(quanto menor for o valor de eta, maior será a capacidade de exploração).

Sendo assim a politica é dada pela expressão:

**np.exp(eta\*Q)/np.dot(np.exp(eta\*Q),np.array([[1,1],[1,1]])).**

Seguindo a politica apreendida para onde é que o agente vai se começar no estado 3?

O agente começando do estado 3 desloca-se para o estado 4 com a ação 0.

Representação gráfica do ambiente no qual o ambiente se move

../Desktop/Captura%20de%20ecrã%202017-12-09,%20às%2016.51.17.pngQual é a função de recompensa?

Como é que o agente se move?

O agente passa para um estado seguinte se a ação a executar for a ação 0 e retorna para um estado anterior se a ação a executar for a ação 1.

No caso de o estado atual ser o estado 0, a ação 1 faz com que o agente permaneça no mesmo estado com uma recompensa de 1.

Também consegue obter uma recompensa de 1 se passar do estado 0 para o estado 1.