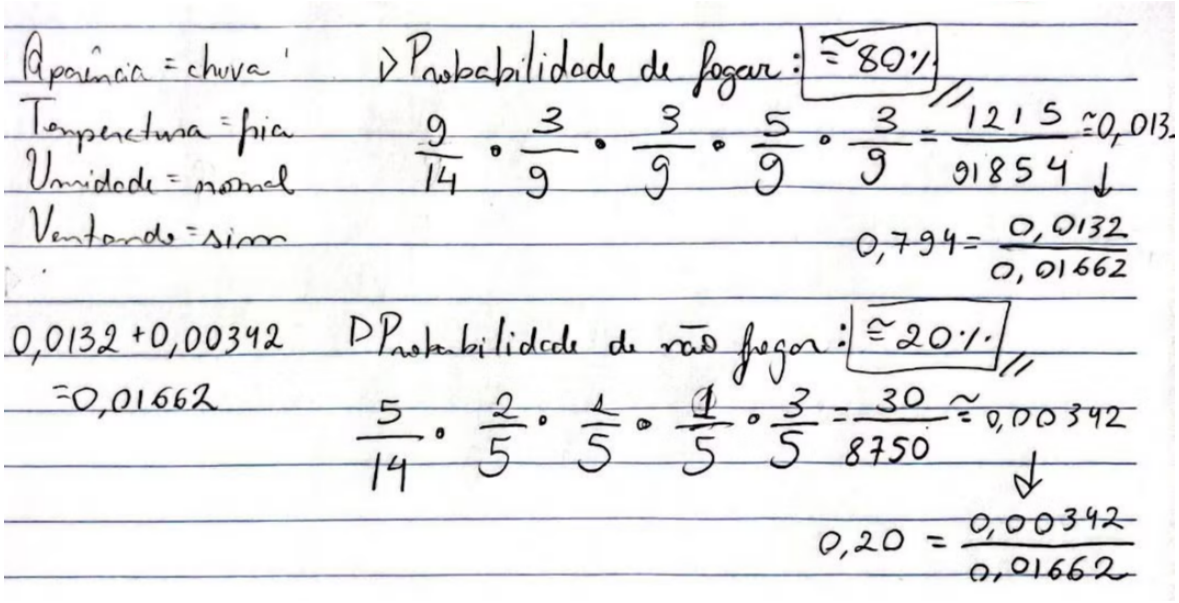
Lista 3 - IA

Sophia Carrazza Ventorim de Sousa

PUC Minas - 2024

1-

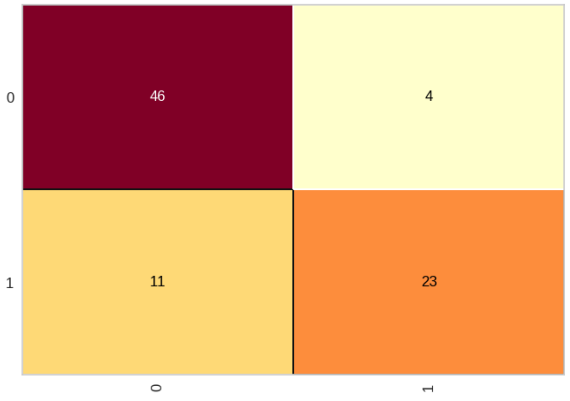
Raciocínio para a questão 01:



3.1- Random Forest:

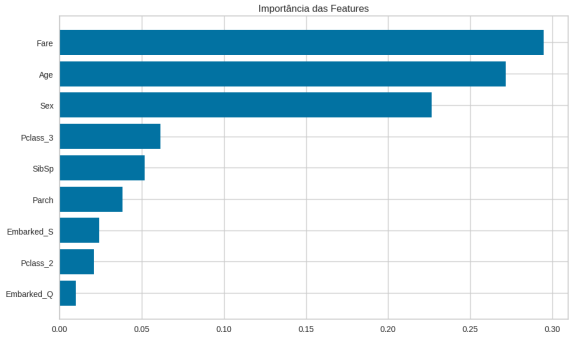
https://colab.research.google.com/drive/1vjj2bzcBVU0NWZ-Iwx9s3lCwsczNerHC?usp=sharing

- Acurácia do modelo: 0.8214285714285714
- Matriz de confusão: array([[90, 15], [19, 55]])




```
print(classification_report(y_teste, previsoes))
```

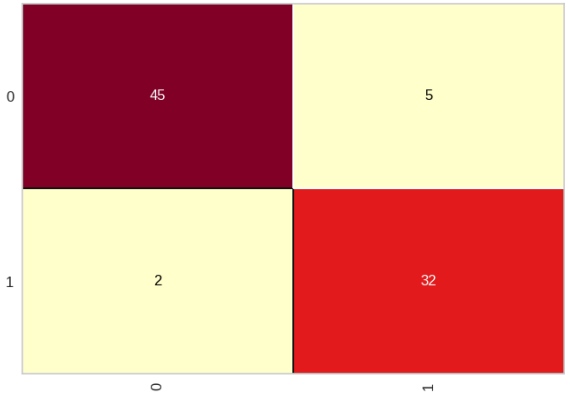
	precision	recall	f1-score	support
0	0.81	0.92	0.86	50
1	0.85	0.68	0.75	34
accuracy			0.82	84
macro avg	0.83	0.80	0.81	84
weighted avg	0.83	0.82	0.82	84



Naive Bayes:

https://colab.research.google.com/drive/1ddiiSAMYs40UGYA_1C4qGo4H0KKOLPm3?usp=sharing

- Acurácia do modelo: 0.9166666666666666
- Matriz de confusão: array([[45, 5], [2, 32]])




```
print(classification_report(y_teste, previsoes))
```

	precision	recall	f1-score	support
0	0.96	0.90	0.93	50
1	0.86	0.94	0.90	34
accuracy			0.92	84
macro avg	0.91	0.92	0.91	84
weighted avg	0.92	0.92	0.92	84

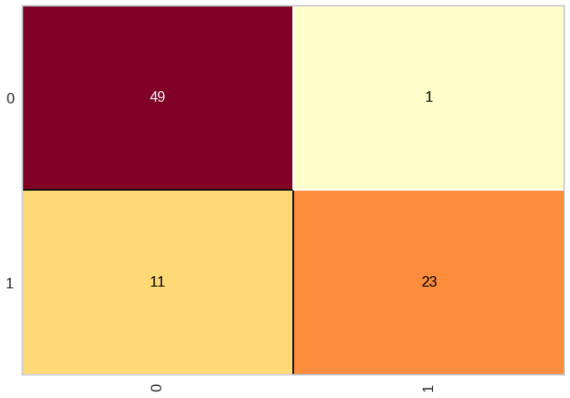
```
[14] modelo = GaussianNB(var_smoothing=1e-9)
```

Árvore de decisão:

https://colab.research.google.com/drive/1l_tyOX8w5ucqUzbHfPuQDCk7VOo8_k6h?usp=sharing

- Acurácia do modelo: 0.8571428571428571

- Matriz de confusão: array([[49, 1], [11, 23]])




```
[21] print(classification_report(y_teste, previsoes))

              precision    recall  f1-score   support

           0       0.82      0.98      0.89        50
           1       0.96      0.68      0.79        34

    accuracy                           0.86        84
   macro avg       0.89      0.83      0.84        84
weighted avg       0.87      0.86      0.85        84
```

3.2-

Código usado para ajustar os melhores hiperparâmetros:

Encontrando os melhores parametros

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import RandomizedSearchCV
```

```
[ ] # Definindo o modelo
    modelo = RandomForestClassifier()

    # Definindo a distribuição dos hiperparâmetros a serem testados
    param_dist = {
        'n_estimators': [10, 50, 100, 200],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4],
        'max_features': ['sqrt', 'log2', 0.5],
        'bootstrap': [True, False],
        'criterion': ['gini', 'entropy']
    }
```

```
[ ] # Configurando RandomizedSearchCV
    random_search = RandomizedSearchCV(estimator=modelo,
                                       param_distributions=param_dist,
                                       n_iter=100,
                                       cv=3,
                                       verbose=2,
                                       random_state=42,
                                       n_jobs=-1)
```

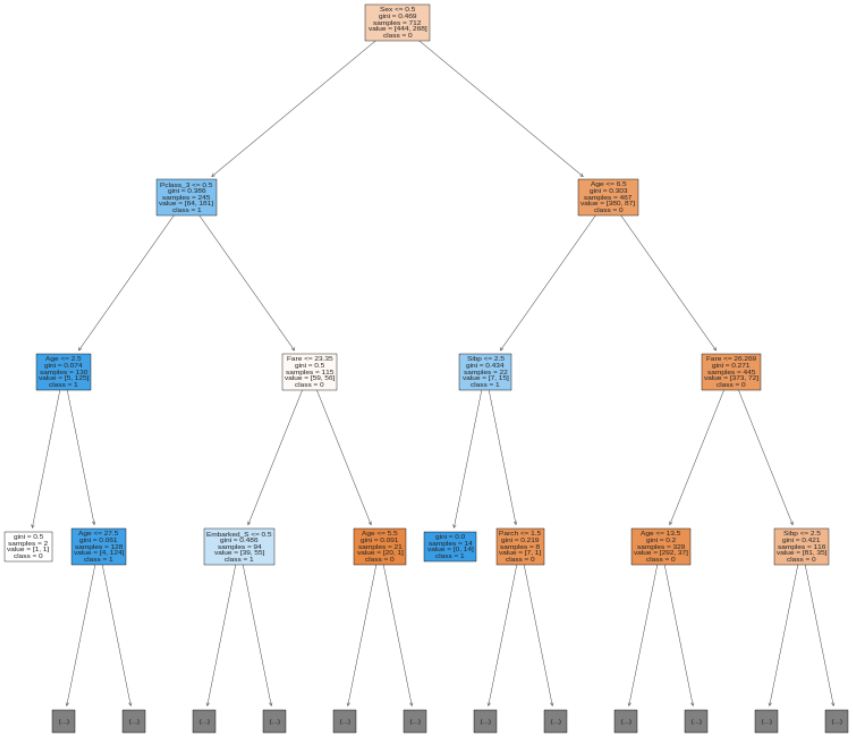
```
[ ] # Ajustando o modelo
    random_search.fit(X_treino, y_treino)
```

```
    # Melhores hiperparâmetros encontrados
    print(random_search.best_params_)
```

ajuste dos hiperparâmetros (do DecisionTree):

```
[46] modelo = DecisionTreeClassifier(criterion='gini', max_depth=10, min_samples_leaf=2)
     modelo.fit(X_treino, y_treino)
```

```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=2)
```



ajuste dos hiperparâmetros (do Random Forest):

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
{'n_estimators': 10, 'min_samples_split': 5, 'min_samples_leaf': 4, 'max_features': 'log2', 'max_depth': None, 'criterion': 'entropy', 'bootstrap': True}
```

```
modelo = RandomForestClassifier(n_estimators=200, max_features='log2', criterion='entropy', random_state = 42)
modelo.fit(X_treino, y_treino)
```

```
RandomForestClassifier(criterion='entropy', max_features='log2',
                       n_estimators=200, random_state=42)
```

3.3-

Código usado para o pré-processamento:

https://colab.research.google.com/drive/1D70ZVVhVnrcpHV67eOLalFvSdbi4qhiO?usp=sharing

O tratamento de valores nulos se deu nas colunas “Age”, “Cabin”e “Embarked” (os que possuiam valores nulos) da seguinte forma:

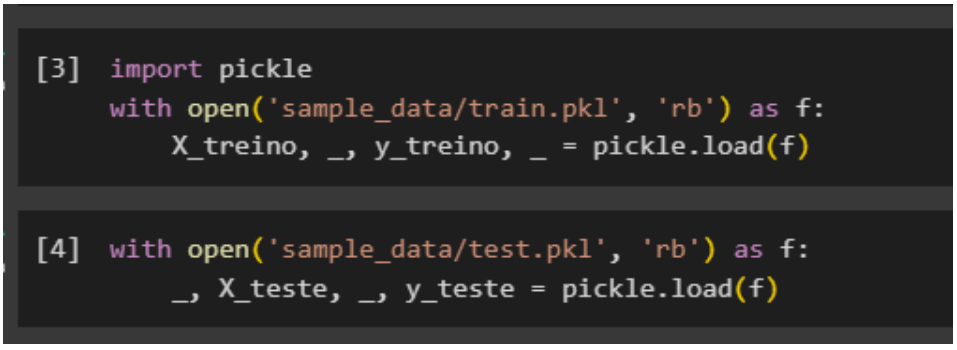
- Age: Os nulos foram substituídos pela mediana por serem variáveis numéricas com outliers;
- Cabin: Como havia muitos valores nulos,a coluna foi excluída;
- Embarked: Nullos foram substituídos pela moda, por terem poucos valores ausentes.

Outras colunas que também não auxiliam na classificação foram deletadas, como “PassengerId”, “Name” e ”Ticket”.

O LabelEncoder foi usado para o atributo “Sex”, por ter somente duas categorias transformáveis em 0 e 1.

O OneHotEncoder foi usado para os atributos "Embarked" e "Pclass, por serem categóricos e terem mais de duas categorias.

A base de teste foi combinada à base de resultados e tratada da mesma forma que a de treino, com a adição da substituição de uma linha nula do campo “Fare” pela moda.



```
[3]  import pickle
     with open('sample_data/train.pkl', 'rb') as f:
         X_treino, _, y_treino, _ = pickle.load(f)

[4]  with open('sample_data/test.pkl', 'rb') as f:
         _, X_teste, _, y_teste = pickle.load(f)
```