**Relatório Inteligência Artificial**

Tiago Almeida¹, João Figueira²

**P1 - Métodos de Classificação**

**Análise e escolha de parâmetros**

Para abordarmos este problema, onde tínhamos como objectivo predizer as palavras que seriam aceites ou rejeitadas de acordo com um certo conjunto de dados.

Para tal utilizámos um DecisionTreeClassifier num espaço com as seguintes features:

1. Tamanho da palavra
2. Número de vogais na palavra
3. Número de consoantes na palavra
4. Número de ocorrências da letra ‘o’
5. Número de ocorrências da letra ‘a’

Estas duas últimas features foram escolhidas com base em informação estatística recolhida acerca dos conjuntos de dados fornecidos. Desta análise, descobrimos que a letra ‘a’ e a letra ‘o’ eram as que mais apresentavam divergências de frequência entre palavras aceites e rejeitadas.

Recorremos a um DecisionTreeClassifier com os parâmetros standard.

Os parâmetros do classificador foram os standard uma vez que a sua alteração não revelou quaisquer melhorias nos resultados obtidos.

Por outro lado, a adição das features foi feita progressivamente (uma a uma), pela ordem apresentada em cima, tendo-se obtido os resultados apresentados no Gráfico1.

**P2 - Métodos de Regressão**

**Análise e escolha de parâmetros**

Depois de analisar e testar uma parte do leque de regressores do sklearn, concluímos que os regressores kernel\_ridge.KernelRidge e tree.DecisionTreeRegressor eram os que mais se aproximavam dos resultados desejados e optámos por otimizar estes dois métodos de Regressão.

Da natureza dos dados concluímos que o KernelRidge teria que ter os parâmetros “kernel” como “polynomial” e “degree” com um número igual ou maior que quatro (uma vez que no primeiro exemplo, uma boa regressão poderia ser subida ou descida de forma a ter quatro zeros).

Após testados alguns valores para os outros parâmetros dos regressores escolhidos e de não ter sido obtida nenhuma instância que passasse o primeiro teste, optamos por testar múltiplos valores para esses mesmos parâmetros automaticamente: No KernelRidge foram corridos ciclos a testar valores para os parâmetros “alpha”, “degree” e “gama”, pelo que foram encontradas várias combinações dos mesmos que passavam ambos os testes; Para o DecisionTreeRegressor foram testados valores para “min\_samples\_split”, “minsamples\_leaf”, “max\_feature” e “min\_impurity\_decrease” mas não foi encontrada qualquer combinação que passasse qualquer um dos testes.

Optámos por isso por comparar os dois Regressores com os seguintes parâmetros:

tree.DecisionTreeRegressor()

e

KernelRidge(alpha=0.023,kernel="polynomial", degree=7, gamma=0.085)

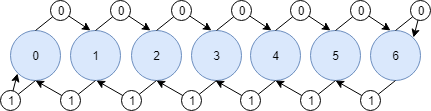
Note-se que como os resultados da experimentação com parâmetros para o DecisionTreeRegressor foram insatisfatórios, concluímos que os parâmetros por omissão eram tão apropriados como os melhores conseguidos.

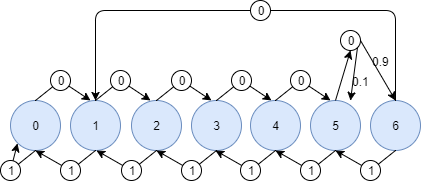
Os resultados obtidos para o DecisionTreeRegressor apresentam-se nos gráficos 2 e 3. E para o KernelRidge nos gráficos 4 e 5

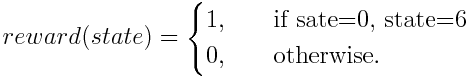
**P3 - Aprendizagem por Reforço**

**Interpretação dos resultados**

Nos testes 1 e 2 a representação gráfica do ambiente no qual o agente se move pode ser dada, respectivamente, pelos MDPs (Markov Decision Process) seguintes.





O agente é premiado quando está no estado 0 ou no estado 6, pelo que a reward function em ambos os ambientes pode ser definida por: 

Com estes reforços, o agente constrói a matriz Q. Esta matriz define, para cada estado, qual a ação mais vantajosa a tomar.

Para cada estado, o agente vai então, de acordo com a matriz Q aprendida, escolher a ação a tomar.

No primeiro caso, ele começa no estado 5, sobe para o estado 6 e mantém-se neste estado uma vez que tem uma reward (reforço) positiva.

Por sua vez, no segundo caso, o agente começa no estado 5, sobe para o estado 6, em seguida vai para o estado 1 e por fim desce para o estado 0.

**Registo e Tratamento de Resultados:**

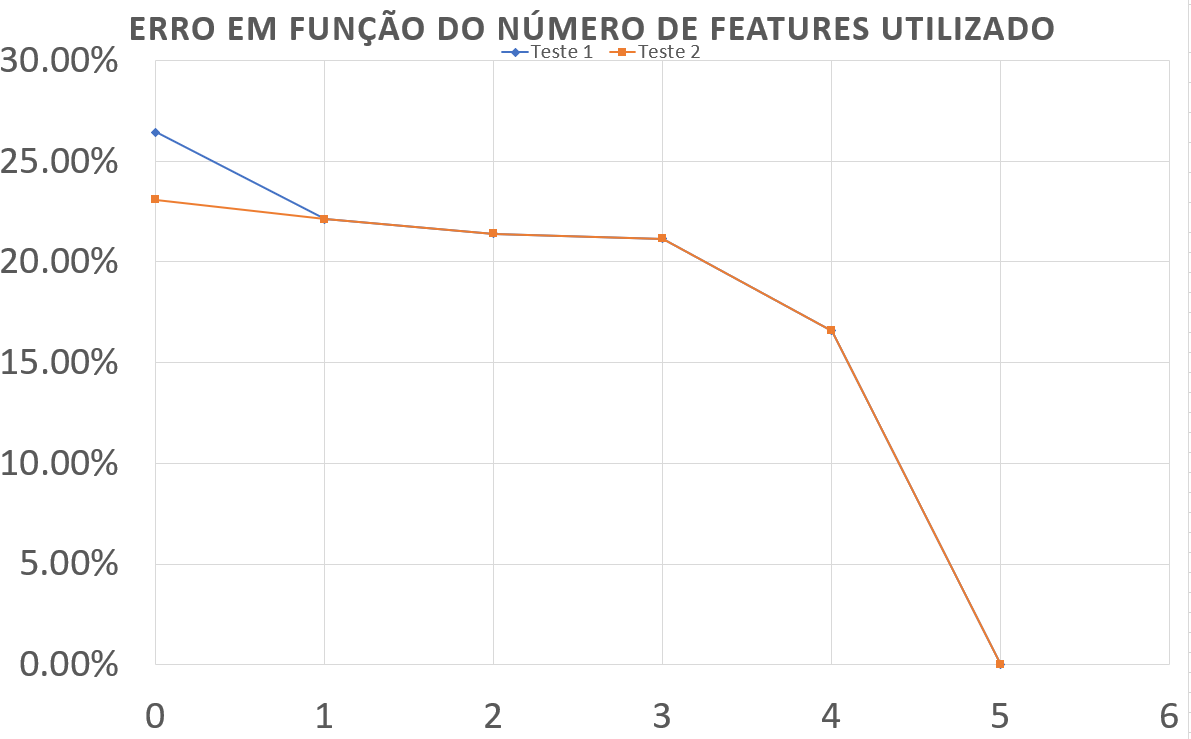


Gráfico 1

Obteve-se para o primeiro teste um erro ‒ percentagem de decisões erradas ‒ de 0.0% e para o segundo 0.24% (ambos abaixo do objectivo de 5%).

Teste wordsclass.npy: Teste wordsclass2.npy:

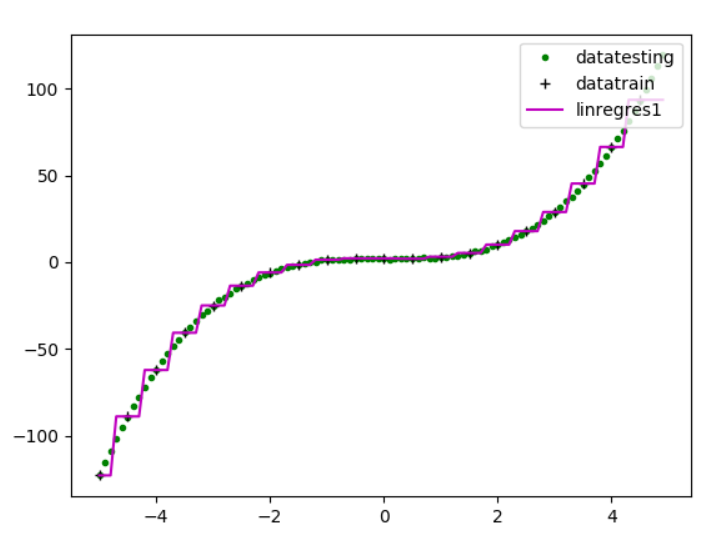
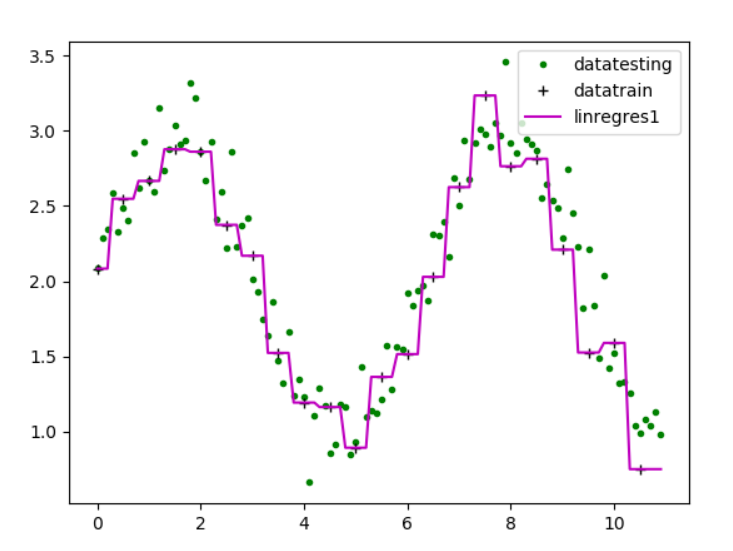


Gráfico 2 Gráfico 3

O erro quadrático médio no primeiro teste foi de 0.731477405523, acima do máximo de 0.3.

O erro médio quadrático no segundo teste foi de 1290.38888543, acima do máximo de 800.

Teste wordsclass.npy: Teste wordsclass2.npy:

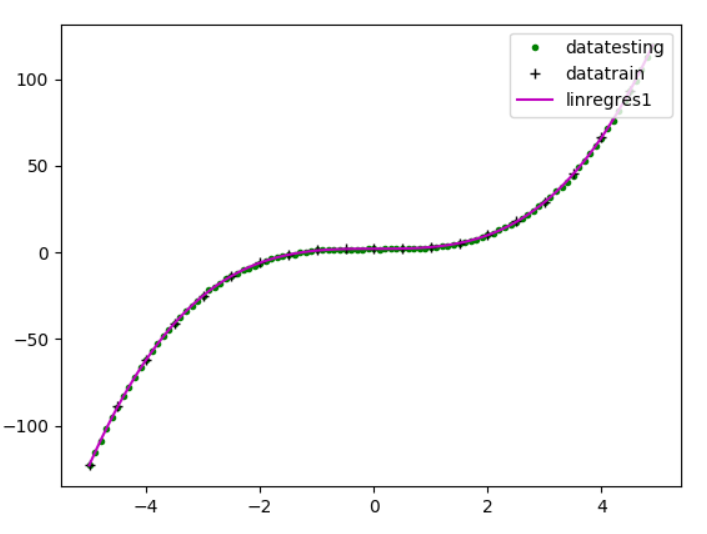
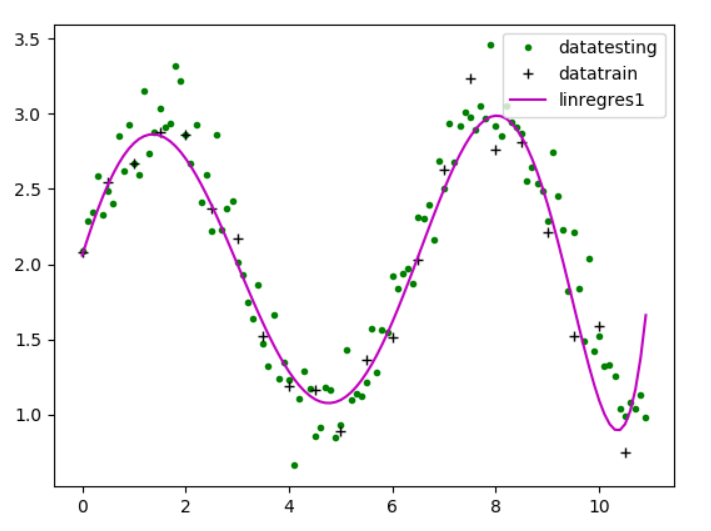


Gráfico 4 Gráfico 5

O erro quadrático médio no primeiro teste foi de 0.110436719258, abaixo do máximo de 0.3.

O erro médio quadrático no segundo teste foi de 16.3777010006, abaixo do máximo de 800.