Reconhecimento de pessoas autistas utilizando aprendizado de máquina supervisionado

Rafael Tenfen[1], **Douglas Felipe Bussmann Schiedlag**[1]

[1]Departamento de Engenharia de Software – Universidade Estadual de Santa Catarina (UDESC)
89.140-000 – Ibirama – SC – Brazil

Abstract. *This article has a objective, describe the use of two supervised machine learning classifiers to distinguish when someone has autism, following the Data-Set available in Autism Screening Adult Data Set (UCI). One of the techniques is using decision tree and the other one is using SVM - Support Vector Machine.*

Resumo. *Este artigo tem por objetivo, descrever o uso de duas técnicas de classificação de aprendizado de máquina supervisionado para distinguir quando uma pessoa adulta é autista, seguindo os dados dispostos no dataset Autism Screening Adult Data Set (UCI). Uma das técnicas utilizadas é a árvore de decisão, e a outra é suporte a vetores de máquina.*

1. Introdução

Aprendizado de máquina têm sido cada vez mais utilizados como uma ferramenta para a predição ou classificação de dados e auxiliar na tomada de decisão em inúmeros problemas. Por exemplo a distinção se uma paciente possui ou não uma doença específica, uma análise de dados com várias amostras rotuladas gravadas em um banco de dados, podem auxiliar trazendo a propabilidade de acerto da máquina e se o paciente possui a doença, e assim auxiliando a decisão do médico.

Existem várias formas de aprendizados de máquina, aprendizado por reforço, supervisionado e não supervisionado. Porém, o aprendizado a ser tratado neste trabalho é o aprendizado supervisionado, que é chamado de supervisionado pois os dados rotulados servem como exemplos, que são utilizados durante o processo de aprendizagem.Este paradigma exige um esforço humano em rotular um conjunto de dados para a aprendizagem, mas produzem hipóteses que podem ser utilizadas em tarefas de predição futuras.

Este artigo apresenta uma proposta de previsão de classificação de pessoas adultas se elas são autistas, tendo como cenário 704 amostras rotuladas de pessoas com 10 questões, idade, etínia, gênero, icterícia, país de origem, a relação de quem preencheu o formulário e o resultado de se essa pessoa era autista. Com base nas amostras, foram desenvolvidos dois classificadores utilizando árvores de decisão e suporte a vetores de máquina.

2. Detalhes de Implementação

Para realizar a implementação dos classificadores foi utilizando uma linguagem de programação que permite trabalhar de forma rápida e eficaz, Python. O módulo utilizado foi o Scikit-learn que integra uma ampla gama de algoritmos de aprendizado de máquina

de última geração para problemas supervisionados e não supervisionados de média escala. Este pacote se concentra em trazer o aprendizado de máquina para não especialistas usando uma linguagem de alto nível de propósito geral [Pedregosa et al. 2011a].

As importações que foram utilizadas para a realização desse trabalho seguem na Figura 1 abaixo.

Importações utilizadas:

- arff -Leitura do DataSet
- panda -Criação do DataFrame
- numpy -Operações com números
- tree -Criação da árvore de Decisões
- svm -Criação da máquina de suporte a vetores
- train_test_split -Separação de amostras de teste e treinamento
- accuracy_score -Obtenção da acurácia
- time -Tempo para comparações
- display -Gráficos
- pl -Plotar os dados
- mpatche -Auxiliar para medidas

In [1]:
```
from scipy.io import arff
import pandas as pd
import numpy as np
import sklearn.tree as tree
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from time import time
from IPython.display import display
import matplotlib.pyplot as pl
import matplotlib.patches as mpatches
```

Figure 1. Importações Utilizadas

Como algumas amostras possuiam dados faltantes ou era necessário criar um decionário de dados muito extenso para a conversão de strings em números, foram deletado 6 colunas [’ageDesc’, ’Class/ASD’, ’result’, ’contryOfRes’, ’ethnicity’, ’relation’] demonstrados na Figura 2 e também todas as amostras com dados incompletos, ao fim totalizando 609 amostras completas e formatadas para o treino e teste dos classificadores.

In [3]:
```
replace_map = { 'gender': {
                    'f' : 1,
                    'm' : 0
                },
                'jundice': {
                    'yes': 1,
                    'no': 0
                },
                'austim': {
                    'yes': 1,
                    'no': 0
                },
                'used_app_before': {
                    'yes': 1,
                    'no': 0
                },
                'Class/ASD' : {
                    'YES': 1,
                    'NO': 0
                }
              }

df.replace(replace_map, inplace=True)

df = df[df != '?']

df.dropna(axis = 0, inplace = True)

y = df['Class/ASD']

df.drop(['age_desc', 'Class/ASD', 'result', 'contry_of_res', 'ethnicity', 'relation'], axis=1, inplace=True)

df = df.astype(np.float)
```

Figure 2. Implemetação da formatação dos dados

2.1. Árvores de decisão

Árvores de decisão são um método de aprendizagem supervisionada não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que prediz o valor de uma variável de destino, aprendendo regras de decisão simples inferidas a partir de amostras de dados[Pedregosa et al. 2011b].

A implementação realizada para árvore de decisão e visualização da matriz de confusão podem ser observadas na Figura 3, para esse classificador foram utilizado vinte porcento dos dados para teste que equivale a 122 amostras, e oitenta porcento dos dados para o treinamento do modelo o equivalente a 487 amostras.

Decision Tree

- Matriz de Confusão
- Tempo de Treinamento
- Acurácia total

```
In [5]: clfDecisionTree = tree.DecisionTreeClassifier(random_state=50)

start = time()
clfDecisionTree.fit(X_train, y_train)
end = time()

prediction = clfDecisionTree.predict(X_test)

print (pd.crosstab(y_test, prediction, rownames=['Autista'], colnames=['Predição'], margins=True))

print ("Tempo Total de Treinamento:", end - start, "segundos")

print ("Acurácia total: ", accuracy_score(y_test, prediction))
```

```
Predição   0   1  All
Autista
0         68   7   75
1          9  38   47
All       77  45  122
Tempo Total de Treinamento: 0.0039722919464411133 segundos
Acurácia total:  0.8688524590163934
```

Figure 3. Implementação da árvore de decisão e matriz de confusão

Uma representação da árvore de decisão criada com o código referente a Figura 3 pode ser visualizado abaixo na Figura 4.

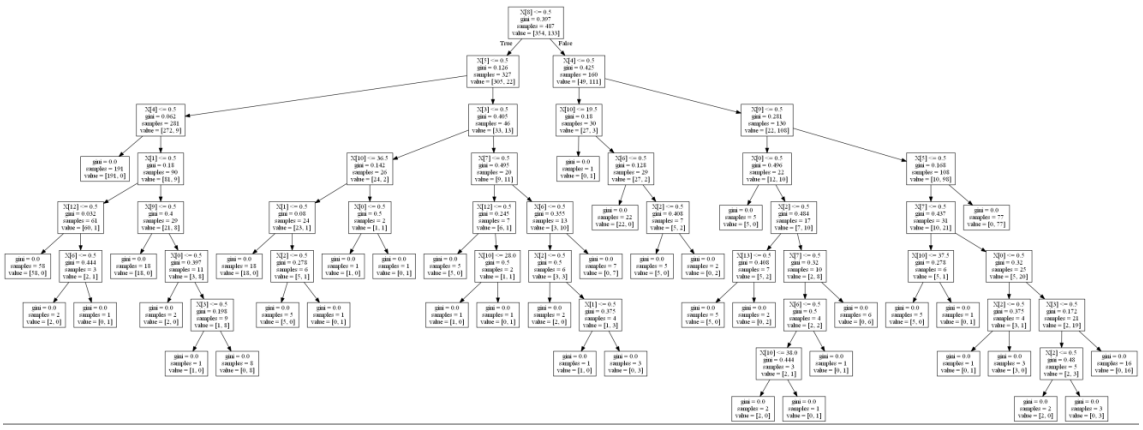

Figure 4. Visualização da árvore de decisão

2.2. Suporte a Vetores de Máquina

Support Vector Machine (SVM) são modelos de aprendizado de máquina suporveisionados associados a algoritmos de análise de dados, que podem ser utilizados tanto para

classificação quanto para regressão. Dado um conjunto de amostras, o classificador cria um modelo de atribuição de categorias a cada amostra, e assim ao receber uma nova amostra, visualiza com qual categoria melhor se encaixa, tornando um classificador linear binário não probabilístico, como o escalonamento Platt [Wikipedia contributors 2018]. Enfim, um modelo SVM é uma representação das amostras como pontos no espaço, categorizados para que seja possível dividir pelo melhor plano possível e categorizar uma nova amostra [Pedregosa et al. 2011b].

A implementação realizada para suporte a vetores de máquina e visualização da matriz de confusão podem ser observadas na Figura 5, para esse classificador foram utilizado vinte porcento dos dados para teste que equivale a 122 amostras, e oitenta porcento dos dados para o treinamento do modelo o equivalente a 487 amostras.

Figure 5. Implementação SVM e matriz de confusão

3. Análise e Resultados

Para análisar e demonstrar os resultados foram criados duas funções ‘train_predict‘ para treinar os classificadores e calcular as métricas de análise, que são ‘acc_test‘ - Acurácia do Teste, ‘acc_train‘ - Acurácia do Treino, ‘f_test‘ - Média Harmônica Ponderada de Precisão e Recordação do Teste, ‘f_train‘ - Média Harmônica Ponderada de Precisão e Recordação do Treino, ‘pred_time‘ - Tempo em Segundos da Predição, ‘train_time‘ - Tempo em Segundos do Treino do Modelo. E também a função ‘evaluate‘ que é utilizada para demonstrar de forma gráfica as análises realizadas.

Para a comparação dos classificadores foram utilizados três medidas de treinos: 1, 10 e 100 % dos dados como treinamento dos modelos. Conforme representação da implementação demonstrada na Figura 6.

Comparações Entre os Classificadores

- Treinamentos com 1, 10 e 100 % dos dois classificadores
- Representação dos resultados com tabelas
- Representação dos resultados em gráficos

In [30]:
```
samples_100 = len(y_train)
samples_10 = int(len(y_train)/10)
samples_1 = int(len(y_train)/100)

clf_A = tree.DecisionTreeClassifier(random_state=50)
clf_B = svm.SVC(random_state=50, gamma='auto')

# Collect results on the learners
results = {}
for clf in [clf_A, clf_B]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    for i, samples in enumerate([samples_1, samples_10, samples_100]):
        results[clf_name][i] = \
        train_predict(clf, samples, X_train, y_train, X_test, y_test)

for i in results.items():
  print(i[0])
  display(pd.DataFrame(i[1]).rename(columns={0:'1% of train', 1:'10% of train', 2:'100% of train'}))

evaluate(results)
```

Figure 6. Implementação da Comparação dos Classificadores

Análise representada em tabelas podem ser mais fáceis de visualizar a diferença entre as porcentagens de treinamento, conforme disponível na Figura 7.

```
DecisionTreeClassifier trained on 4 samples.
DecisionTreeClassifier trained on 48 samples.
DecisionTreeClassifier trained on 487 samples.
SVC trained on 4 samples.
SVC trained on 48 samples.
SVC trained on 487 samples.
DecisionTreeClassifier
```

	1% of train	10% of train	100% of train
acc_test	0.663934	0.803279	0.868852
acc_train	0.716632	0.854209	1.000000
f_test	0.583090	0.748899	0.837004
f_train	0.538462	0.716096	1.000000
pred_time	0.002000	0.001002	0.001001
train_time	0.001996	0.001998	0.001999

SVC

	1% of train	10% of train	100% of train
acc_test	0.631148	0.598361	0.909836
acc_train	0.741273	0.733060	0.987680
f_test	0.238095	0.079365	0.917874
f_train	0.298507	0.263158	0.981735
pred_time	0.001983	0.001999	0.005012
train_time	0.001999	0.002997	0.006982

Figure 7. Tabela de Demonstração de Análise dos Classificadores

A comparação entre os dois classificadores já é mais simples de ser analisada com gráficos intuitivos conforme disponível na Figura 8.

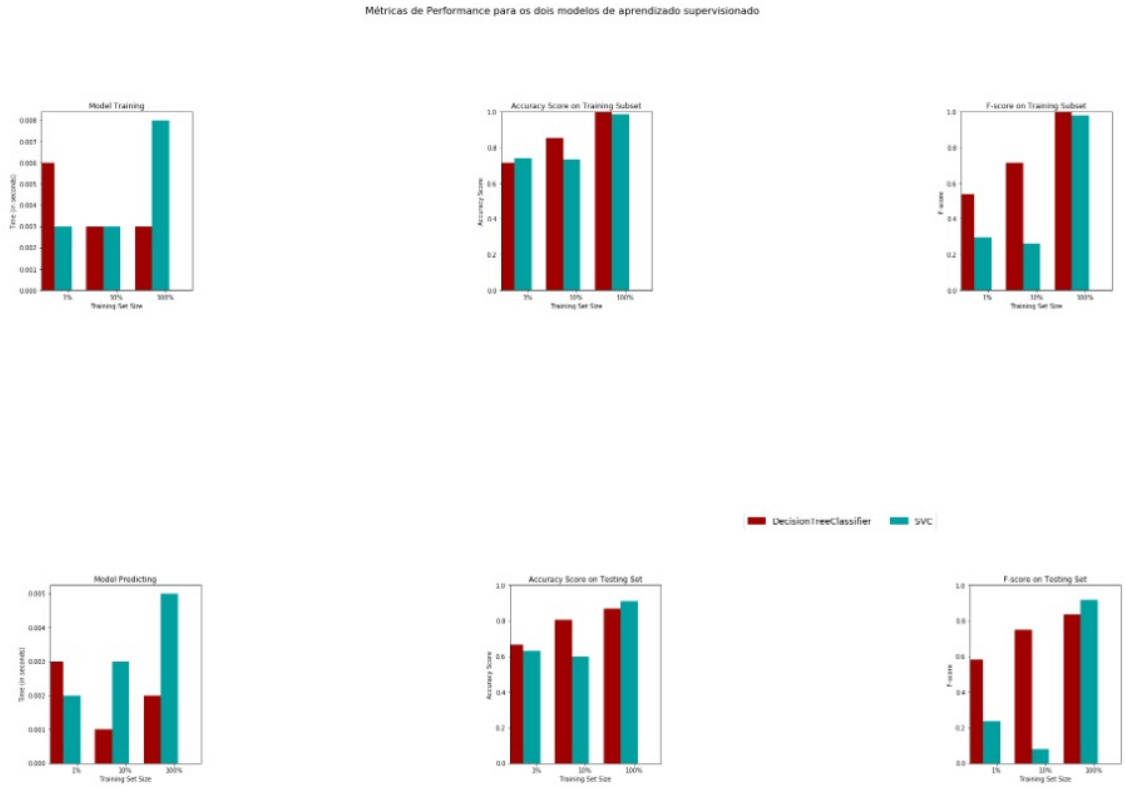

Figure 8. Gráfico de Comparação dos Classificadores

4. Conclusão

Enfim com base em o que foi dito, é possível identificar se uma pessoa adulta é autista utilizando as técnicas de aprendizado de máquina supervisionado com 90% de probabilidade de acerto com o classificador de suporte a vetores de máquina e com 86% utilizando o classificador de árvores de decisão.

References

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011a). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011b). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Wikipedia contributors (2018). Support vector machine — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=866892968`. [Online; accessed 11-November-2018].