Classificação usando Floresta Aleatória

David Walter Jansen
Iarah Almeida
Paulo Augusto Borges

Junho de 2019

História

- Tim Kam Ho: Artigo “Floresta aleatória de decisão”, 1995.
  - Criação do método.
- Tim Kam Ho: Artigo “O método de subespaços aleatórios para construção de florestas de decisão”, 1998.
  - Anteriormente eram divisões em hiperplanos oblíquos.
- Leo Breiman: Artigo “Floresta aleatória”, 2001.
  - Importância relativa das características.

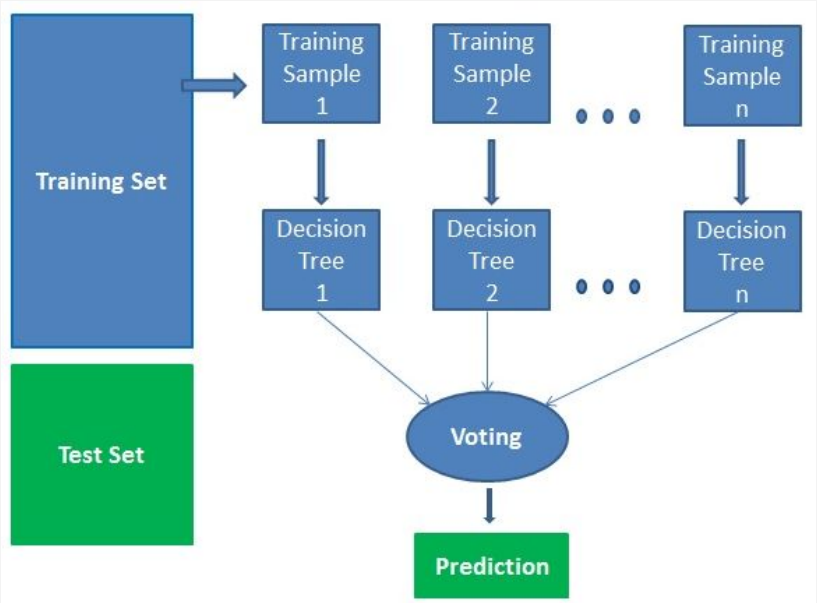
Noção Intuitiva do Algoritmo (Divisão e Conquista)

Analogia com escolha de local para viagem.

(1) Perguntar aos amigos qual local sugerem.
(2) Agrupar todas as recomendações dos seus amigos em uma lista.
(3) Realizar uma votação sobre a lista gerada no passo 2.
(4) Escolher o local para viagem que teve maior votação.

Ideia do Algoritmo

1. Selecione características aleatórias de um determinado conjunto de dados.
2. Construa uma árvore de decisão para cada conjunto de características selecionados no passo anterior e obtenha um resultado de previsão de cada árvore de decisão.
3. Faça um voto para cada resultado previsto.
4. Selecione o resultado da previsão com o maior número de votos como previsão final.



Vantagens

- O uso de muitas árvores de decisão torna a Random Forest uma método preciso e robusto.

- Como podemos fazer uso de várias árvores de decisão, esse método não sofre com overfitting.

- Como dissemos antes, podemos usar a Random Forest para classificação e regressão.

- É possível obter a importância relativa da característica, o que ajuda na seleção das melhores para treinar e testar o modelo.

Desvantagens

- Por usar as n-árvores de decisão sempre que se deseja fazer uma previsão e depois realizar a votação, a execução completa do algoritmo pode se tornar lenta.

- Esse modelo requer uma análise mais cuidadosa na interpretação em comparação com uma única árvore de decisão, onde apenas precisamos olhar para o caminho dessa árvore.

- A ordenação das características mais importantes pode alterar até para bases pequenas como a **Iris**. Isso ocorre devido à natureza aleatória do algoritmo na construção das árvores de decisão.

Coeficiente de Gini

- É uma das métricas utilizadas para decidir qual dos *m* atributos de certo conjunto de dados deve ser escolhido para ser a raiz ou nó intermediário da árvore de decisão.

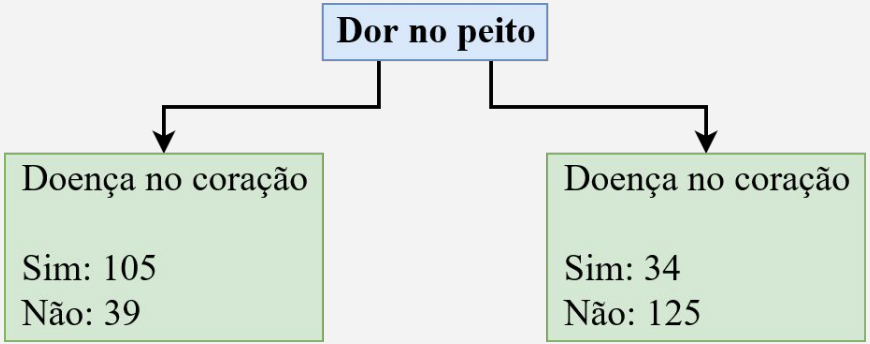
- Diz qual a frequência que um elemento escolhido aleatoriamente é identificado incorretamente.

- É dado por um número entre 0 e 1, que diz a medida de desigualdade em que 0 significa completa igualdade e 1 completa desigualdade, portanto, um atributo com menor coeficiente deve ser preferido.

Coeficiente de Gini

$$
\begin{aligned}
G_{esq} &= 1 - (\text{probabilidade de sim})^2 - (\text{probabilidade de no})^2 \\
&= 1 - (\frac{105}{105+39})^2 - (\frac{39}{105+39})^2 \\
&= 0.395
\end{aligned}
$$

$$
\begin{aligned}
G_{dir} &= 1 - (\text{probabilidade de sim})^2 - (\text{probabilidade de no})^2 \\
&= 1 - (\frac{34}{34+125})^2 - (\frac{125}{34+125})^2 \\
&= 0.336
\end{aligned}
$$



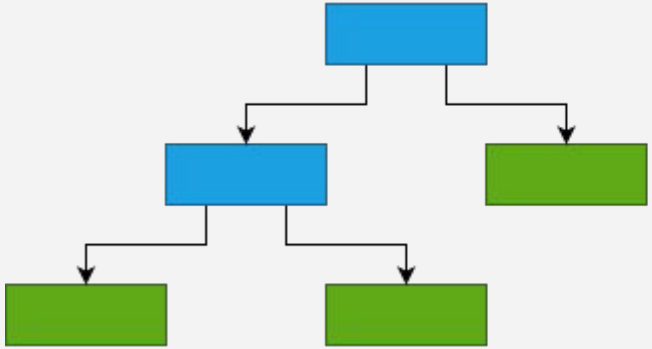
x = número de pacientes da folha esquerda = 144.

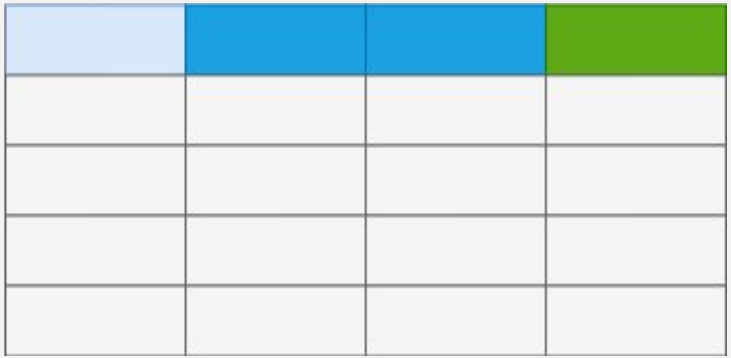
y = número de pacientes da folha direita = 159.

$$
\begin{aligned}
G_{raiz} &= \frac{x}{x+y} \cdot G_{esq} + \frac{y}{x+y} \cdot G_{dir} \\
&= \frac{144}{144+159} \cdot 0.395 + \frac{159}{144+159} \cdot 0.336 \\
&= 0.364
\end{aligned}
$$

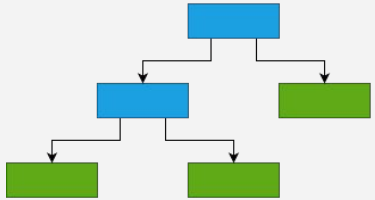
Pseudocódigo - Criação

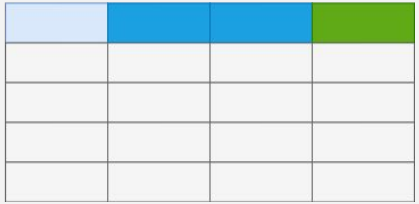
1. selecione, de forma aleatória, **k** características dentre as **m** características totais (**k < m**)
2. dentre as **k** características, calcule o nó **d** utilizando o coeficiente de Gini
3. divida o nó em nós filhos utilizando o coeficiente de Gini
4. repita os passos 1 ao 3 até que **l** nós tenham sido alcançados
5. construa a floresta repetindo os passos 1 ao 4 por **n** vezes e assim crie **n** árvores



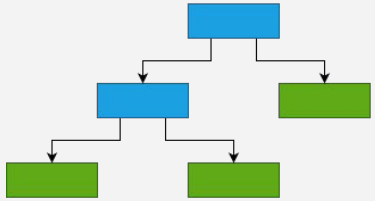


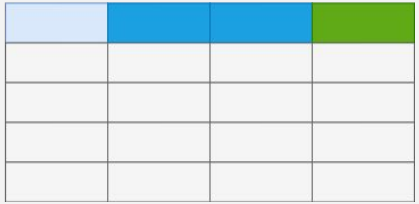
Pseudocódigo - Criação

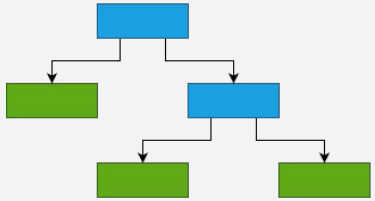


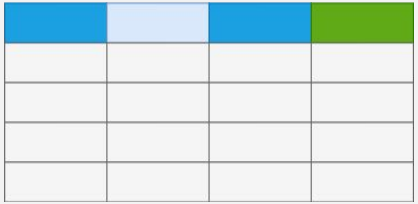


Pseudocódigo - Criação

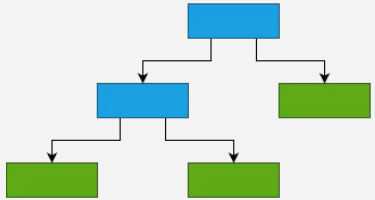


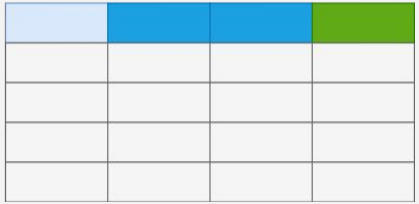


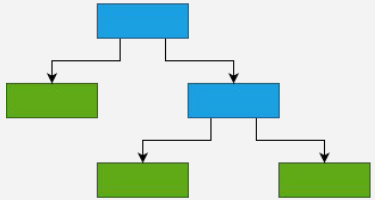


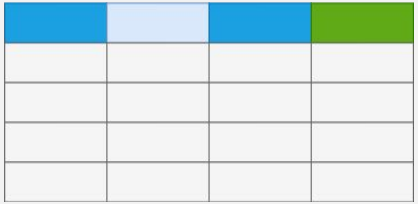


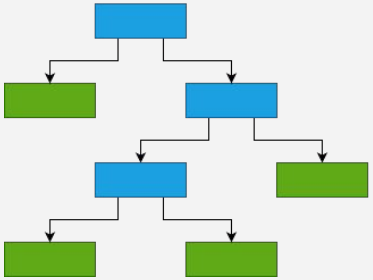
Pseudocódigo - Criação

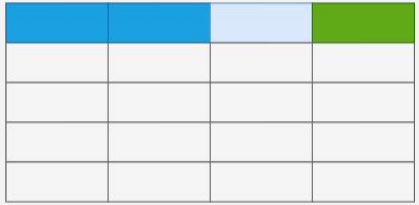








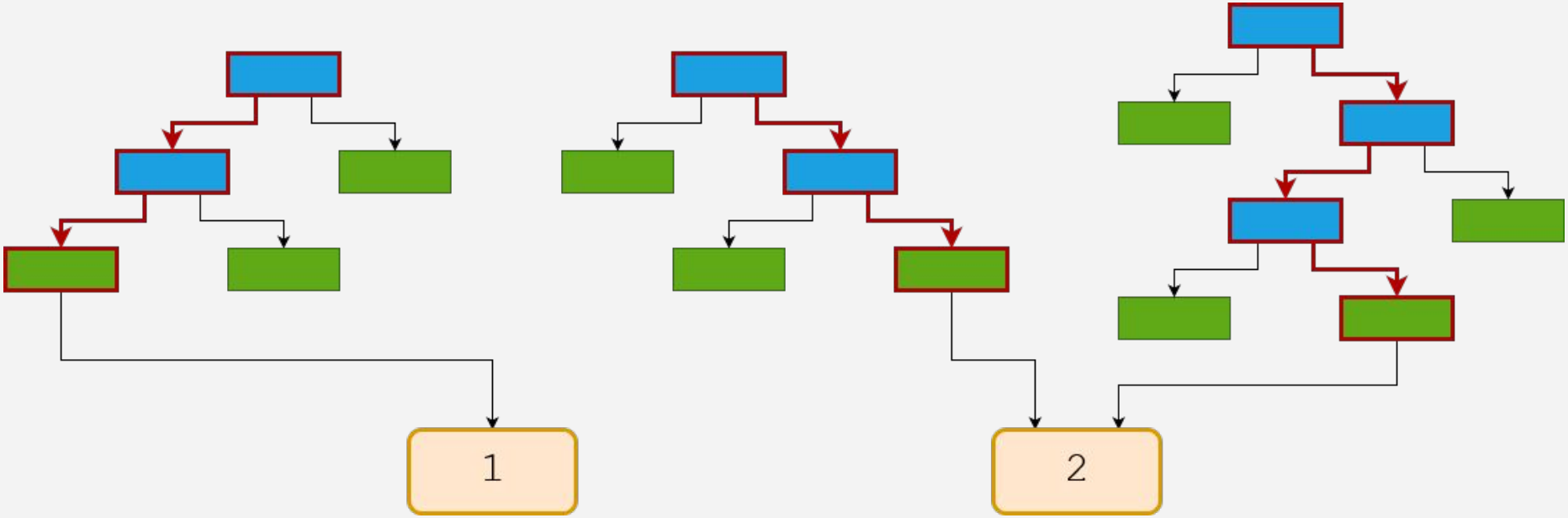




Pseudocódigo - Predição

1. dada as características da instância de teste, as utiliza em cada árvore de decisão criada na etapa anterior para predizer o resultado, guardando o resultado predito (classificação)
2. calcula os votos de cada classificação predita
3. considera a classificação mais votada como a predição final do algoritmo de floresta aleatória

Pseudocódigo - Predição



Implementação

Fornecida pela Scikit Learn:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

class sklearn.ensemble.**RandomForestClassifier**(*n_estimators='warn'*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=None*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*)

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

- Número de Atributos: 34 (ID, **outcome**, 32 valores reais correspondentes às características).
- Outcome (R = recorrência, N = não-recorrência)
- Número de instâncias: 198
- Usar o ***Random Forest Classifier*** para classificar o atributo **outcome**.

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

Divisão da base entre características e target.

```
#6.1)

#Separa as colunas em dependentes e independentes

namesrfc = ['radius_mean',
        'texture_mean', 'perimeter_mean', 'area_mean',
        'smoothness_mean', 'compactness_mean', 'concavity_mean',
        'concave_points_mean', 'symmetry_mean',
        'fractal_dimension_mean', 'radius_se', 'texture_se',
        'perimeter_se', 'area_se', 'smoothness_se',
        'compactness_se', 'concavity_se', 'concave_points_se',
        'symmetry_se', 'fractal_dimension_se',
        'radius_worst', 'texture_worst', 'perimeter_worst',
        'area_worst', 'smoothness_worst',
        'compactness_worst', 'concavity_worst',
        'concave_points_worst', 'symmetry_worst',
        'fractal_dimension_worst', 'tumor_size', 'lymph_node_status']

X = wpbc[namesrfc]  # Características
y = wpbc['outcome']  # Rótulo


# Treinamento do modelo
# Importa 'RandomForestClassifier' de 'sklearn.ensemble'
from sklearn.ensemble import RandomForestClassifier
# Importa 'metrics' de 'sklearn'
from sklearn import metrics
```

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

Importações.

```
# Cria um classificador gaussiano
clf = RandomForestClassifier(n_estimators=100)

# Importa o K-fold para separar entre treino e teste
from sklearn.model_selection import KFold

# Valor do K (número de divisões na base de dados)
kf = KFold(n_splits = 10)

best_precision = 0.0000
sum_precisions = 0.0000
```

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

Treinamento e Teste.

```
# 6.1.1)
i = 0
for train_index, test_index in kf.split(X):
    i = i + 1 #contado de split
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    # Treina o modelo usando o conjunto de treinamento
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Modelo de precisão, que diz qução frequente o classificador está correto
    precision = metrics.accuracy_score(y_test, y_pred)
    print("Precisão encontrada para %dº split: " %(i) ,end="")
    print(precision)
    sum_precisions = sum_precisions + precision
    if best_precision < precision:
        best_precision = precision
        clf_best = clf
        y_test_best = y_test
        y_pred_best = y_pred
```

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

Exibição dos Resultados.

```
print("Resultados Atuais..")
print("\nPrecisão média para as %d predições: " %i, end="")
print(sum_precisions/i);
print("Melhor resultado: ", end="")
print(best_precision)
```

```
Precisão encontrada para 1º split: 0.7
Precisão encontrada para 2º split: 0.8
Precisão encontrada para 3º split: 0.65
Precisão encontrada para 4º split: 0.75
Precisão encontrada para 5º split: 0.9473684210526315
Precisão encontrada para 6º split: 0.5789473684210527
Precisão encontrada para 7º split: 0.631578947368421
Precisão encontrada para 8º split: 0.631578947368421
Precisão encontrada para 9º split: 0.7894736842105263
Precisão encontrada para 10º split: 1.0
Resultados Atuais..

Precisão média para as 10 predições: 0.7478947368421053
Melhor resultado: 1.0
```

Exemplo: Breast Cancer Wisconsin (Prognostic) Dataset

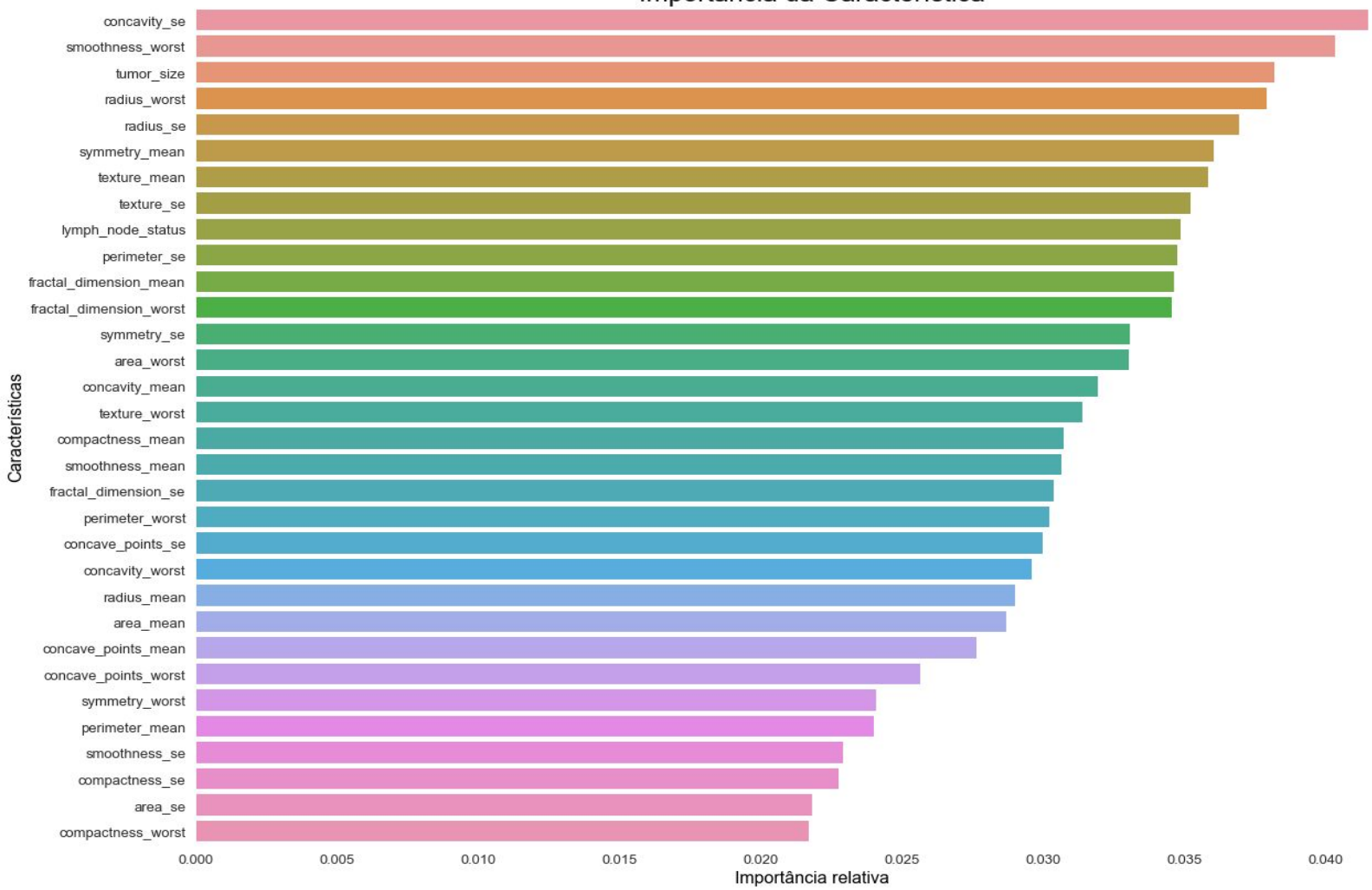
Importância das características.

```
# 6.1.3

# Ordenação das as características importantes pela pontuação usando a iterção com a melhor precisão
print("\nCaracterísticas ordenadas pela pontuação:")
# O número dentro de round é o limite de casas decimais
result = sorted(zip(map(lambda x: round(x, 6), clf_best.feature_importances_), namesrfc), reverse=True)

# Criando dataframe com o resultado
df = pd.DataFrame(data=result)
df = df.rename(index=str, columns={0: "importance", 1: "feature"})

import matplotlib.pyplot as plt
#Configurações para exibição do gráfico
sns.set_color_codes("dark")
sns.set_style("white")
sns.set_context("talk")
plt.figure(figsize=(20,15))
g = sns.barplot(x="importance", y="feature", data=df)
g.axes.set_title('Importância da Característica', fontsize=24,color="black",alpha=2)
g.set_xlabel("Importância relativa", size = 16,color="black")
g.set_ylabel("Características", size = 16,color="black")
sns.despine(left=True, bottom=True)
```


Importância da Característica
concavity_se
smoothness_worst
tumor_size
radius_worst
radius_se
symmetry_mean
texture_mean
texture_se
lymph_node_status
perimeter_se
fractal_dimension_mean
fractal_dimension_worst
symmetry_se
area_worst
concavity_mean
texture_worst
compactness_mean
smoothness_mean
fractal_dimension_se
perimeter_worst
concave_points_se
concavity_worst
radius_mean
area_mean
concave_points_mean
concave_points_worst
symmetry_worst
perimeter_mean
smoothness_se
compactness_se
area_se
compactness_worst
Características
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
Importância relativa

Referências

- Understanding random forests classifiers in python.https://www.datacamp.com/community/tutorials/random-forests-classifier-python#algorithm.Accessado: 2019-05-25.
- Leo Breiman. Random forests.Machine Learning, 45(1):5–32, Oct 2001.
- Tin Kam Ho. Random decision forests. InProceedings of the Third International Conferenceon Document Analysis and Recognition (Volume 1) - Volume 1, ICDAR '95, pages 278–,Washington, DC, USA, 1995. IEEE Computer Society.
- B. M. Paulo Augusto J. David Walter, A. Iarah Gon¸calvez.Random Forest para predição sobre a Breast Cancer Wisconsin (Prognostic) Data Set.2018.
- Tin Kam Ho. The random subspace method for constructing decision forests.IEEE Tran-sactions on Pattern Analysis and Machine Intelligence, 20(8):832–844, Aug 1998.