Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática

SI - Computação Natural

TP4
Support Vector Machines

Grupo 2:
André Gonçalves, A75625
Miguel Miranda, A74726
Rogério Moreira, A74634
Tiago Sá, A71835

Braga, 29 de Abril de 2018

Conteúdo

1. Introdução

Maquinas de vetores de suporte (SVMs) apresentam-se como um algoritmo de *Machine Learning*, enquadrado no paradigma de aprendizagem supervisionada.

Estas técnicas oferecem uma elevada capacidade de generalização, conseguindo obter resultados robustos sem necessitar de um elevado volume de dados para o processo de treino. Pelas suas características, SVMs são utilizadas em diversos contextos e áreas de conhecimento, como em reconhecimento de padrões, imagens, problemas de classificação ou regressão numérica.

Devido à sua utilização no presente, torna-se assim relevante a compreensão dos conceitos que dão base a esta técnica de extração de conhecimento.

Como estrutura do presente relatório: o capitulo 2 realiza uma breve introdução teórica dos fundamentos por detrás das SVMS, abordando de que forma esta técnica interpreta e extrai padrões dos dados; o capitulo 3 descreve a interpretação de resultados do uso de SVMs, de forma a demonstrar algumas das características destes algoritmos. Por fim, o capitulo 4 apresenta as conclusões levantadas após o desenvolvimento deste projeto.

2. Máquinas de Vetores Suporte

Máquinas de vetores de suporte ou, *Support Vector Machines* (SVM) considerando a convencional designação em inglês, são algoritmos de otimização matemática baseados no paradigma de aprendizagem supervisionada. Tendo ganho atualmente uma atenção crescente, estas técnicas são normalmente aplicados a contextos de tratamento, classificação de dados e processamento de imagens.

Num algoritmo SVM, cada ponto do conjunto de dados é representado num espaço de ***N*** dimensões, onde ***N*** representa o número de atributos das instâncias em estudo. Um ponto é assim projetado pelo valor numérico dos seus atributos, nos respetivos eixos que definem o espaço de representação.

Cada um destes pontos individuais é visto como um *vetor de suporte* e a fronteira que delimita os dados em classes é designada de *máquina de vetores suporte*. A determinação destas "fronteiras"é assim o ponto chave destes algoritmos, dado que a sua equação determina o limite entre diferentes classes, no processo de classificação.

Com a aplicação do algoritmo SVM a um conjunto de dados, pretende-se criar um conjunto de regras que permitam analisar os atributos de cada registo e classifica-lo como pertencente a uma das classes do contexto do problema.

Num exemplo simples, considerando um conjunto de dados para avaliar o sexo de um indivíduo, usando como parâmetros a sua altura e o comprimento do seu cabelo, existirão duas classes (sexo feminino e sexo masculino) e dois atributos (altura e comprimento do cabelo do indivíduo). Neste cenário, os vetores de suporte do problema são representados num espaço de 2 dimensões (Figura 2.1),

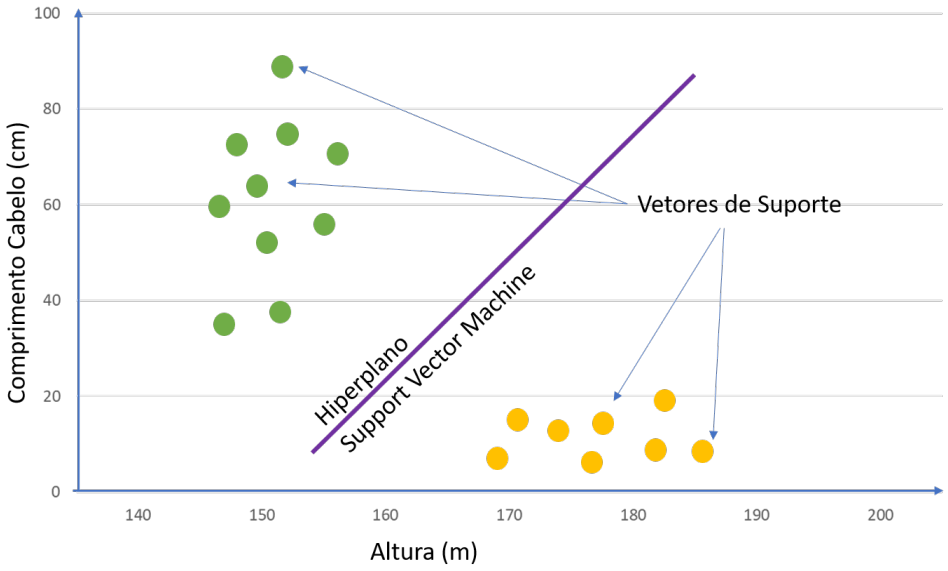


Figura 2.1: Ilustração da projeção de vetores de suporte, num problema de classificação binário com duas dimensões.

Considerando problemas onde é possível realizar uma separação linear dos dados, como o ilustrado no exemplo anterior, a tarefa do algoritmo SVM passa pelo reconhecendo de padrões associados a cada uma das classes, analisando o valor dos atributos dos casos do dataset de treino que recaem nessas classes [3, 1].

Mais especificamente, o SVM irá descrever a equação de um hiperplano capaz de delimitar e categorizar as classes da melhor forma. De referir que em alguns casos podem ser usados mais do que um hiperplano para delimitar as classes.

No caso de existirem diversos hiperplanos aparentemente semelhantes para dividir as classes, o critério de seleção recai sobre o conceito de maximização das margens: Para cada hiperplano, é calculada a distância até ao vetor de suporte mais próximo de cada classe. O hiperplano que maximizar esta distância entre as diferentes classes é o selecionado. [2, 4]

A definição de um hiperplano mais próximo de uma classe do que de outra, pode levar à classificação incorreta de novos casos futuros, assim como levar a tendências de overfitting face a uma determinada classe. Este procedimento de maximização das margens entre as classes e o hiperplano, é assim uma forma de aumentar a robustez da solução ótima calculada.

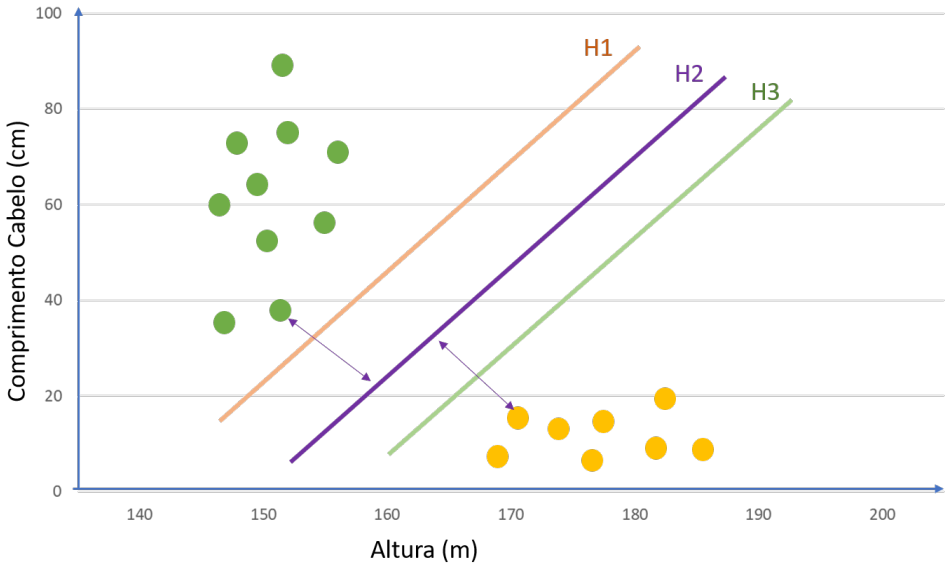


Figura 2.2: Exemplo de escolha de um hiperplano, com base no critério de maximização das margens

Considerando os hiperplanos apresentados na Figura 2.2:

- O plano de corte **H1**, por ter uma margem muito próxima aos casos da classe *Sexo Feminino*, classificaria como pertencentes à classe *Sexo Masculino* casos que podem ser da classe *Sexo Feminino*;
- De forma análoga, o plano de corte **H3**, leva a uma sobrevalorização da classe *Sexo Feminino*, podendo falhar futuramente em casos que pertencem à classe *Sexo Masculino*;
- O hiperplano **H2**, representa assim a solução ótima, por procurar maximizar as margens entre as duas classes.

Desta forma permite que em avaliações com novos dados, exista uma flexibilidade na variação dos valores dos atributos, sem que os mesmos sejam classificados de forma errada.

Acima deste critério de maximização das margens entre classes, o algoritmo SVM procura selecionar o hiperplano que maximiza a precisão da previsão para cada classe. Desta forma, o hiperplano vai tentar dividir explicitamente todos os casos analisados no treino do classificador, mesmo que este processo implique que a solução final se apresente com uma margem inferior perante uma determinada classe.

No caso do conjunto de dados conter instâncias que se identifiquem como *outliers*, o algoritmo tem ainda a capacidade de os detetar e os ignorar, mantendo os critérios acima referidos.

A Figura 2.3 procura esquematizar o cenário em que:

- O critério de precisão na divisão de classes se sobrepõe ao critério de maximização das margens entre classes;
- Na presença de *outliers*, que impedem que um plano linear corte as duas classes, estes são descartados da análise, permitindo assim definir o hiperplano ***H1***.

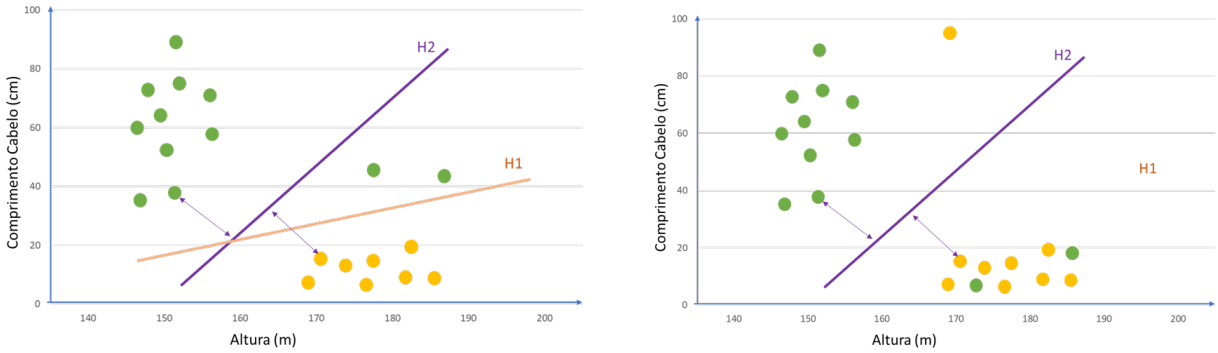


Figura 2.3: Esquema conceptual de cenários excecionais na definição de um hiperplano.

2.1 SVM não linear e *kernel trick*

Em contextos de classificação e análise de relações entre dados mais complexas, nem sempre é possível uma separação linear entre as classes. Nestas situações, o algoritmo SVM tem a capacidade de adaptar o contexto do problema, considerando um domínio de projeção dos dados superior ao inicial. A procura de uma separação entre as classes é assim realizada neste novo espaço de representação dos dados [4, 1].

Esta conversão é realizada internamente pelo algoritmo, através de um processo chamado de *Kernel Trick*. Normalmente, as plataformas que implementem funções baseadas em *SVMs* permitem a escolha da função que se quer utilizar no *Kernel Trick*.

O aumento do espaço de representação é conseguido introduzindo uma nova *feature* (eixo ortogonal) que geralmente relaciona os valores de outros atributos. Apesar do aumento das dimensões do espaço de representação, ser visualmente mais complexo de representar, a formulação matemática que suporta esta técnica permite por vezes encontrar hiperplanos de corte de forma mais simples e precisa. Nestes contextos, os hiperplanos deixam geralmente de ser representados por equações lineares.

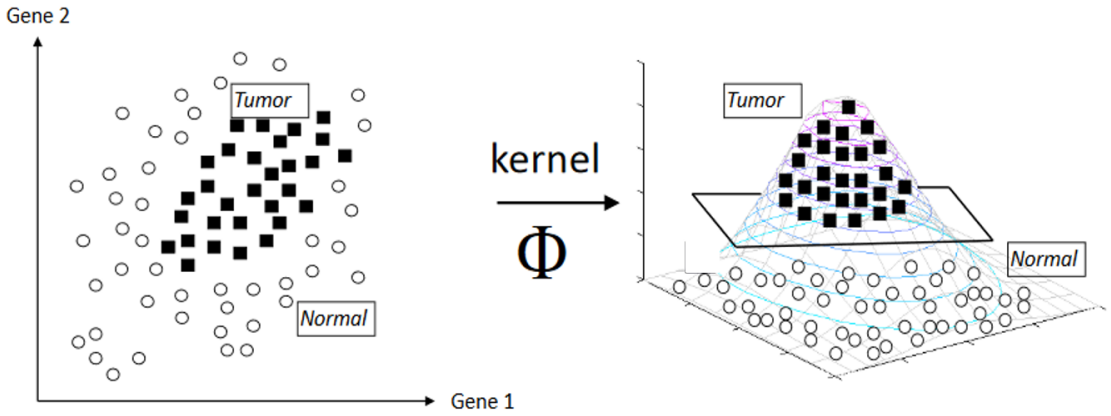


Figura 2.4: Exemplo da aplicação do *Kernel Trick*

3. Previsão numérica SVMs

3.1 Dataset em Estudo

Como base de dados para explorar uma aplicação prática de *SVMs* num problema de previsão/regressão, foi utilizado um conjunto de datasets relacionados com as condições climatéricas (HAVAC).

Os dados recolhidos pela monitorização de um ambiente apresentam a seguinte organização:

- A primeira coluna indica a data em foram realizados os registos da respetiva linha.

  Este atributo apresenta-se no formato Ano/Mês/Dia;

- As restantes colunas, devem ser interpretadas aos pares. Cada par de entradas de uma linha, representa as medições de dois sensores registadas na mesma hora.

  São recolhidas informações relativas à Temperatura (C) da divisão e da humidade

3.2 Exercícios Propostos

Os seguintes exercícios servem como base para compreender os conceitos gerais do funcionamento de um SVM, aplicada num contexto de regressão numérica. Os exercícios foram resolvidos na plataforma R, utilizando a função *SVM* implementada no package *e1071*. Junto de cada resultado será realizada um pequena interpretação teórica dos mesmos, procurando relaciona-los com o funcionamento das SVMs.

3.2.1 Exercício 1

Procedimento:

- Utilizar ficheiro de input *HVAC24hS16-11-2016—0*;
- Realizar a execução da SVM com apenas 6 inputs de treino em vez dos 10 fornecidos.

  Nesse sentido, foram removidos/descartados os primeiros três registos da primeiro e segunda folha de cálculo (folhas com os inputs e outputs de treino);

- O quarto registo de treino é removido da primeira folha e copiado para a terceira, sendo assim usado como input de TESTE;

```
> runSVM("HVAC24hS16-11-2016--0.xls")
       1
524.2618
>
```

Figura 3.1: Resultados obtidos no exercicio 1

Resultados:

Com as alterações referidas acima, reduzindo ainda mais o dataset, a SVMs previu um valor de humidade de 524.2618 para o caso de teste utilizado.

Nos dados iniciais, a mesma instância apresentava um valor de output esperado igual a 525,1277.

Apesar deste exercicio ser um contexto especifico e simples, permite denotar a capacidade de generalização das SVMs, mesmo utilizando poucos casos no processo de aprendizagem/treino.

3.2.2 Exercício 2

Procedimento:

- O procedimento anterior, realizado no exercício 1, foi repetido com 6, 7, 8 e 9 observações de input;
- Em cada execucção, foi calculado o erro percentual médio (MAPE) das previsões.
- As métricas MAPE calculadas foram utilizadas para avaliar a eficácia das previsões. Os valores foram registados e no final calculada a media para as 4 execuções.

```
valoresReais <- c(538.006475880555, 486.103636638889,
↪  498.314949536111, 725.948419627777)
valoresPrevistos <-c(runSVM("HVAC24hS16-11-2016--6.xls"),
                     runSVM("HVAC24hS16-11-2016--7.xls"),
                     runSVM("HVAC24hS16-11-2016--8.xls"),
                     runSVM("HVAC24hS16-11-2016--9.xls"))
mape <- ((valoresReais-valoresPrevistos) / valoresReais)
mediaMape <- mean(mape)
```

Resultados:

A Figura 3.2 demonstra os resultados obtidos para a média da métrica *MAPE* (mean absolute percentage error).

Esta medida de avaliação é bastante utilizada em contexto de dados meteorológicos, sendo assim uma possível forma adequada de quantificar o erro no contexto dos dados em análise.

```
> valoresReais
[1] 538.0065 486.1036 498.3149 725.9484
> valoresPrevistos
[1] 507.1844 499.1777 495.3134 494.6759
> mape <- ((valoresReais-valoresPrevistos)/valoresReais)
> mape
[1]  0.057289414 -0.026895630  0.006023399  0.318579824
> mean(mape)
[1] 0.08874925
>
```

Figura 3.2: Resultados obtidos no exercicio 2

3.2.3 Exercício 3

Procedimento:

- Repetir os passos do exercício 2, experimentando diferentes parametrizações e comparando os erros obtidos;
- Na criação de SVMs foram exploradas diferentes funções de *Kernel*, valores de *Epsilon*;
- Comparar o MAPE médio obtido para as 4 execuções (diferente número de inputs) para as diferentes parametrizações.

Resultados:

Uma vez que o projeto recorrer às SVMs implementadas no package *e1071* para a linguagem R, foram executadas diferentes parametrizações conforme os argumentos apresentado na sua API.

- O valor do parâmetro Kernel indica a função utilizada para projetar os dados num espaço de maiores dimensões.

  Por omissão o package oferece como opções uma função linear, polinomial, radial e sigmoid.

- O argumento *epsilon* indica o valor do parâmetro ϵ da função *insensitive-loss*.

  O valor de ϵ permite aumentar ou atenuar a rigidez do processo de aprendizagem da SVM, controlando a largura da margem na qual são aceites vetores suporte, sem influenciar negativamente a função de *loss*. (O erro é 0 para todos os vetores suporte dentro da margem definida por ϵ - Figura 3.3)

  Se o valor de ϵ for zero, todas as instâncias incorretamente separadas são vistas como erros e penalizam a função de otimização. Para valores de ϵ maiores, aumenta-se a margem na qual se aceitam vetores suporte sem penalização na função de *loss*.

  Por este motivo, a escolha deste valor afeta tanto a complexidade como a capacidade de generalização do algoritmo. Valores próximos de 0 levam a overfitting aos dados de treino e valores mais altos permitem uma maior generalização do algoritmo aos dados, sendo assim mais robusto para previsões futuras com novos casos.

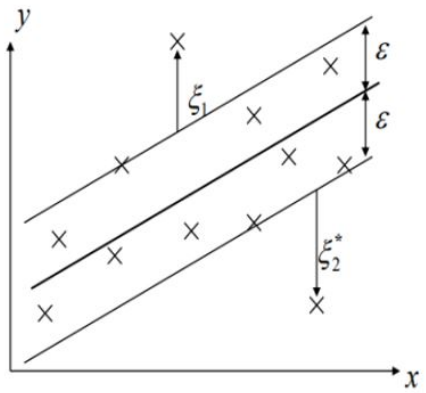


Figura 3.3: Representação gráfica da banda ϵ e dos respetivos erros ϵ_i

Kernel	Linear			Polinomial			Radial			Radial		
Epsilon	0	0.5	1	0	0.5	1	0	0.5	1	0	0.5	1
Mean MAPE	0,082	0,088	0,071	0,068	0,069	0,059	0,088	0,088	0,074	0,074	0,086	0,074

Figura 3.4: Tabela com a sumarização dos resultados obtidos.

Os resultados obtidos (figura 3.4) permitem de facto observar que para uma mesma função de Kernel, quanto maior o valor de epsilon menor o valor da métrica *MAPE*.

Estes resultados verificam que utilizando uma maior folga na margem definida por ϵ, é possível realizar uma aprendizagem mais generalizada e, com isso, obter melhores resultados de regressão para novos casos em que a SVM seja utilizada.

Para o contexto dos dados utilizados, a escolha entre diferentes funções de *Kernel* apresenta aproximadamente os mesmos resultados a nível da métrica de avaliação *Mape*. Contudo, destaca-se o uso da função de kernel polinomial, uma vez que independente do valor de ϵ é a que apresenta os valores *MAPE* mais baixos.

4. Conclusão

No sentido de compreender o modo de funcionamento e conceitos por detrás de *SVMs*, o presente relatório apresenta uma introdução geral das características que suportam estas técnicas de Machine Learning. A introdução inicial, realizada no capitulo 2, oferece assim uma ponte para compreender as interpretações de resultados realizadas sobres os exercícios propostos.

Numa análise geral destes algoritmos, como aspetos que favorecem a utilização de *SVMs* destaca-se o facto de apresentarem um desempenho e resultados muito satisfatórios, mesmo em *datasets* com poucos casos de treino ou com um grande número de atributos.

Na prática, não existe nenhum limite ao número de atributos que a SVM consiga analisar, a não ser possíveis limitações a nível do hardware (memória) usado para computar o algoritmo. Apesar de não abordado neste projeto, técnicas de aprendizagem baseadas em Redes Neuronais tradicionais não apresentam um desempenho tão bom sobre datasets com um grande volume de atributos, em comparação com *SVMs*.

Tal como acontece com outros tipos de classificadores, a aplicação de SVM exige que o conjunto de dados de treino esteja preferencialmente normalizado, estando os atributos dentro do domínio numérico. No caso de atributos categóricos (não numéricos) deve ser criada uma "tabela de conversão"entre os valores iniciais e um mapeamento numérico, contornando assim este problema.

A. Resultados Exercício 2

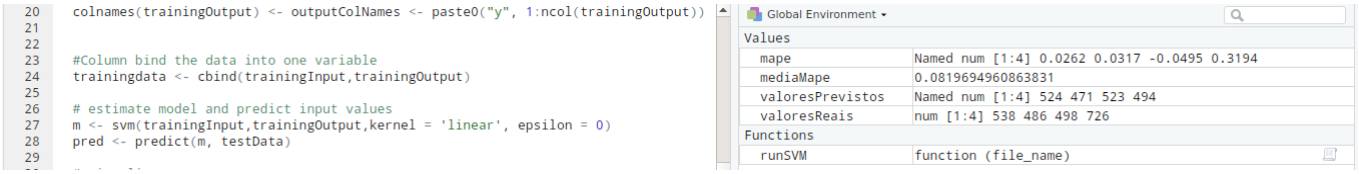


Figura A.1: Execução de SVM com função de Kernel Linear e Epsilon = 0.

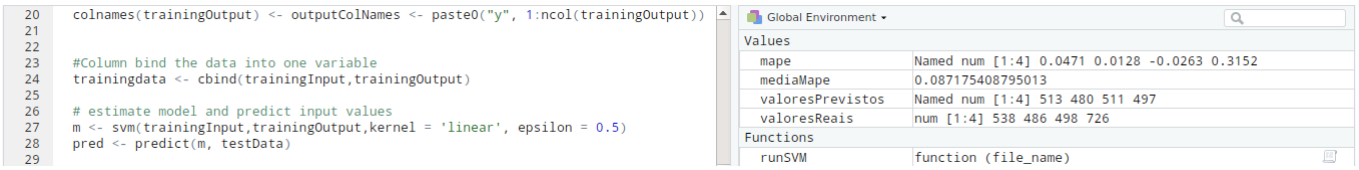


Figura A.2: Execução de SVM com função de Kernel Linear e Epsilon = 0.5.

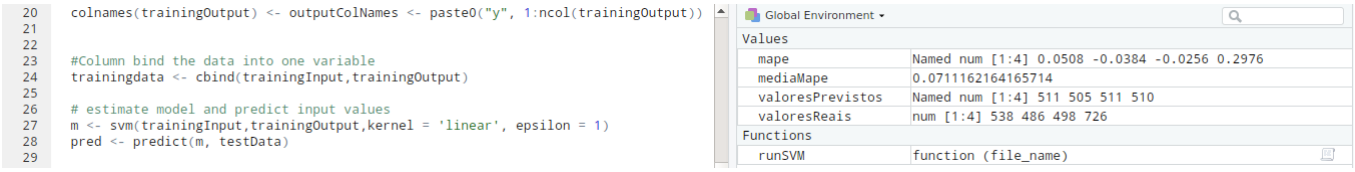


Figura A.3: Execução de SVM com função de Kernel Linear e Epsilon = 1.

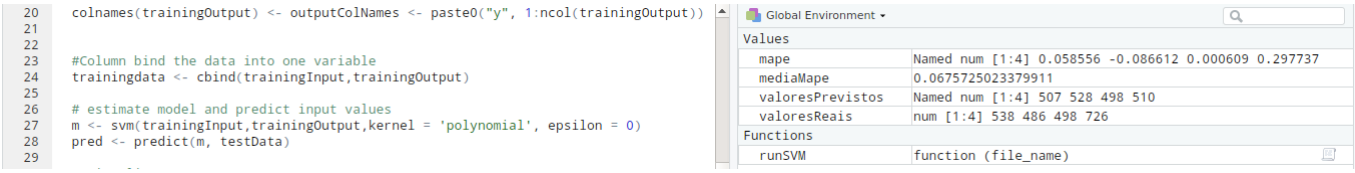


Figura A.4: Execução de SVM com função de Kernel polinomial e Epsilon = 0.

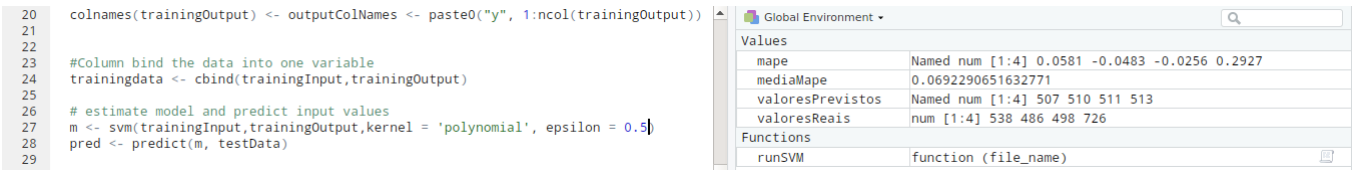


Figura A.5: Execução de SVM com função de Kernel polinomial e Epsilon = 0.5.

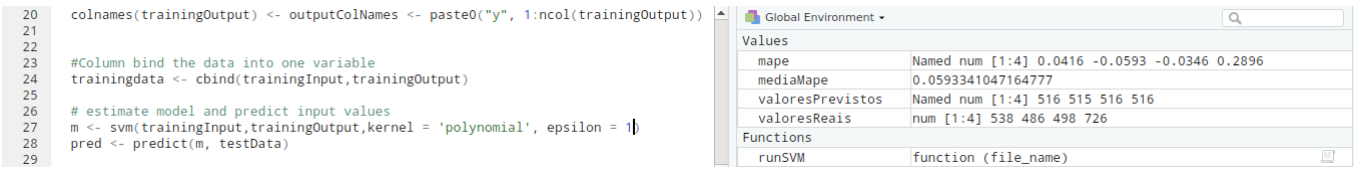
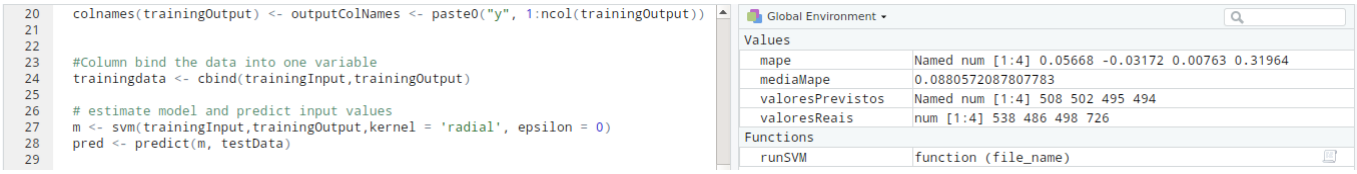


Figura A.6: Execução de SVM com função de Kernel polinomial e Epsilon = 1.

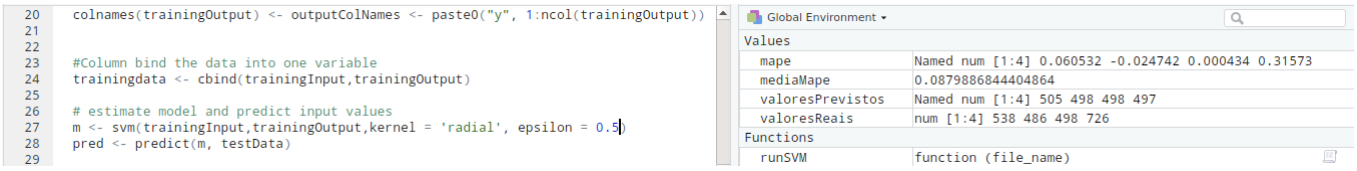


```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'radial', epsilon = 0)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.05668 -0.03172 0.00763 0.31964
mediaMape	0.0880572087807783
valoresPrevistos	Named num [1:4] 508 502 495 494
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.7: Execução de SVM com função de Kernel Radial e Epsilon = 0.

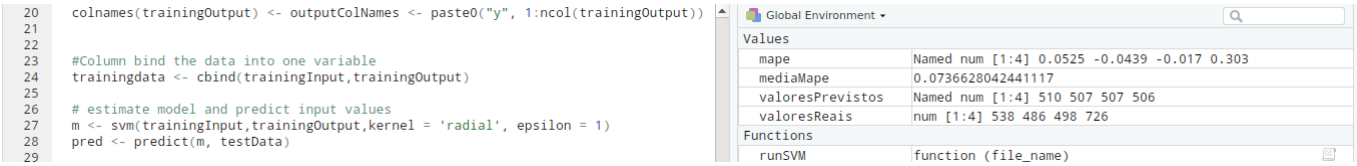


```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'radial', epsilon = 0.5)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.060532 -0.024742 0.000434 0.31573
mediaMape	0.0879886844404864
valoresPrevistos	Named num [1:4] 505 498 498 497
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.8: Execução de SVM com função de Kernel Radial e Epsilon = 0.5.

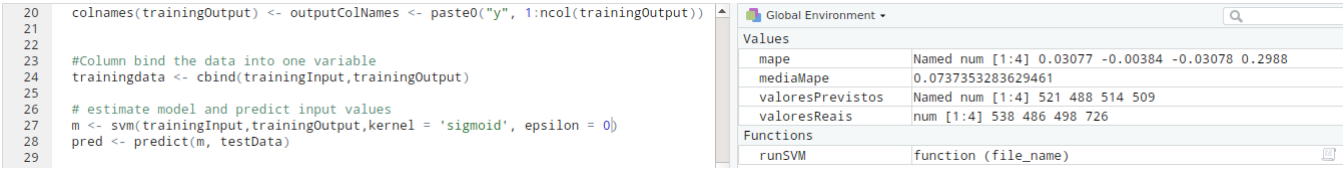


```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'radial', epsilon = 1)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.0525 -0.0439 -0.017 0.303
mediaMape	0.0736628042441117
valoresPrevistos	Named num [1:4] 510 507 507 506
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.9: Execução de SVM com função de Kernel Radial e Epsilon = 1.

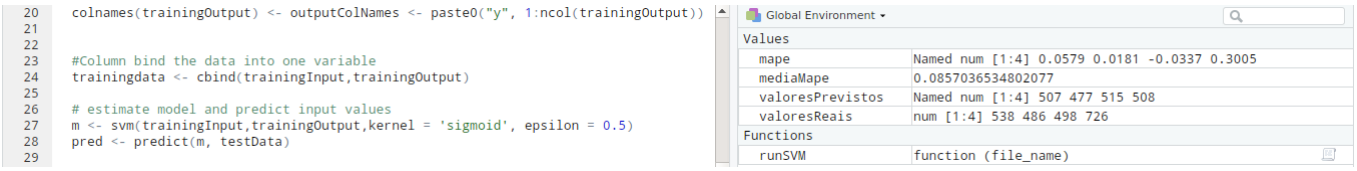


```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'sigmoid', epsilon = 0)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.03077 -0.00384 -0.03078 0.2988
mediaMape	0.0737353283629461
valoresPrevistos	Named num [1:4] 521 488 514 509
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.10: Execução de SVM com função de Kernel sigmoid e Epsilon = 0.

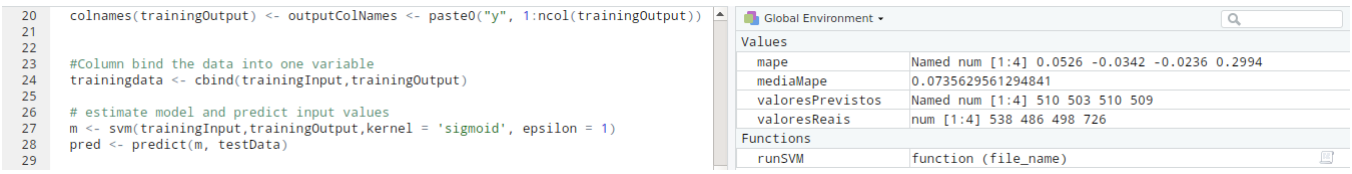


```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'sigmoid', epsilon = 0.5)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.0579 0.0181 -0.0337 0.3005
mediaMape	0.0857036534802077
valoresPrevistos	Named num [1:4] 507 477 515 508
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.11: Execução de SVM com função de Kernel sigmoid e Epsilon = 0.5.



```
colnames(trainingOutput) <- outputColNames <- paste0("y", 1:ncol(trainingOutput))


#Column bind the data into one variable
trainingdata <- cbind(trainingInput,trainingOutput)

# estimate model and predict input values
m <- svm(trainingInput,trainingOutput,kernel = 'sigmoid', epsilon = 1)
pred <- predict(m, testData)
```

Global Environment

Values	
mape	Named num [1:4] 0.0526 -0.0342 -0.0236 0.2994
mediaMape	0.0735629561294841
valoresPrevistos	Named num [1:4] 510 503 510 509
valoresReais	num [1:4] 538 486 498 726
Functions	
runSVM	function (file_name)

Figura A.12: Execução de SVM com função de Kernel sigmoid e Epsilon = 1.

Bibliografia

[1] Svm tutorial. `https://www.svm-tutorial.com/svm-tutorial/`. [Online; accessed 27-Abril-2018].

[2] J. Ben-hur, A. Weston. A user's guide to support vector machines. *Department of Computer Science, Colorado State University*, 2007.

[3] Eulanda Miranda dos Santos. Teoria e aplicação de support vector machine à aprendizagem e reconhecimento de objetos baseados na aparência. *Dissertação (Programa de Pós-Graduação em Informática) – Universidade Federal da Paraíba*, 2002.

[4] Vikramaditya Jakkula. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 20 de Outubro de 2017. [Online; accessed 25-Abril-2018].