Avaliação Automotiva por Algoritmos de Classificação

Marcello Fonseca de Oliveira

Universidade Federal de São Paulo
marcello.fonseca27@unifesp.br
Av. Cesare Monsueto Giulio Lattes, 1201

***Abstract*— This document gives the information and results behind multiple algorithms that analyze a car evaluation dataset. Those algorithms predict if the vehicle is worthy to afford based on their characteristics.**

***Palavras-chave*— Classificadores, Inteligência Artificial, KNearest Neighbors, Random Forest, Árvore de Decisão, Dataset.**

I. Introdução

O aprendizado de máquina é uma poderosa ferramenta de inteligência artificial que vem sendo extremamente usada nos dias atuais a fim de designar tarefas à algoritmos diversos. A importância dessa técnica se deve ao fato de que, a níveis computacionais, esse é o mais próximo que podemos chegar da capacidade interpretativa humana. Isso se dá pelo fato de que é apresentado ao algoritmo dados pré-processados que fornecem informações que contribuem para o aprendizado e tomada de decisão de acordo com o contexto almejado.

Baseando-se nisso, neste trabalho foram utilizadas três técnicas de aprendizagem de máquina, a fim de predizer, de acordo com as características do automóvel, qual carro é o mais adequado para se adquirir. Estas técnicas são: Árvores de decisão, Random Forest Classifier e KNearest Neighbors Classifier. A meta da utilização de mais de uma técnica é verificar qual delas apresenta resultados mais satisfatórios acerca dos automóveis dadas diversas execuções de cada algoritmo de classificação.

II. Tratamento dos dados

A fim de utilizar os métodos de classificação, foi necessário entender e tratar os dados anteriormente.

A. Entendimento do dataset

A base de dados de automóveis possui 6 atributos que compõem a classificação final do mesmo, sendo eles:

- Valor de compra (vhigh, high, med, low)
- Manutenção (vhigh, high, med, low)
- Número de portas (2, 3, 4, 5more)
- Lugares (2, 4, more)
- Capacidade do porta-malas (small, med, big)
- Segurança (low, med, high)

Esses atributos contribuem para a concepção da classificação final de cada automóvel, a qual é dada na base de dados por:

- Unacceptable (Inaceitável)
- Acceptable (Aceitável)
- Good (Bom)
- Very Good (Muito Bom)

A partir da biblioteca “pandas” foi verificado que todos atributos se tratavam de variáveis categóricas.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   preco       1727 non-null   object
 1   manutencao  1727 non-null   object
 2   portas      1727 non-null   object
 3   capacidade  1727 non-null   object
 4   p_malas     1727 non-null   object
 5   seguranca   1727 non-null   object
 6   classe      1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
None
```

Fig 1 - Utilização da função info() da biblioteca pandas.

Além disso, também foi previamente constatado que dentre as classes, a Inaceitável foi a mais

frequente de forma considerável.

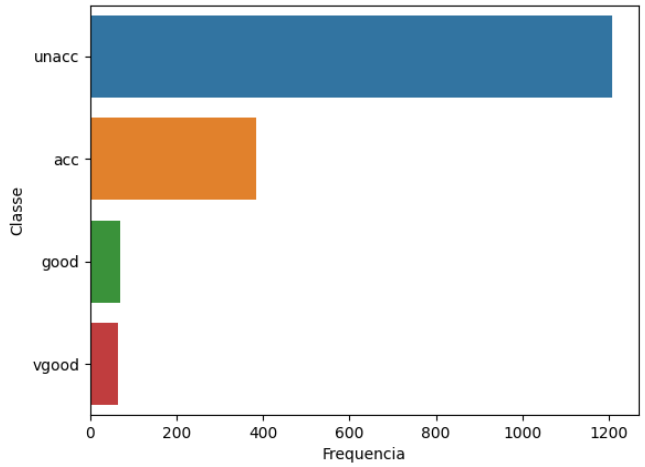

Fig. 2 - Distribuição das classificações dos automóveis.

B. Formatação dos dados

Dado que as variáveis são categóricas, temos a necessidade da utilização de um encoder que possa padronizá-las. Então, a partir da biblioteca category_encoders e de sua função OrdinalEncoder(), a qual foi aplicada em todo dataset, as variáveis e as classes foram normalizadas em números inteiros. Após a normalização, os dados foram divididos entre 33,33% para teste e 66,67% para treino.

<bound method NDFrame.head of

	preco	manutencao	portas	capacidade	p_malas	seguranca	classe
0	1	1	1	1	1	1	
1	1	1	1	1	2	1	
2	1	1	1	1	2	3	1
3	1	1	1	1	2	1	1
4	1	1	1	1	2	2	1
...	...	...	...	...	...	...	...
1722	4	4	4	3	2	1	4
1723	4	4	4	3	2	2	3
1724	4	4	4	3	3	3	1
1725	4	4	4	3	3	1	4
1726	4	4	4	3	3	2	3

Fig. 3 - Formato dos dados já normalizados.

III. Metodologia

Com os dados normalizados, agora é possível aplicar os métodos de classificação e verificar os resultados. Mas antes é importante entendê-los.

A. KNN (K-Nearest Neighbors)

O KNN é um método de regressão/classificação de aprendizagem de máquina supervisionado, o qual usa da proximidade dos dados vizinhos para realizar suas classificações e predições. O algoritmo calcula a distância do elemento que está sendo analisado dos outros elementos do dataset, ordenando as distâncias de menor para o maior, e assim, verifica a classe dos K primeiros elementos dessa lista. Nesse estudo, o valor do K foi dado pela raiz quadrada do número de automóveis avaliados, a fim de obter maior variação dos dados.

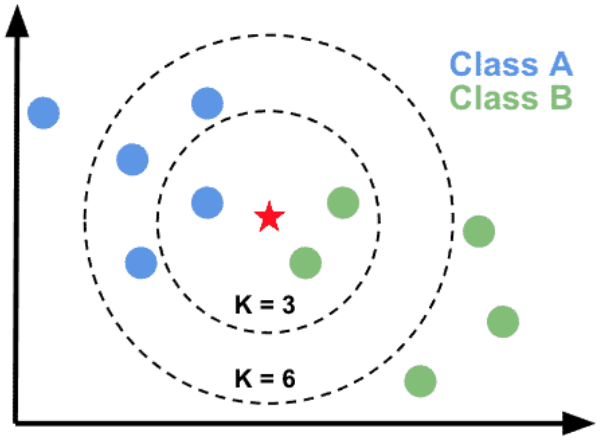

Fig. 4 - Demonstração do algoritmo KNN.

B. Random Forest

Random Forest também é um método de aprendizado de máquina supervisionado o qual é baseado em aprendizado por agrupamento (ensemble) e pode ser utilizado para fins classificatórios e de regressão. O algoritmo cria diversas árvores de decisão para os dados em questão e adota as previsões de cada árvore e seleciona a melhor por meio de votação. Esse método tem como característica que quanto maior o número de árvores, melhor será a precisão do algoritmo. Logo, neste trabalho foram utilizadas duas florestas: uma com 10 árvores e outra com 100, com o objetivo de verificar se a variação no resultado médio é pertinente. Além disso, as árvores de decisão das florestas utilizadas estão configuradas com uma profundidade máxima de 7 níveis, a fim de melhorar o desempenho em questão de tempo, acurácia e evitar o overfitting.

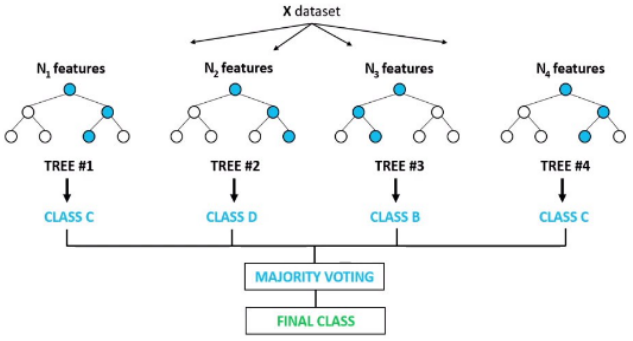

Fig. 5 - Demonstração do algoritmo Random Forest.

C. Árvore de Decisão

Árvore de decisão é um método de aprendizado de máquina supervisionado não-paramétrico que utiliza um conjunto de regras para tomar decisões.

O nó interno representa o atributo da classe analisada e o galho a regra para a tomada de decisão.

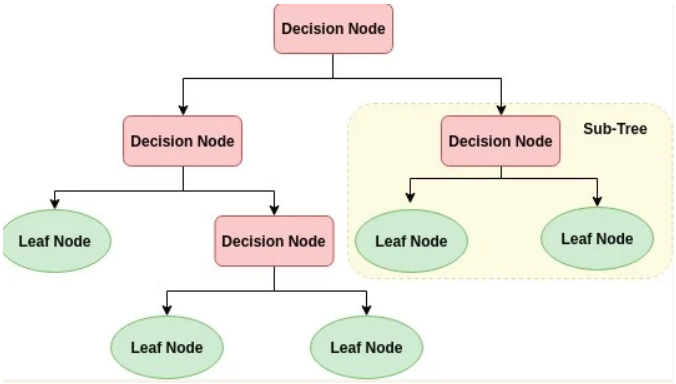

Fig. 6 - Demonstração de uma Árvore de Decisão.

Árvores de decisão são de fácil entendimento e interpretação, além de ser possível visualizá-las no ambiente de desenvolvimento. Também requerem pouca preparação dos dados, apesar dos mesmos terem sido normalizados. Para a realização deste trabalho, foi utilizada apenas uma árvore de decisão, a qual também conta com configuração de profundidade máxima de 7 níveis, a fim de retornar melhores resultados em questões de tempo de execução e acurácia e evitar o overfitting.

IV. Análise Experimental

A. Ambiente de Desenvolvimento e Computacional.

Neste trabalho o ambiente computacional era caracterizado por uma CPU AMD Ryzen 5 2600G, com 6 núcleos e 12 threads. Também conta com dois pentes de memória RAM Corsair Vengeance de 8GB, 2666MHz de frequência e DDR4. O sistema operacional utilizado foi o Windows 10 Pro e o ambiente de desenvolvimento foi a interface do Visual Studio Code.

Dentro desse ambiente foram utilizadas as seguintes bibliotecas:

```
import numpy as np
import pandas as pd
import time
start_time = time.time()
from sklearn import neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
import category_encoders as ce
from sklearn import preprocessing, linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_repo
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig. 7 - Bibliotecas utilizadas.

Todas as bibliotecas foram utilizadas seguindo instruções de documentação das mesmas.

A biblioteca scikit-learn é caracterizada por ser amplamente utilizada no meio de aprendizado de máquina e inteligência artificial, e devido a isso, contém todos os métodos de classificação utilizados neste estudo.

Já o pacote Numpy é caracterizado por possuir operações algébricas e de manipulação de listas. Esta biblioteca auxiliou nas operações matemáticas realizadas durante o desenvolvimento.

O Pandas é um pacote de gerenciamento de arquivos e dados no geral. Foi a partir dele que foi possível a importação dos dados utilizados para a predição e treinamento.

A biblioteca Time foi utilizada para visualização do tempo de execução das funções e de todo algoritmo e a Seaborn e Matplotlib para geração dos gráficos.

B. Análise dos Dados e funcionamento do código

Após o entendimento e formatação dos dados, foi dado início à análise dos mesmos. Uma particularidade da função train_test_split é que além de configurar as variáveis de treino e teste, e predizer o tamanho do conjunto de teste, também é possível passar o parâmetro random_state como "None". Esse valor atribuído a esse parâmetro faz com que a função separe os dados de treino e teste de forma diferente a cada execução, respeitando sempre o tamanho do "test_size". Logo, isso nos ajuda a conseguir resultados diferentes a cada execução das funções, já que os dados serão aleatoriamente separados a cada chamada.

A idéia principal por trás do algoritmo é verificar qual método possui melhor resultado de classificação após usar 66,67% da base como treino em 100 execuções. Para isso, em cada uma das funções foi utilizada a função "predict"de cada técnica, a qual usa os 33,33% dos atributos separados para teste para gerar as predições. A função retorna uma lista com as classes que foram geradas pelo método de classificação. Essa lista é usada como parâmetro da função accuracy_score, juntamente com a lista de atributos usados para teste. Esta função compara essas duas listas e retorna a porcentagem de acerto. É importante citar que esse método foi importado da biblioteca "scikit learn". Para que cada função fosse executada 100 vezes, um laço de repetição foi criado e a cada execução do mesmo, uma chamada de cada método de classificação é feita. Os métodos de classificação retornam um objeto do tipo ClassificationObject,

que possui dois atributos: tempo de execução e porcentagem de acertos. As funções “averageExecutionTime” e “averageResultClassifier”, respectivamente, somam os atributos em listas de números reais, as quais serão responsáveis pelo cálculo da média de tempo e precisão de cada classificador.

C. *Resultados*

Após as 100 execuções de cada técnica, as médias foram computadas e chegou-se nos seguintes resultados de precisão:

- Precisão Média KNN: 85.0333333333333%
- Precisão Média Random Forest com profundidade 10: 93.359649122807%
- Precisão Média Random Forest com profundidade 100: 94.22982456140343%
- Precisão Média Árvore de Decisão: 92.05087719298241%

```
Average KNN:  85.0333333333333
Average Random Forest 10:  93.359649122807
Average Random Forest 100:  94.22982456140343
Average Decision Tree:  92.05087719298241
```

Fig 8 - Média da porcentagem de acerto dos algoritmos.

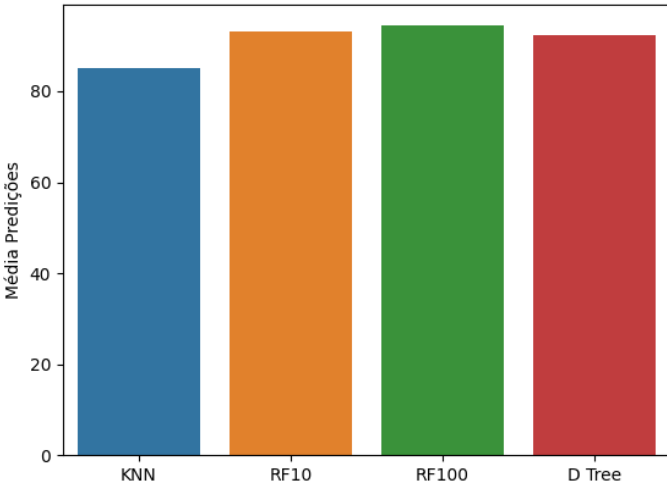

Fig. 9 - Gráfico de barras das médias das porcentagens de acerto.

Também foi computado o tempo de execução médio de cada algoritmo e o tempo total de execução do programa.

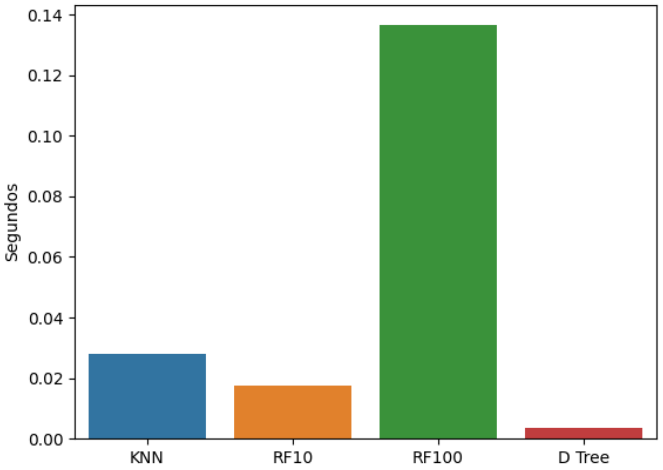

Fig. 10 - Tempo médio da execução de cada técnica.

```
Execution time: 24.06250548362732
```

Fig. 11 - Tempo de execução total do programa.

A técnica de Random Forest com 10 e com 100 árvores, juntamente com o método de árvore de decisão, apresentaram os melhores resultados, com aproximadamente 93%, 94% e 92%, respectivamente. O pior resultado apresentado foi o do método KNN, com aproximadamente 85% de aproveitamento nas predições.

Ao relacionar os tempos de execução dos algoritmos que utilizam árvores de decisão, percebemos que apesar de aumentar significativamente o número de árvores no Random Forest, a predição não foi melhor de forma significativa. Além disso, o custo computacional exercido pelo método com muitas árvores é extremamente elevado quando comparado aos outros métodos.

Apesar desses fatos, o aproveitamento geral dos algoritmos é extremamente satisfatório, dado que é uma base de dados consideravelmente populosa.

V. Conclusão

Baseando-se nas informações adquiridas anteriormente, temos que a custo computacional o Random Forest é muito mais custoso que a técnica de KNN e Árvores de decisão, porém apresenta melhor aproveitamento de predição.

Num escopo geral, o programa se trata de um algoritmo de complexidade de tempo extremamente alta, já que manipula diversas listas e realiza múltiplas iterações entre elas. Além disso, a recorrente verificação das variáveis, construção dos gráficos e processo de treinamento das técnicas

contribui muito para esse aspecto complexo do código.

Durante o estudo para o desenvolvimento desse projeto, foi de extrema importância o conhecimento adquirido durante o curso de Inteligência Artificial, além de reforçar ainda mais todo o aprendizado. O conhecimento prévio de pré processamento de dados e principalmente das técnicas empregadas enriqueceram muito o processo de pesquisa e desenvolvimento do algoritmo.

References

[1] Vovk, V. Gammerman, A.,Shafer, G.: Algorithmic Learning in a Random World.

[2] Devetyarov, D., Nouretdinov, I.: Prediction with Confidence Based on a Random Forest Classifier.

[3] Breiman, L.: Random Forests. Mach. Learning.

[4] OSA, João Luís Garcia. Fundamentos da Inteligência Artificial.

[5] Stuart Russel; Peter Norvig. Artificial Intelligence: A Modern Approach, 3rd edition, Prentice Hall, 2009.

[6] Luger, G. Artificial Intelligence: Structures and Strategies for Complex Problem Solving.

[7] https://archive.ics.uci.edu/ml/datasets/Car+Evaluation