Prever se um cavalo pode sobreviver ou não com base em condições médicas anteriores

Ana Paula Falcão[1], **Gabriel Bulhões**[1]

[1]Escola Politécnica – Universidade Federal do Rio de Janeiro
COC361 - Inteligência computacional

1. Introdução

Um cavalo costuma viver entre 25 e 30 anos. Porém, alguns fatores além da idade podem determinar se esse cavalo vai chegar na idade esperada por ele ou não. Se focarmos nas condições passadas médicas desse animal e também considerarmos o que ocorreu com outros equinos, podemos esperar ter uma ideia do futuro que espera para um determinado cavalo. O objetivo desse trabalho é, dado condições médicas de um cavalo, prever se ele irá viver, morrer ou deverá ser eutanasiado. Para tanto, foi utilizado uma base de dados do ano de 1989 encontrado no UCI Machine Learning Repository [1] [7] , o qual possui inicialmente 299 instâncias, 27 atributos e uma variável alvo. Cabe ressaltar que a base contém 30% de valores faltantes, fato esse que será lidado durante o projeto.

2. Dataset e Tecnologia

O significado de cada atributo é bem documentado na fonte na qual foi encontrado os dados [1]. Os tipos dos dados dos atributos presentes são 17 objetos, 7 flot64 e 4 do tipo int64. Ou seja, inspecionando a natureza dos dados, pode-se ver que os dados consistem em 17 características categóricas e os restantes numéricos de 28. A variável alvo (*target*) é a *outcome*, e esta pode ter três valores: *lived, died, euthenized* (em Português viveu, morreu, eutanásia). A visualização da distribuição desses valores do *target* pode ser visto na Figura 1.

Para analisarmos esses dados, foi utilizada a linguagem Python, com bibliotecas que ajudam com essa análise, como Pandas, Sckit-Learn e Numpy.

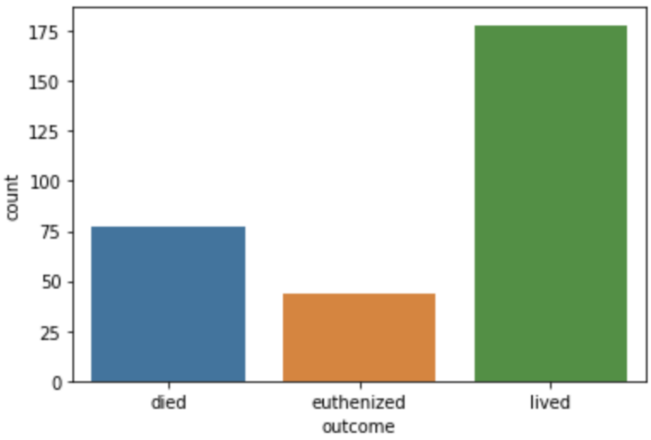

Figura 1. Distribuição dos valores do target outcome

2.1. Informações sobre os atributos

A seguir, são apresentadas as informações acerca dos atributos disponíveis no dataset. São dispostas o nome da variável, uma sucinta descrição e o tipo de dado presente.

1. **Surgery** - se foi realizada cirurgia, sendo 1 *sim* e 2 *não*.
2. **Age** - idade do cavalo, podendo ser Adulto ou Novo.
3. **Hospital Number** - número único do hospital, podendo variar caso o cavalo seja tratado mais de uma vez.
4. **Rectal temperature** - temperatura do reto. Linear em graus Celsius. Uma temperatura elevada pode ocorrer devido à infecção.
5. **Pulse** - pulso, linear. A frequência cardíaca em batimentos por minuto é um reflexo da condição cardíaca.
6. **Respiratory rate** - frequência respiratória linear. A taxa normal é de 8 a 10 - utilidade é duvidosa devido às grandes flutuações.
7. **Temperature of extremities** - temperatura das extremidades. Uma indicação subjetiva de circulação periférica valores possíveis: 1 = Normal 2 = Quente 3 = Frio 4 = Frio extremidades frias a frias indicam possível choque as extremidades quentes devem se correlacionar com uma temperatura retal elevada.
8. **Peripheral pulse** - pulso periférico. Uma indicação subjetiva. Os valores possíveis são: 1 = normal 2 = aumentado 3 = reduzido 4 = ausente
9. **Mucous membranes** - membrana mucosa. Uma medida subjetiva de cor. Os valores possíveis são: 1 = rosa normal 2 = rosa claro 3 = rosa claro 4 = cianótico claro 5 = vermelho brilhante / injetado 6 = cianótico escuro.
10. **Capillary refill time** - tempo de recarga capilar. Um julgamento clínico. Quanto mais tempo o refil, pior a circulação. Valores possíveis 1 = ¡3 segundos, 2 =¿ = 3 segundos.
11. **Pain** - dor, julgamento subjetivo do nível de dor do cavalo. Valores possíveis: 1 = alerta, sem dor 2 = deprimido 3 = dor leve intermitente 4 = dor intensa intermitente 5 = dor intensa contínua.
12. **Peristalsis** - peristaltismo. Uma indicação da atividade no intestino do cavalo. Conforme o intestino se torna mais distendido ou o cavalo torna-se mais tóxico, a atividade diminui. Valores possíveis: 1 = hipermotila 2 = normal 3 = hipomotila 4 = ausente.
13. **Abdominal distension** - distensão abdominal. Valores possíveis 1 = nenhum 2 = leve 3 = moderado 4 = grave.
14. **Nasogastric tube** - sonda nasogástrica. Isso se refere a qualquer gás saindo do tubo. Valores possíveis: 1 = nenhum 2 = leve 3 = significativo.
15. **Nasogastric reflux** - refluxo nasogástrico. Valores possíveis 1 = nenhum, 2 =¿ 1 litro, 3 = ¡1 litro.
16. **Nasogastric reflux PH** - refluxo nasogástrico PH. Linear. Escala é de 0 a 14 com 7 sendo neutro. Os valores normais estão na faixa de 3 a 4.
17. **rectal examination** - exame retal. Valores possíveis 1 = normal, 2 = aumentado, 3 = diminuído, 4 = ausente.
18. **Abdomen** - abdômen. Valores possíveis 1 = normal, 2 = outro, 3 = fezes firmes no intestino grosso, 4 = intestino delgado distendido, 5 = intestino grosso distendido.
19. **Packed cell volume** - volume de células embaladas. Linear. O número de glóbulos vermelhos por volume no sangue.

20. **Total protein** - proteína total. Linear. Quanto maior o valor, maior será a desidratação.
21. **Abdominocentesis appearance** - aparência abdominocentese. Valores possíveis: 1 = claro, 2 = turvo, 3 = serossanguinolento.
22. **Abdominocentesis total protein** - proteína total da abdominocentese. Linear. Quanto mais alto o nível de proteína, maior é a probabilidade de haver um intestino comprometido.
23. **Outcome** - resultado. O que acabou acontecendo com o cavalo? Valores possíveis: 1 = viveu, 2 = morreu, 3 = foi sacrificado.
24. **Surgical lesion** - lesão cirúrgica. Todos os casos são operados ou autopsiados para que este valor e o tipo de lesão sejam sempre conhecidos. Valores possíveis: 1 = Sim, 2 = Não.
25. **Type of lesion** - tipo de lesão. São três atributos desse tipo. Aqui, cada número representa alguma região da lesão, tipo e subtipo da lesão. Os valores presentes aqui são códigos para representar essas lesões.
26. **Cp_data** - estão presentes dados de patologia para este caso? 1 = Sim, 2 = Não. Esta variável não é significativa, uma vez que os dados da patologia não são incluídos ou coletados para esses casos.

3. Metodologia

3.1. Inspeção da correlação entre vários atributos e resultados

A correlação mostra o quão fortemente os atributos estão relacionados uns com os outros. Na Figura 2, pode-se verificar a correlação de cada coluna com o resultado. Se o valor de correlação for positivo, a fetaure está positivamente correlacionada com o resultado. Se o valor de correlação for negativo, o recurso está negativamente correlacionado ao resultado. Se o valor de correlação for 0, duas colunas não estão correlacionadas.

3.2. Remoção de atributos

Em um primeiro momento, foram excluídos alguns atributos, sendo esses: o *cp_data*, pois na própria descrição do dataset já é alertado que não é uma variável significativa, já que os valores da patalogia não coletados para os casos estudados, o *hospital number*, pois representa um id quase único de onde o cavalo foi tratado, e os *type of lesion*, já que são valores que representam um código para determinadas comorbidades.

Na Figura 3 é representada em um gráfico de barras a quantidade de valores nulos em cada atributo. Pode-se notar que alguns atributos possuem uma quantidade alta desse tipo de valor, o que precisa ser tratado.

Assim, foi decidida a remoção de alguns atributos com mais de 50% de valores faltantes, sendo esses 'nasogastric_tube','nasogastric_reflux', 'nasogastric_reflux_ph', 'abdomo_appearance', 'abdomo_protein', 'rectal_exam_feces' e 'abdomen'.

3.3. Lidando com valores nulos

Apesar de terem sido retiradas as colunas com alto percentual de valores faltantes, algumas colunas ainda apresentavam valores nulos, como pode ser visto na Figura 3. Para tanto, foi realizado o método *interpolate*, utilizando a técnica linear, o qual ignora o índice e trata os valores como espaçados igualmente.

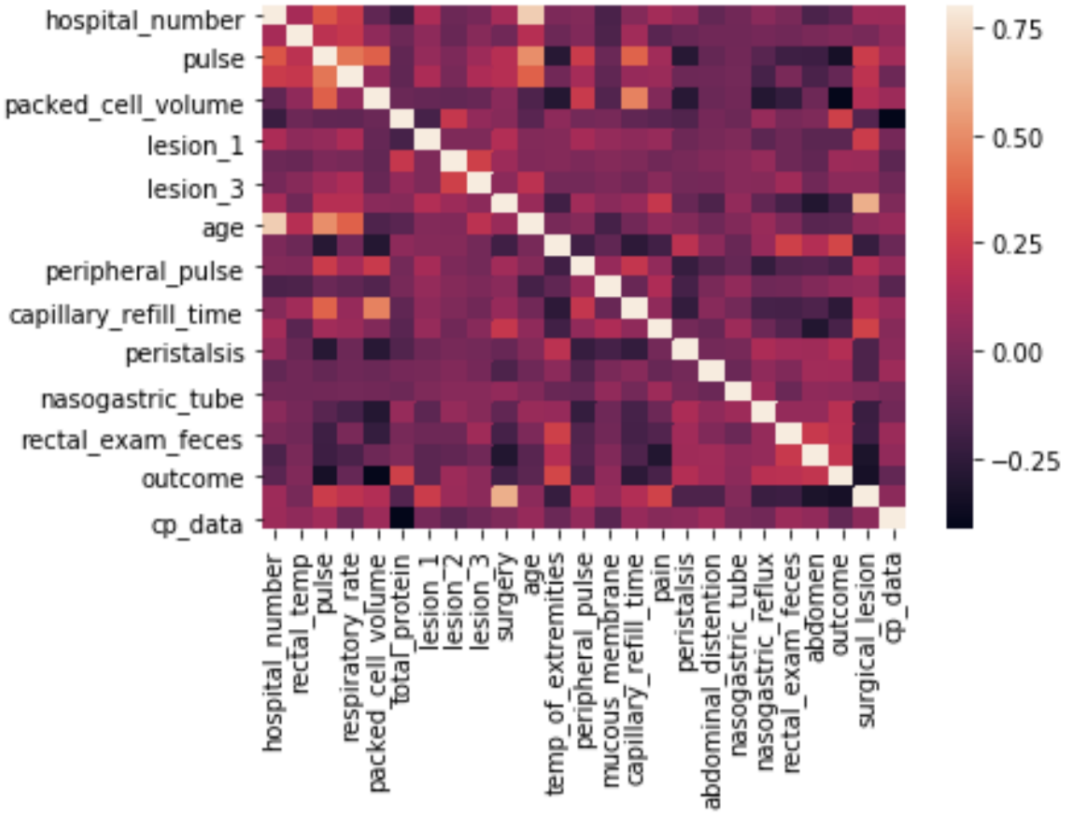

Figura 2. Mapa de calor de correlação

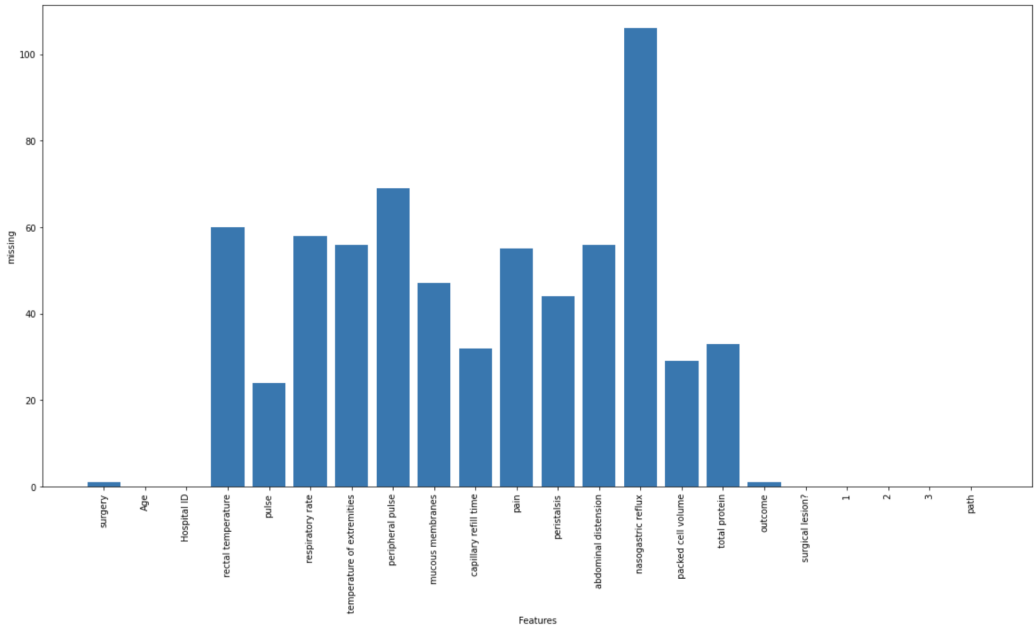

Figura 3. Quantidade de nulos em cada atributo

pulse_absent	pulse_increased	pulse_normal	pulse_reduced
0	0	1	0
0	0	1	0
0	0	1	0
0	0	1	0
1	0	0	0
...	...	...	...
0	0	0	1
0	0	1	0
0	1	0	0
0	0	0	1
0	0	0	1

Figura 4. One hot enconding do atributo Peripheral Pulse

3.4. Lidando com variáveis categóricas

Após a retirada de algumas colunas, ainda restaram 10 atributos categóricos. Essas variáveis são: 'surgery', 'age', 'surgical_lesion', 'temp_of_extremities', 'peripheral_pulse', 'mucous_membrane', 'capillary_refill_time', 'pain', 'peristalsis' e 'abdominal_distention'. Para as três primeiras variáveis, que são binárias, foi aplicado one-hot-encoding, utilizando get_dummies da biblioteca Pandas. O one-hot-encoding transforma um valor em 0 e outro em 1, e, no caso do projeto, o 1 está representando o valor *Não* e 0 o valor *Sim*. No caso da idade, adulto é representado como 1, e jovem como 0.

Para os outros atributos, também foi utilizado o one-hot-encoding, porém de uma maneira um pouco diferente, já que não possuiam apenas dois possíveis valores dentro da coluna. Uma variável temporária recebe a codificação que fica no formato de exemplo da Figura 4, a qual cria quatro colunas para a variável Peripheral Pulse, já que possuiam 4 tipos de valores possíveis para a mesma. Assim, a variável 'peripheral_pulse' é removida, e substituída pelas 4 colunas que representam a codificação dela. Esse processo foi realizado para todos os atributos categóricos que não possuiam valores primariamente binários.

3.5. Ajeitando a escala

A padronização de um conjunto de dados é um requisito comum para muitos estimadores de aprendizado de máquina, já que eles podem se comportar de maneira errônea se os recursos individuais não se parecerem mais ou menos com dados normalmente distribuídos padrão. Então, agora que todas as variáveis são numéricas, precisamos acertar a escala delas. A maioria dos nossos dados são binários, sendo apenas 5 atributos em outra escala, que precisam ser normalizados. Para tanto, foi utilizado o StandardScaler, da biblioteca Scikit-learn. O Standard Scaler padroniza os recursos removendo a média e escalonando

para a variância da unidade, e a pontuação padrão de uma amostra x é calculada como [9]:

$$z = (x - u)/s$$

Onde u é a média das amostras de treinamento ou 0 se 'with_mean = False', e 's' é o desvio padrão das amostras de treinamento ou 1 se 'with_std = False'.

3.6. Diminuição da dimensionalidade

Depois de todos os tratamentos anteriores que foi realizado com o dataset, ficamos com 39 colunas e 176 linhas. O número de elementos de treinamento requeridos para que um classificador tenha um bom desempenho é uma função monotonicamente crescente da dimensão do espaço de características [4]. Isso é conhecido como *curse of dimensionality* (Problema da dimensionalidade). Então, as duas principais razões para que a dimensionalidade seja a menor possível são: custo de medição e precisão do classificador. Quando o espaço de características contém somente as características mais salientes, o classificador será mais rápido e ocupará menos memória. Além disso, quando o conjunto de exemplos de treinamento não é muito grande, como é o caso desse projeto, um espaço de características pequeno pode evitar a maldição da dimensionalidade e propiciar pequenas taxas de erro ao classificador [6].

Para lidar com isso, foi utlizado PCA (Principal Components Analysis), com apoio da biblioteca Scikit-learn. A redução de dimensionalidade linear usando Decomposição de Valor Singular (sigla em inglês, SVD) dos dados é usada para projetá-los para um espaço dimensional inferior. Os dados de entrada são centralizados, mas não escalados para cada recurso antes de aplicar o SVD. Ao chamarmos PCA, precisamos defininir n_components, o qual representa o número de componentes a serem mantidos e o Minka's MLE foi escolhido [8].

Os dados padronizados que obtivemos na Seção 3.5 são então usados para encontrar a matriz de covariância. Precisamos da matriz de covariância para encontrar valores próprios e vetores na próxima etapa. Uma matriz de covariância é usada para analisar a relação linear entre duas variáveis, ou seja, como as variáveis mudam juntas.

```
[0.17703818 0.07028271 0.05970323 0.05089737 0.04835032 0.04313814
 0.04172701 0.04091964 0.03919529 0.03616345 0.03397518 0.0310148
 0.0291708  0.0279958  0.02662241 0.02551003 0.02397729 0.02201847
 0.02036801 0.01954633 0.01832475 0.01637976 0.01495069 0.0140986
 0.01368528 0.01277208 0.01185211 0.00985247 0.00907574 0.00671618
 0.00467791]
```

Figura 5. pca - explained variance ratio

Quando um vetor aleatório é multiplicado por uma matriz de covariância, ele se move na direção de maior variação. Desta forma podemos extrair dimensão com maior dispersão de dados. Uma maneira de fazer isso é encontrar o vetor exato que não converge (ou seja, sua direção não muda), significando que ele já está fornecendo a variação máxima dos dados. Isso é feito calculando os valores próprios e vetores, onde os valores próprios representam a escala do vetor e o vetor próprio representa a direção, dada por

```
[[-2.99126577e+00 -1.21494122e+00 -1.85940663e-01]
 [ 4.37400917e+00  9.89770263e-01  4.41508528e+00]
 [-1.46173579e+00 -3.45889283e-01 -3.88883666e-01]
 [-1.99853241e+00 -2.62154288e+00 -1.94800426e-01]
 [ 1.74920508e+00  8.55733275e-01 -1.13581642e+00]
 [ 1.06629092e-01 -1.45769925e+00  2.13769315e-01]
 [-2.35239706e+00 -1.54517630e+00 -1.62050733e+00]
 [-2.32992203e+00 -7.50223194e-01  1.64944129e+00]
 [-1.00594619e+00  1.92006588e-01 -5.83548214e-01]
```

Figura 6. Resultado que representa os três componentes principais após a multiplicação

Matriz de covariância x Vetor = Componente Escalar x Vetor

onde, Vector é o vetor próprio e o componente escalar é o valor próprio. Uma amostra do resultado pode ser visto na Figura 6

No caso do dataset escolhido, PCA projetou os dados originais que possuiam 39 dimensões em apenas 3 dimensões. Cabe ressaltar que, após a redução da dimensionalidade, geralmente não há um significado particular atribuído a cada componente principal. Os novos componentes são apenas as três principais dimensões de variação. Assim, basta concatenar o resultado do PCA com o *target* outcome, resultando na Figura 7. Nessa Figura, temos a visualização parcial do conjunto de dados com a dimensionalidade reduzida.

PC_1	PC_2	PC_3	outcome
-2.991266	-1.214941	-0.185941	lived
4.374009	0.989770	4.415085	died
-1.461736	-0.345889	-0.388884	lived
-1.998532	-2.621543	-0.194800	lived
1.749205	0.855733	-1.135816	euthanized
...	...	...	...
0.161800	-1.828578	-0.625006	died
-3.682420	2.031823	0.842128	lived
1.711655	0.814602	-2.311713	euthanized
3.015896	0.662283	-1.748541	died
1.260408	-2.798511	-2.229725	lived

Figura 7. Visualização parcial do conjunto de dados com a dimensionalidade reduzida

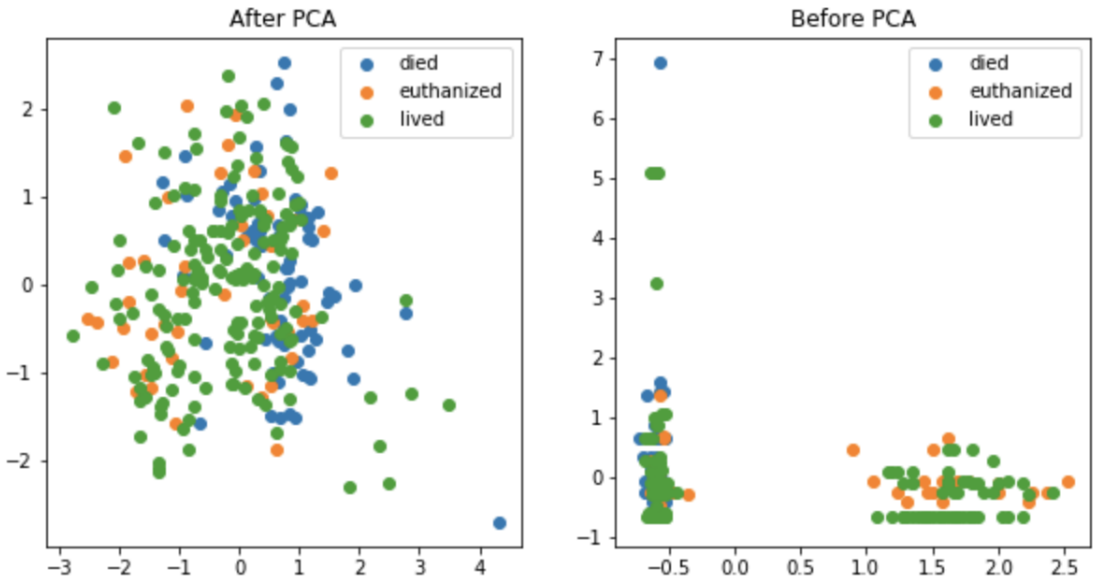

Figura 8. Visualização de dados antes e depois do PCA (Primeiro componente principal vs segundo componente principal)

3.7. Descrição teórica dos modelos utilizados

Todos os modelos foram treinados usando validação cruzada k-fold de 10 ciclos. Sucintamente, esta avaliação separa o conjunto de dados em blocos de tamanhos iguais, sendo a quantidade de blocos igual a quantidade de ciclos. A cada iteração, o bloco de teste muda e no final calcula-se a média dos erros de cada iteração.

3.7.1. Regressão Logística

É um modelo de regressão que permite classificar a saída em classes categóricas. Para começar, é necessário calcular os coeficientes do modelo de tal forma que:

$$y = \beta_0 + \beta_1 x_1 + ... + beta_n x_n$$

Beta é o vetor de coeficiente e x é o vetor de atributos escolhidos. Após isso, para colocar a resposta no intervalo desejado de classificação, deverá ser realizado o seguinte cálculo:

$$p = \frac{exp(y)}{exp(y) + 1}$$

Sendo p o vetor de pontuação dos registros, estima-se as classes baseado nos intervalos definidos de cada uma.

3.7.2. Classificador Naive Bayes

É um classificador probabilístico que toma a decisão considerando uma alta independência entre os atributos. Usando o Teorema de Bayes, conseguimos calcular a pro-

babilidade condicional usando:

$$p(C_x|x) = \frac{p(C_x)p(x|C_k)}{p(x)}$$

Sendo:

- $p(C_k|x)$ a probabilidade a posteriori
- $p(x|C_k)$ a distribuição de probabilidade condicional
- $p(C_k)$ a probabilidade a priori
- $p(x)$ o termo de padronização

Este classificador usou uma distribuição gaussiana, tal que:

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\theta^2}e^{-\frac{(v-m_k)^2}{2(\theta_k)^2}}}$$

3.7.3. Árvore de decisão

É um classificador que possui apenas dois elementos: nós e folhas. O nós são testes condicionais de um atributo, ditando o caminho do algoritmo em duas vias caso seja positivo ou negativo. As folhas são o ponto final do classificador, quando o modelo diz qual é a saída calculada. Um exemplo construído com scikit-learn em Python pode ser visto na Figura 9.

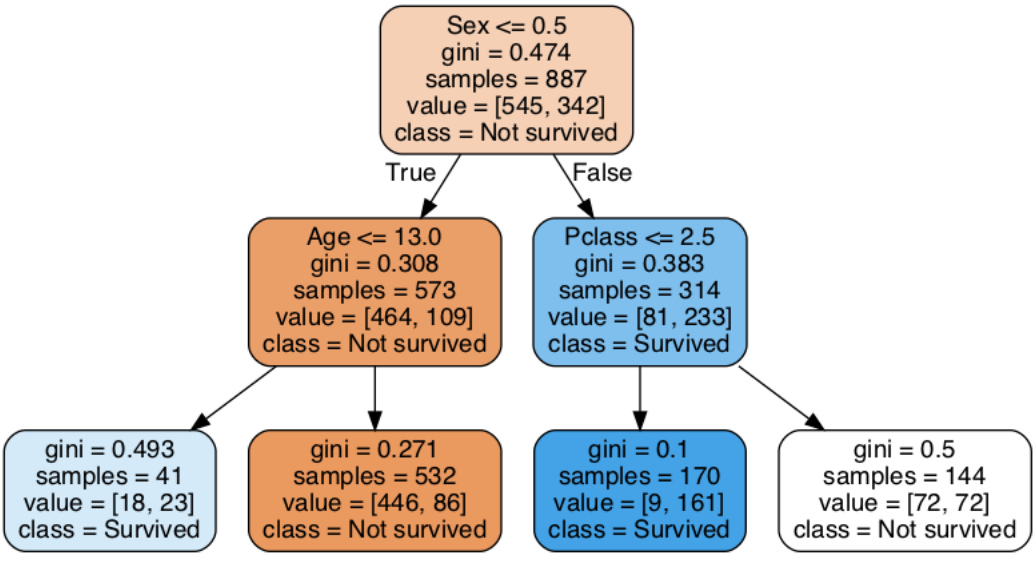

Figura 9. Exemplo de uma Decision Tree. Fonte: [3]

3.7.4. Random Forest

Random Forest como o próprio nome indica, consiste em um grande número de árvores de decisão individuais que operam como um conjunto. Cada árvore individual na floresta

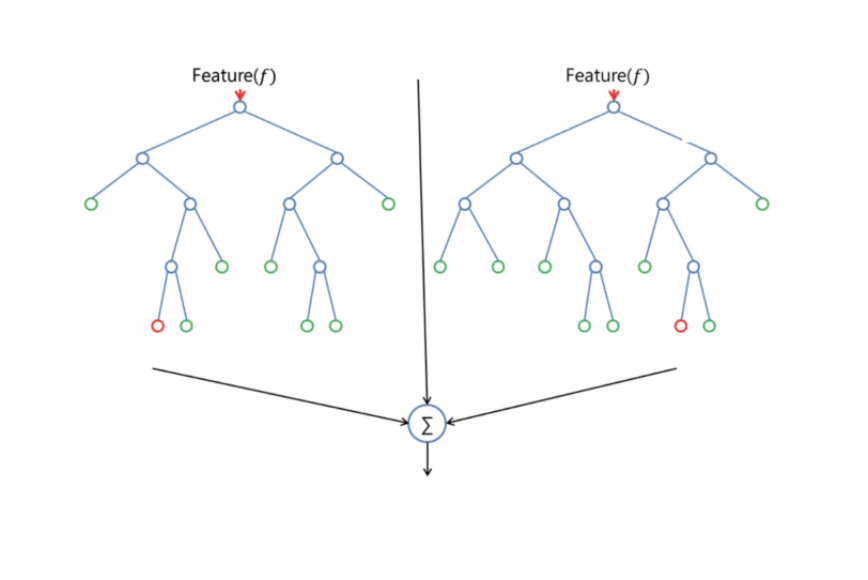

Figura 10. Exemplo de Random Forest com duas árvores. Fonte: [2]

aleatória exibe uma previsão de classe e a classe com mais votos torna-se a previsão do nosso modelo. É um algoritmo de aprendizado supervisionado.

A "floresta" que constrói é um conjunto de árvores de decisão, geralmente treinadas com o método de "ensacamento". A ideia geral do método de bagging é que uma combinação de modelos de aprendizagem aumenta o resultado geral. Um exemplo de uma random forest com duas árvores pode ser vista na Figura 10.

3.7.5. Gradient Boosting

Gradient Boosting treina muitos modelos de maneira gradual, aditiva e sequencial. Identifica as deficiências com o uso de gradientes na função de perda ($y = ax + b + e$, sendo e o termo de erro). A função de perda é uma medida que indica quão bons são os coeficientes do modelo no ajuste dos dados subjacentes. Uma compreensão lógica da função de perda dependeria do que estamos tentando otimizar.

Envolve três elementos:

1. Uma função de perda a ser otimizada.
2. Um weak learner para fazer previsões.
3. Um modelo aditivo para adicionar weak learners para minimizar a função de perda.

3.7.6. SVM

Através de um hiperplano, este modelo de classificação com aprendizado supervisionado separa as amostras mais próximas de ambas as classes utilizando um vetor de suporte. Um simples exemplo ilustra esse cenário na Figura 11:

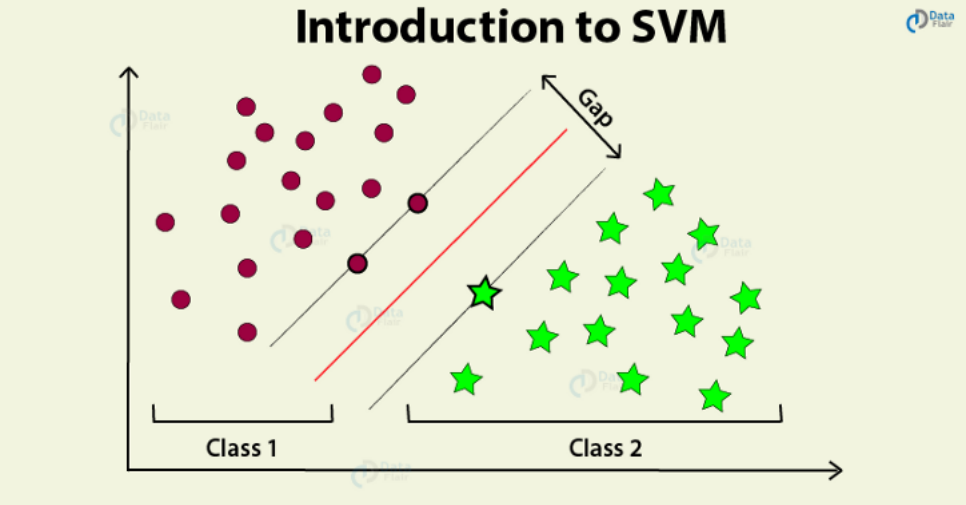

Figura 11. Exemplo de SVM. Fonte: [5]

Neste trabalho foi usada a SVM Linear. Além disso, após resultados iniciais ruins, também foi usada Soft Margins, o que ignora algumas amostras de difícil classificação mas consegue tenta prever corretamente o resto.

3.7.7. Rede Neurais Artificiais

Algoritmo mais usado na área de aprendizado de máquina. Por meio de uma rede de camadas interligadas por neurônios, o algoritmo reconhece os padrões nos dados e ajusta os pesos desses neurônios a cada amostra, criando um modelo de classificação poderoso para muitos tipos de problemas.

Neste caso, foi usado apenas uma camada intermediária com 10 neurônios. A função de ativação foi a Sigmoid, calculada a seguir:

$$f(x, a, c) = \frac{1}{1 + e^{-a(x-c)}}$$

4. Resultados

4.1. Modelos lineares

4.1.1. Regressão Logística

4.1.2. Classificação Bayesiana

4.2. Modelos não-lineares

4.2.1. Árvores de decisão

4.2.2. Random Forest

4.2.3. Gradient Boosting

4.2.4. SVM

4.2.5. Redes Neurais

5. Conclusões

Referências

[1] Mary McLeish Matt Cecile. Horse colic data set, 1989.

[2] Niklas Donges. A complete guide to the random forest algorithm, 2019.

[3] Mikkel Duif. An introduction to decision trees with python and scikit-learn, 2020.

[4] Jain et al. Statistical pattern recognition: A review., 2000.

[5] ichi.pro. Introdução ao support vector machine (svm), Acesso em: Junho, 2021.

[6] IME-USP. Teofilo emidio de campos, 2000-09-18.

[7] UCI Machine Learning. Horse colic data set, 2017.

[8] Thomas P Minka. Automatic choice of dimensionality for pca, 2000.

[9] scikit learn. sklearn.preprocessing.standardscaler, Acesso em: Maio, 2021.