Compra de Carros Feita por Classificadores de Aprendizado de Máquina

Christoffer de Paula Oliveira
Departamento Ciência da Computação, UFSJ, São João del Rei

1 Introdução

Cada vez mais encontramos em nosso cotidiano pessoas com dúvidas na hora de comprar um veículo. Removendo coisas mais complexas como as novas tecnologias contidas nos carros temos algumas características comuns que os consumidores buscam, como por exemplo o custo de manutenção do veículo. Dadas as características podemos utilizar classificadores de aprendizado de máquinas para auxiliar no processo de escolha da compra de um veículo. Temos antes que trabalhar na base de dados, ao contrário da base dados flores Iris, uma base de dados amplamente conhecida em Machine Learning, onde as características são números temos características escrita como 'Muito Boa', 'Boa' e etc. Uma vez trabalhada podemos utilizar dos algoritmos para obtermos uma melhor classificação, para comparação de resultados foi implementado o algoritmo KNN sem o uso de bibliotecas prontas.

1.1 Base de Dados

Nossa base de dados consiste em algumas características que consumidores observam antes de efetuar a compra de um veículo. São elas:

- Preço da Venda
- Preço de Manutenção
- Número de Portas
- Capacidade de Pessoas
- Tamanho do Porta Malas
- Segurança

Temos um total de 1728 veículos que foram avaliados, onde são classificados como: Inaceitável, aceitável, bom e muito bom, em relação as características apresentadas. Falando um pouco mais sobre as características dos veículos, cada característica tem sua forma de peso.

Características	**Pesos**			
Preço Venda	Muito Alto	Alto	Médio	Baixo
Preço Manutenção	Muito Alto	Alto	Médio	Baixo
Número de Portas	2	3	4	5 ou mais
Cap. Pessoas	2	4	mais	
Tam. Porta Malas	Pequeno	Médio	Grande	
Segurança	Baixa	Média	Alta	

Tabela1: Características com seus possíveis pesos

2 Fundamentação Teórica

2.1 K-Nearest Neighbors

A classificação com Algoritmo K-Nearest Neighbors é um tipo de aprendizado baseado em instância ou de aprendizado não generalizante: esta técnica não tenta construir um modelo interno geral, mas simplesmente armazena instâncias dos dados de treinamento [4].
A classificação é calculada com base em uma votação dos k-vizinhos mais próximos de cada ponto (cada instância de dados ou exemplo de treinamento são vistos como pontos no espaço). Após a classificação de um novo ponto (instância), o algoritmo utiliza este novo ponto como vizinho para a classificação de outros novos pontos que podem eventualmente se tornar seus vizinhos.

2.2 Support Vector Machines

O algoritmo Support Vector Machine (SVM) induz um modelo que representa os exemplos (instâncias) como pontos no espaço, mapeados de maneira que os exemplos de cada categoria (classe) sejam divididos por um espaço claro que seja tão amplo quanto possível. Os novos exemplos são então mapeados no mesmo espaço e preditos como pertencentes a uma categoria de acordo com o lado do espaço onde estão localizados [5].
Em outras palavras, o algoritmo SVM busca encontrar uma linha de separação entre os dados das classes. Essa linha é chamada de hiperplano e ela busca principalmente maximizar a distância entre os pontos próximos em relação a cada uma das classes.

2.3 Redes Neurais Artificiais

Redes Neurais Artificiais são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. Uma grande rede neural artificial pode ter centenas ou milhares de unidades de processamento [6].
A propriedade mais importante das redes neurais é a habilidade de aprender [6]. Com base nas experiências obtidas por cada iteração na fase de treinamento, é feito o ajuste dos pesos em seus neurônios. A partir destes pesos calculados, a rede neural estará pronta para realizar a classificação dos dados.

2.4 Naive Bayes

Os métodos Naive Bayes são um conjunto de algoritmos de aprendizado supervisionados de redes bayesianas, baseados na aplicação do teorema de Bayes, com a suposição "ingênua" de independência condicional entre as variáveis de entrada, dado os valores da variável de classe [7]. O teorema de Bayes afirma o seguinte: dada a variável de classe y e vetor de variáveis de entrada x1, ..., xn , o valor da variável classe y é calculado de acordo com a Eq.1:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \qquad (1)$$

onde P(y | x1, ..., xn) é a probabilidade a posteriori da variável classe, dado os valores dos atributos x1,..., xn; P(y) é o valor da probabilidade à priori da variável classe e P(x1, ..., xn) é a probabilidade à priori dos atributos x1, ..., xn.
Existem diferentes versões dos classificadores ingênuos de Bayes e elas diferem-se principalmente pelas suposições que fazem sobre a distribuição P(xi | y) [7]. Apesar de suas suposições aparentemente simplificadas, os classificadores ingênuos de Bayes têm funcionado muito bem em muitas situações do mundo real [7]. As subseções 2.4.1, 2.4.2, 2.4.3 e 2.4.4 apresentam, respectivamente, as seguintes versões do Naive Bayes utilizadas neste projeto: o Naive Bayes Gaussian, o Naive Bayes Multinomial, o Naive Bayes Complement e o Naive Bayes Bernoulli.

2.4.1 Naive Bayes Gaussian

O algoritmo Naive Bayes Gaussian calcula a probabilidade das características (variáveis) utilizando uma função gaussiana (Eq.2):

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \qquad (2)$$

Os parâmetros σy e μy são estimados usando a máxima verossimilhança [7].

2.4.2 Naive Bayes Multinomial

O algoritmo Naive Bayes Multinomial (MNB) é uma das duas variantes ingênuas Bayes usadas na classificação de texto. A distribuição é parametrizada por vetores para cada y, onde n é o número de características e θyi é a probabilidade de P(xi | y) do valor de xi aparecer em uma amostra pertencente a classe y.
O parâmetro θy é estimado por uma versão suavizada de máxima verossimilhança (Eq.3), ou seja, contagem relativa de frequências (Eq. 3):

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \qquad (3)$$

onde $N_{yi} = \sum_{x \in T} x_i$ é o número de vezes que o valor i aparece em uma amostra de classe y no conjunto de treinamento T. N_y é a contagem total de todos os valores para a classe y [7].

3. Metodologia, Experimentos e Análise de Resultados

Inicialmente, houve um estudo sobre algumas bibliotecas de Machine Learning as quais implementam algoritmos de Aprendizado de Máquina, como Scikit-Learn[1] para diversos algoritmos de classificação e Tensorflow[2] para Redes Neurais Artificiais. Utilizando o Scikit-Learn, foi feito um estudo inicial envolvendo dois algoritmos de classificação: o SVM (Support Vector Machine) e o KNN (K-Nearest Neighbors).

3.1 Base de Dados

Após estudos das bibliotecas e entender a implementação iniciou-se o trabalho voltado para a base de dados, como em

[1] https://scikit-learn.org/stable/index.html
[2] https://tensorflow.org

nossa base de dados tínhamos valores unitários diferentes para representar as características, ver Tabela1, foi feito inicialmente uma padronização dos dados, trocando os valores de frases, para números, e esses números seriam os pesos para a execução dos algoritmos. Ficamos então com as seguintes trocas:

Característica	Número equivalente
Muito Alto	4
Alto	3
Médio	2
Baixo	1
5 ou mais	5
mais	5
Pequeno	1
Grande	3

Tabela2: Padronização de Características

Essa transformação foi feita com devido cuidado para não atribuir o mesmo valor para dois pesos da mesma característica, ou seja, apesar de os números se repetirem, eles se repetem em características diferentes, portanto, usando-se dos algoritmos de classificação obtemos resultados que podem ser utilizados para chegar em conclusões.

Apesar da base de dados estar parecer estar pronta, temos ainda que modificar a classificação, como supracitado nossa base de dados permite as seguintes classificações: Inaceitável, aceitável, bom e muito bom, em relação as características apresentadas. Essas possíveis classificações ainda necessitam passar por uma padronização, para que os algoritmos aprendam somente com números os resultados verdadeiros também precisam ser números. Ficamos então com a seguinte configuração para as possíveis classificações:

Características	Número Equivalente
Inaceitável	1
Aceitável	2
Bom	3
Muito Bom	4

Tabela3: Padronização das Classificações

Temos agora um conjunto total de 1728 veículos em um único arquivo, onde cada veículo pode ser representado em um vetor de tamanho 7, isto é, com todas suas características mais sua classificação verdadeira, também chamada de alvo. Um carro até este momento está representado da seguinte maneira. Veja tabela1 para melhor entendimento dessa parte de representação dos carros:

- Preço Venda[0]
- Preço Manutenção[1]
- Número de Portas[2]
- Capacidade de Pessoas[3]
- Tamanho do Porta Malas[4]
- Segurança[5]
- Alvo[6]

Onde os números entre colchetes representam as posições das características no vetor, observe que todas essas características agora são números nos arquivos, por exemplo:

O carro sendo 1, 1, 1, 1, 1, 1, 1 é mesmo que dizer que este carro possui:

- Preço de Venda: Baixo
- Preço Manutenção: Baixo
- Número de Portas: 1
- Capacidade de Pessoas: 1
- Tamanho do Porta Malas: Pequeno
- Segurança: Baixa
- Classificação: Inaceitável

Para fazer essa transformação basta pegarmos a posição do vetor, que representa uma característica, olhar na Tabela1 os possíveis valores da característica e por fim olhar na tabela 2 ou 3, dependendo ser for a última posição, para saber o que o número representa em tal característica.

Assim, temos os vetores de carros e seus alvos prontos para serem colocados nos algoritmos, para a realização de um teste mais aleatório foi realizado uma mistura dos dados, mesmo que os dados já estavam misturados, dessa forma podemos sempre ter uma base de dados com ordens diferentes, para a realização do aprendizado e dos testes.

Para a realização da mistura, pegamos todo o banco de dados e separamos em mais 4 arquivos, assim, temos um arquivo para cada classificação possível, veja Tabela3. Uma vez com estes arquivos, temos que agora que montar alguns arquivos em, onde a ordem dos dados nesses arquivos são aleatórias. Os arquivos são, separados em Carros, Alvo dos Carros, Testes e Alvo dos Testes. Então agora temos um arquivo para treino com seus alvos em outro arquivo, e um arquivo para teste com seus alvos em outros arquivos. Com estes dados separados dessa forma, começamos a realização das execuções dos algoritmos.

Depois de todos os processos de padronização dos dados obtivemos as seguinte configuração em termos de Teste e Treino, veja Tabela4:

Classificação	Total	Treino	Teste(30%)	Alvo
Inaceitável	1210	847	363	1
Aceitável	284	268,8	115,2	2
Bom	69	48,3	20,7	3
Muito Bom	65	45,5	19,5	4
Total	1728	1211	517	

Tabela4: Configurações dos Dados

Na existência de números quebrados, foram feitos arredondamentos priorizando o treino, isto é, alguns dados tem um pouco menos que 30% para testes.

3.2 Experimentos Iniciais e Discussão de Resultados

Para melhor comparação de resultados foram avaliados dois elementos importantes, são eles a Acurácia e o Tempo de execução. Ainda em comparação foi implementado um Algoritmo KNN, então ficamos com dois algoritmos KNN e os demais algoritmos explicado no tópico de Fundamentos Teóricos.

Os parâmetros de alguns algoritmos foram definidos de forma empírica, utilizando a função randomizedsearchCV()

disponível na biblioteca Scikit-learn. Assim, o KNN obteve melhores resultados com o número de vizinhos k=2. Foi utilizada uma rede neural de múltiplas camadas, contendo 5 camadas ocultas e 100 neurônios em cada. Para os outros algoritmos aplicados nos experimentos, os parâmetros permaneceram com os valores padrão (default) definidos na biblioteca.

Algoritmos	Acurácia (%)	Tempo de Execução (seg)
Support Vector Machines (SVM)	72	0.029
K-Neares Neighbors (KNN)	79	0.0079
Rede Neural MLP	68	3.49
Naive Bayes Gaussian	70	0.00099
Naive Bayes Multinomial	70	0.00099
K-Neares Neighbors (KNN) (Implemetado)	38	2.07

Tabela4: Resultado Obtidos pelo Algoritmos de classificação

A tabela 4 mostra os resultados obtidos pelos algoritmos de classificação, como podemos perceber o algoritmo que se mostrou mais eficiente para essa base de dados foi o algoritmo KNN (79% de certo), além disso seu tempo de execução não ultrapassou 0,008 se mostrando melhor que o SVM que teve tanto acurácia quanto tempo menos desejáveis e também melhor que a Rede Neural que assim como o SVM são técnicas mais complexas e demandam mais tempo para gerar o classificador, assim o KNN fica em terceiro lugar em relação ao tempo. Por sua vez temos os algoritmos Navei Bayes, ambos tanto o Gaussiano e o Multinominal, se mostraram com um tempo extremamente pequeno, e sua precisão de acerto se mostrou em 70% pode parecer uma eficiência não muito boa a primeiro instante, entretanto, estes classificadores não retornam apenas a classe predita. Eles retornam também o valor de probabilidade de todas as classes (a classe predita possui o maior valor de probabilidade). E com um desempenho menos eficiente entre os demais algoritmos ficamos com o KNN implementado, esse algoritmo se mostrou com uma acurácia bem inferior aos demais algoritmos mas ainda sim seu tempo de execução (2.07s) se mostrou inferior ao empo de execução da Rede Neural que foi de quase 3,5s.

4. Conclusão

Neste trabalho é proposto algoritmos de e métodos de classificação que possam auxiliar os consumidores ao comprar um veículo, podendo investigar uma grande quantia de ofertas e vendas em um curto espaço de tempo, e falando de tempo podemos ressaltar ainda ao utilizar dos algoritmos Naive Byes temos uma grande velocidade de execução mostrando ser bastante eficientes quando necessário tempo e além disso eles retornam as probabilidades de todas as classes, podendo fazer predições dos dados.

Podemos também afirmar que os algoritmos fornecidos pelas bibliotecas utilizadas são bastante otimizados para resolverem o problema, isso pode ser visto se compararmos o resultado dos algoritmos KNN da biblioteca e KNN implementado. O tempo de execução e acurácia se mostraram muito inferiores em comparado com o algoritmo da biblioteca Scikit-learning.

Os experimentos forneceram uma resultados para a comparação de diferentes classificadores, mostrando que cada algoritmo se comporta de maneira diferente dado o mesmo problema.

Na literatura e no cotidiano das pessoas cada vez mais os algoritmos se aprendizado de máquinas vem se mostrando presentes, neste trabalho mostra-se que essas algoritmos podem resolver não só problemas complexos mas também pequenos problemas do cotidiano das pessoas e é notável que cada vez mais veremos crescimento dessa área tecnológica.

3 Referências

[1] Paulo, A. M.; Schiavoni, F. L.; Laia, M. A. M.; Madeira, D. L. A. Copista: Sistema de OMR para a recuperação de acervo histórico musical. In: Simpósio Brasileiro de Computação Musical, 2015, Campinas. Proceedings of the 15th Brazilian Symposium on Computer Music. Porto Alegre: Brazilian Computer Society, v. 1, p. 48-59, 2015.

[2] Rebelo, A., Fujinaga, I., Paszkiewicz, F., Marcal, A., Guedes, C., and Cardoso, J. Optical music recognition: state-of-the-art and open issues. International Journal of Multimedia Information Retrieval, 1(3):173–190, 2012.

[3] Bainbridge, D. and Bell, T. The challenge of optical music recognition. Computers and the Humanities, 35(2):95–121, 2001.

[4] Nearest Neightbors. Scikit-learn, 2018. Disponível em: <https://scikit-learn.org/stable/modules/neighbors.html#classification>. Acesso em: 10 Dez. de 2018.

[5] Support Vector Machine. Scikit-learn, 2018. Disponível em: <https://scikit-learn.org/stable/modules/svm.html#svm-classification>. Acesso em: 10 Dez. de 2018.

[6] Koehrsen, William. Classificador de Rede Neural Profunda. Medium, 2018. Disponível em: <https://medium.com/@williamkoehrsen/deep-neural-network-classifier-32c12ff46b6c>. Acesso em: 10 Dez. de 2018.

[7] Naive Bayes. Scikit-learn, 2018. Disponível em: https://scikit-learn.org/stable/modules/naive_bayes.html. Acesso em: 10 Dez. de 2018.