Introdução a Inteligência Artificial
Alexandre Souza Costa Oliveira
170098168
Universidade de Brasília - UnB

Sobre

O objetivo deste trabalho é utilizar a floresta aleatória para diagnosticar pacientes acerca da doença diabetes. Para isso foi utilizado um banco de dados com 768 pessoas com pelo menos 21 anos, com 8 variáveis independentes e uma variável alvo dizendo se ela possui ou não diabetes. O algoritmo separa 80% dos dados para treinamento e 20% para testes. Neste relatório serão apresentados os resultados obtidos com a implementação desta floresta aleatória para 10 estimadores, 100 estimadores e sqrt(768) estimadores, realizando uma comparação com os resultados apresentados no artigo Sisodia, D. & Sisodia, D. S. “Prediction of Diabetes using Classification Algorithms”, 10.1016/j.procs.2018.05.122.

1. Seleção dos dados

Para a leitura dos dados, no qual está em um arquivo **diabetes.csv**, foi utilizado a biblioteca **pandas**, pois a mesma auxilia na leitura dos dados separando as colunas corretamente. E em seguida é separado em duas variáveis X e y, em que na X ficam apenas as features de cada linha e na y ficam apenas as classes de cada linha.

Após a leitura dos dados, foi utilizado a função ***train_test_split*** da biblioteca ***sklearn.model_selection*** para dividir os dados em 20% para testes e 80% para treino. O modo como isso foi feito, pode ser visto abaixo:

```
from sklearn.model_selection import train_test_split
```

```
# Realiza a divisão dos dados com 20% para teste e 80% para treino
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20, train_size=0.80)
```

2. Resultados da Floresta Aleatória

Para criar a Floresta Aleatória, foi utilizado a classe **RandomForestClassifier**, da biblioteca **sklearn.ensemble**. No qual é possível criar uma floresta e definir a quantidade de árvores que ela terá, no caso, nossos estimadores. E usando o método **fit**, as árvores são criadas para a floresta.

```
from sklearn.ensemble import RandomForestClassifier
```

```
# Instância uma floresta aleatória de classificação
floresta = RandomForestClassifier(n_estimators=estimador)

# Treina a floresta / cria as árvores
floresta.fit(X_train, y_train)
```

Após termos as árvores criadas, é chamado o método **predict** para gerar a predição para os dados de testes. E usando as funções **accuracy_score**, **precision_score, recall_score, f1_score e confusion_matrix**, podemos calcular, respectivamente, a acurácia, a precisão, revocação, a medida f1 e a matriz de confusão de nossos resultados preditos. Essas funções estão contidas na biblioteca **sklearn.metrics**.

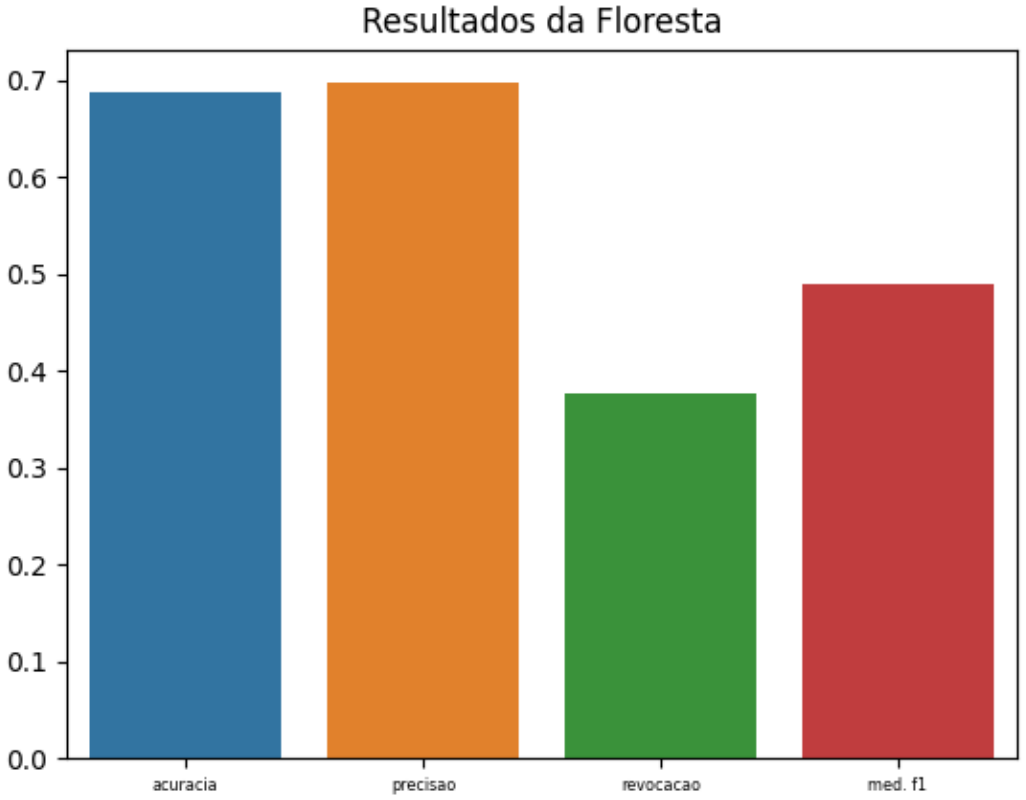
```
# Gera a predição para os dados de testes
y_pred = floresta.predict(X_test)

# Calcula os resultados de acurácia, precisão, revocação, medida f1 e
matriz de confusão
acuracia = accuracy_score(y_test, y_pred)
precisao = precision_score(y_test, y_pred)
revocacao = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
confusao = confusion_matrix(y_test, y_pred)
```

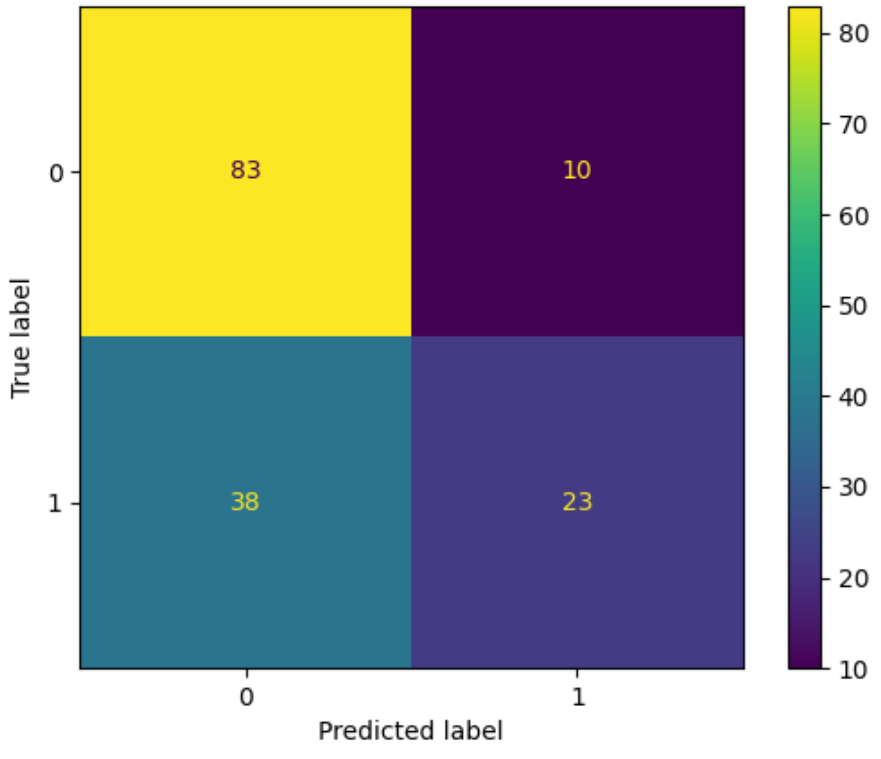
Com a utilização das bibliotecas **matplotlib.pyplot e seaborn**, são plotados três gráficos. O primeiro gráfico com os resultados de acurácia, precisão, revocação e medida f1. O segundo gráfico plotado é a matriz de confusão. E o terceiro gráfico demonstra a importância das 8 variáveis independentes.

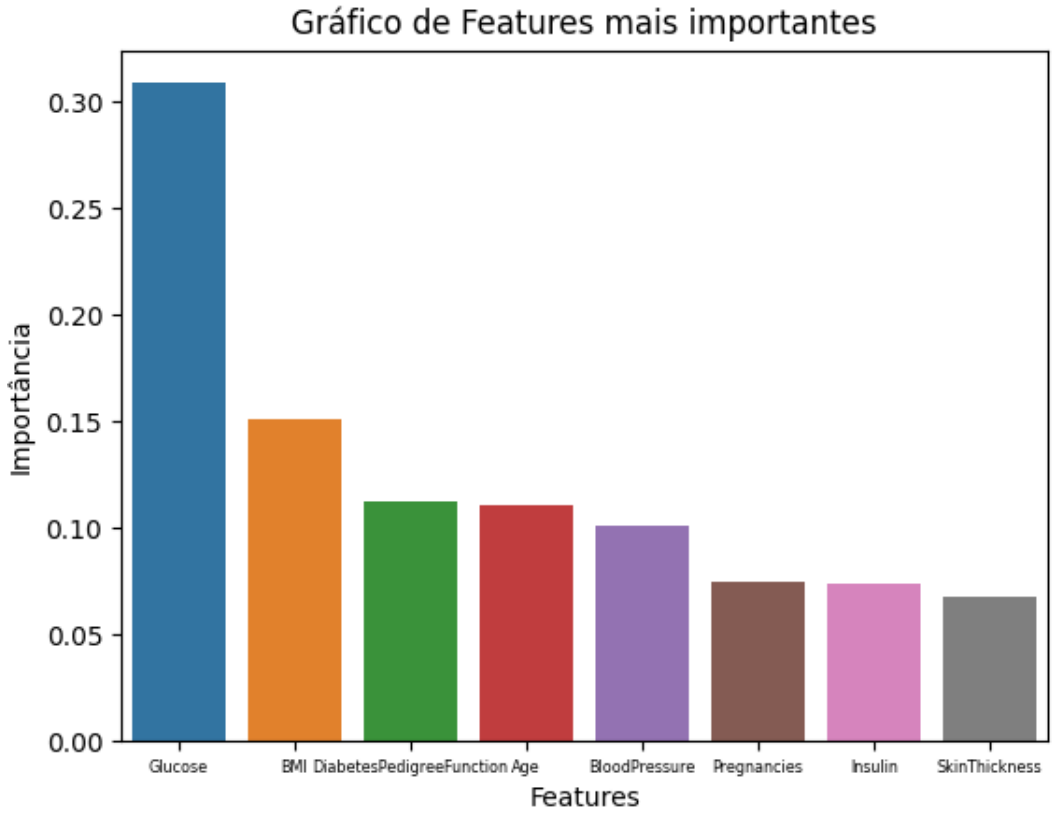
2.1 Resultados para 10 estimadores

Para 10 estimadores na floresta aleatória, ou seja, criação de 10 árvores de decisão, os seguintes resultados plotados em gráficos foram obtidos:



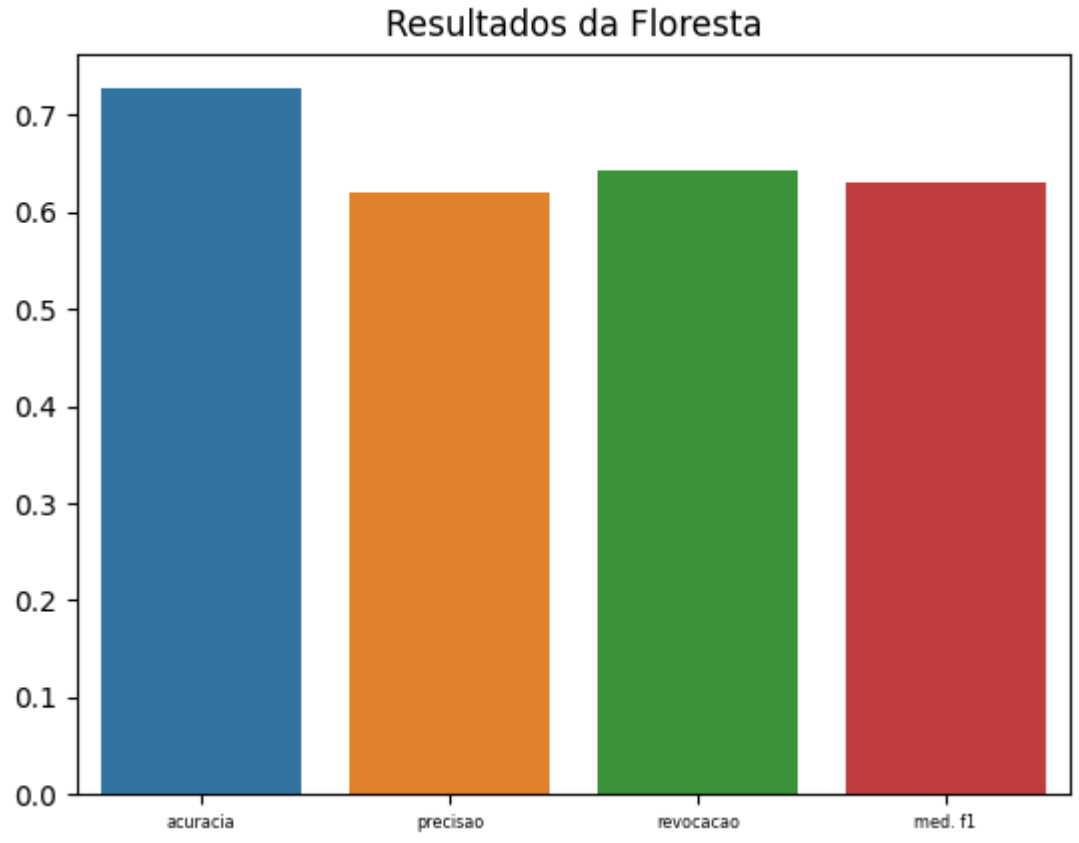
Precisão	Revocação	Medida F1	Acurácia (%)
0.69	0.37	0.48	68.8



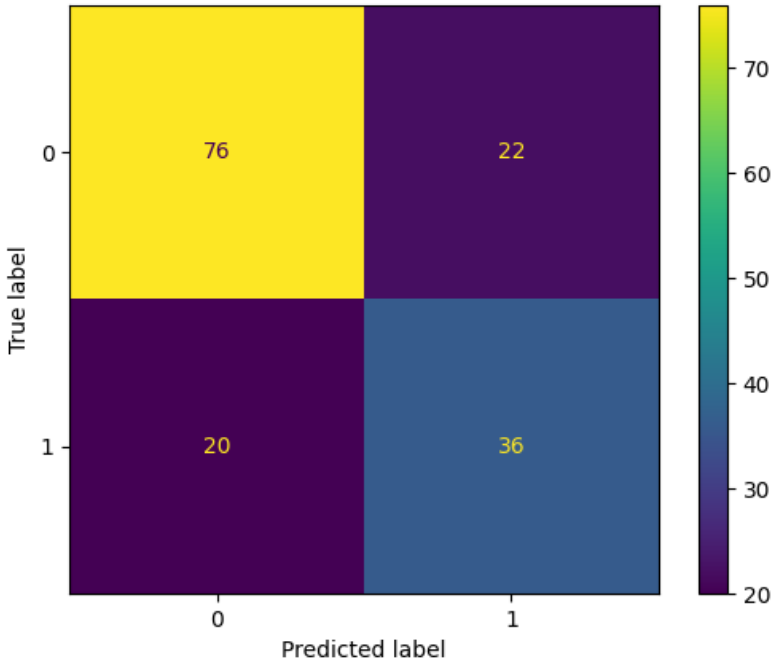


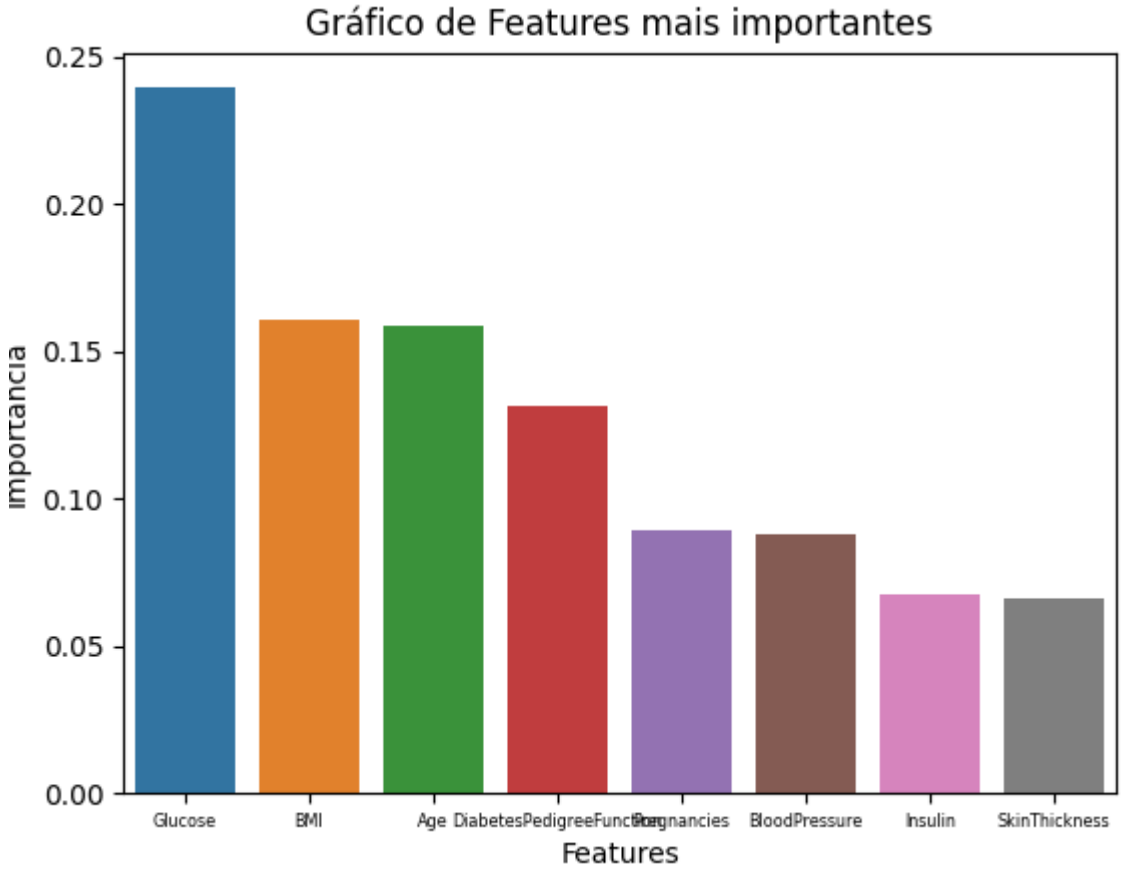
2.2 Resultados para 100 estimadores

Para 100 estimadores na floresta aleatória, ou seja, criação de 100 árvores de decisão, os seguintes resultados plotados em gráficos foram obtidos:

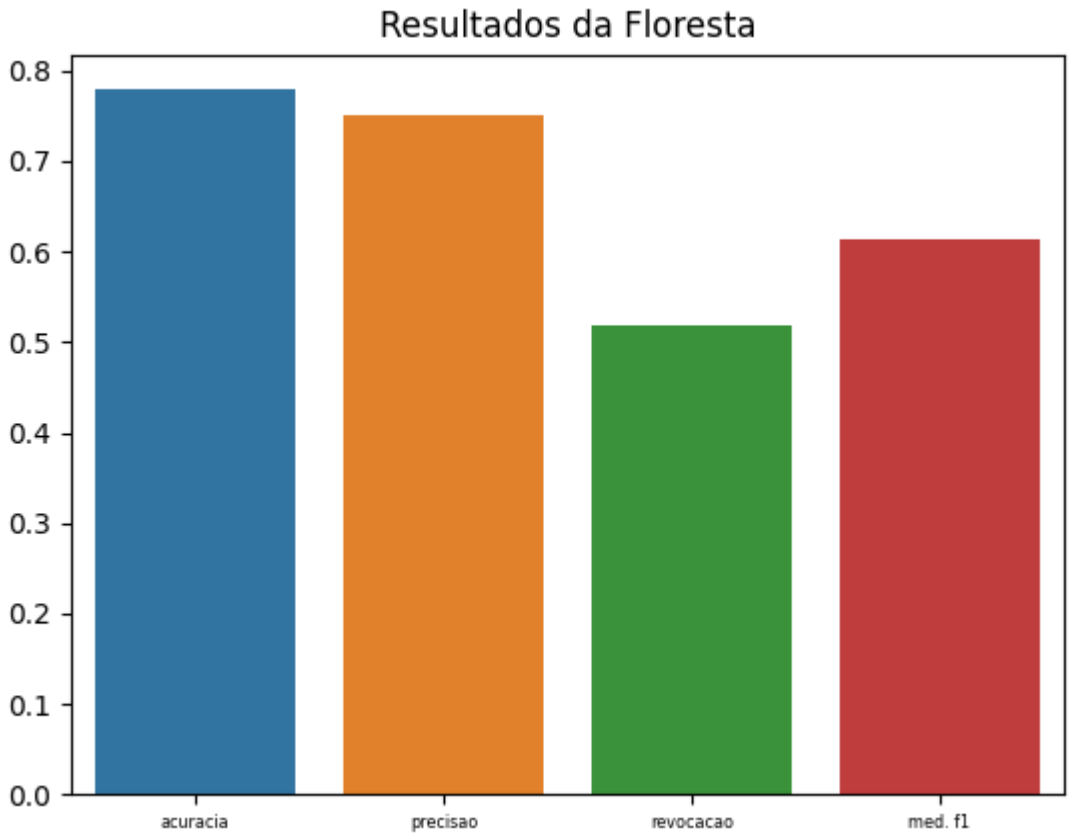


Precisão	Revocação	Medida F1	Acurácia (%)
0.62	0.64	0.63	72.7

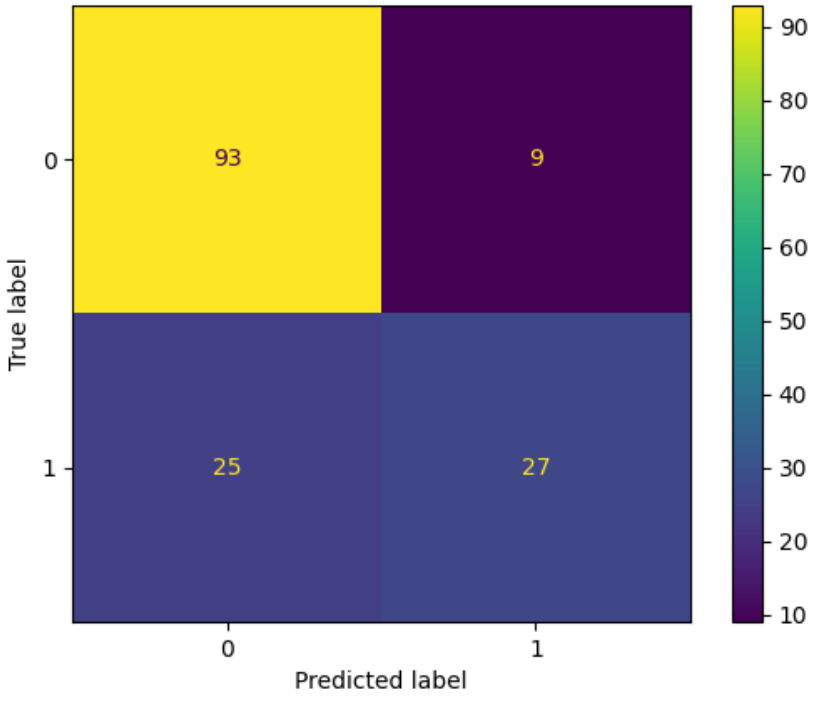


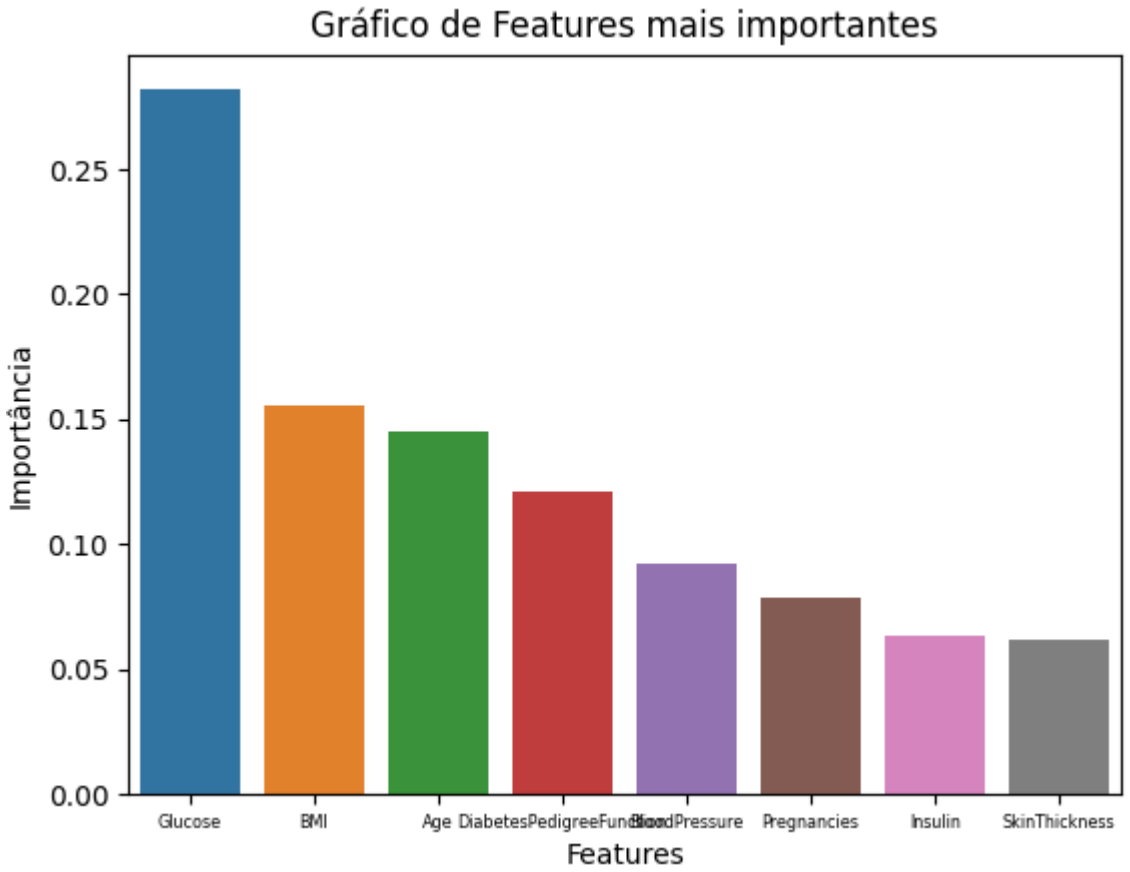


2.3 Resultados para SQRT estimadores



Precisão	Revocação	Medida F1	Acurácia (%)
0.75	0.51	0.61	77.9





2.4 Comparação de resultados com artigo

A seguir serão mostrados os resultados em comparação com os resultados apresentados no artigo Sisodia, D. & Sisodia, D. S. “Prediction of Diabetes using Classification Algorithms”, 10.1016/j.procs.2018.05.122. Observando que, os resultados do artigo em questão são para os algoritmos de classificação Naive Bayes, Support Vector Machine (SVM) e Decision Tree (árvore de decisão). Para essa comparação, será usado o resultado que obteve melhor precisão na floresta aleatória, ou seja, com estimadores sqrt.

	Nosso resultado	Naive Bayes	SVM	Decision Tree
Precisão	0.75	0.75	0.42	0.73
Revocação	0.51	0.76	0.65	0.73
Medida F1	0.61	0.76	0.51	0.73
Acurácia (%)	77.9	76.3	65.1	73.8

Conclusão

Ao se implementar o algoritmo de floresta aleatória, muitas pessoas tendem a pensar que quanto mais alto o número de árvores de decisão que ela gerar, melhores resultados ela obterá, mas podemos ver como isso nem sempre é verdade. Através deste estudo conseguimos mostrar os resultados obtidos para a floresta aleatória com 10 árvores, 100 árvores e o número de árvores estimado pela sqrt.

Vimos que para 10 árvores, a floresta obteve uma precisão maior do que a com 100 árvores, porém com acurácia menor. E a revocação para a de 10 estimadores, ficou abaixo de 0.5 e isso significa que este não é um bom resultado, pois isso significa que tivemos poucos resultados verdadeiramente relevantes. Já a comparação do sqrt com a de 100, nos mostra resultados interessantes, já que a estimativa pela sqrt é menor que 100, e ela obtém precisão e acurácia melhores do que a floresta com 100 árvores, e mesmo que a revocação seja menor, ainda está acima de 0.5, o que demonstra um bom resultado e nos mostra como a estimativa pela sqrt pode ser melhor do que apenas decidir um número alto de árvores para a floresta.

Vimos também um comparativo com o artigo Sisodia, D. & Sisodia, D. S. “Prediction of Diabetes using Classification Algorithms”, 10.1016/j.procs.2018.05.122, e podemos observar como a estimativa sqrt possui resultados de precisão e acurácia melhores que os algoritmos de Naive Bayes, SVM e Decision Tree, mostrando assim como floresta aleatória pode melhorar os processos de classificação em aprendizado de máquina.

Referências

Biblioteca sklearn: https://scikit-learn.org/stable/index.html
Biblioteca numpy: https://numpy.org/
Biblioteca pandas: https://pypi.org/project/pandas/
Biblioteca matplotlib: https://matplotlib.org/stable/users/installing.html
Biblioteca seaborn: https://seaborn.pydata.org/installing.html
Explicação e uma referência de como implementar floresta aleatória: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76
Referência de como implementar floresta aleatória com plotação de gráficos: https://www.datacamp.com/community/tutorials/random-forests-classifier-python
Referencial de como plotar matriz de confusão: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html