Lista 6

Link para github:
https://github.com/AHChaves/Aprendizado-de-Maquina

As questões 1 e 2 foram feitas imputando valores ausentes pela média.

A base utilizada foi titanic_train, que após o tratamento de dados possuía 891 dados, com o atributo de idade sendo o único com valores faltantes.

Q1 - Na realização dessa atividade foram criados 3 arquivos para cada algoritmo. Neles tínhamos os seguintes hiperparâmetros e códigos de implementação.
Árvore de decisão:

```
param_bayes = {
    'criterion':  ['gini', 'entropy'],
    'max_depth': Integer(1, 50),
    'min_samples_split': Integer(2, 20),
    'min_samples_leaf': Integer(1, 10),
}
✓ 0.0s
```

```
bayes_search = BayesSearchCV(DecisionTreeClassifier(), param_bayes, n_iter=50, cv=5, scoring='accuracy', random_state=42)
bayes_search.fit(X_treino, y_treino)
print("\nMelhores parâmetros encontrados com BayesSearchCV:")
print(bayes_search.best_params_)
print("Melhor pontuação de validação cruzada: {:.2f}".format(bayes_search.best_score_))
✓ 17.7s

Melhores parâmetros encontrados com BayesSearchCV:
OrderedDict({'criterion': 'entropy', 'max_depth': 24, 'min_samples_leaf': 10, 'min_samples_split': 2})
Melhor pontuação de validação cruzada: 0.80
```

Random Forest:

```python
param_bayes = {
    'criterion':  ['gini', 'entropy'],
    'max_depth': Integer(1, 50),
    'min_samples_split': Integer(2, 20),
    'min_samples_leaf': Integer(1, 10),
    'n_estimators': Integer(50, 120),
    'max_features': Integer(2, 5)
}
```

✓ 0.0s

```python
bayes_search = BayesSearchCV(RandomForestClassifier(), param_bayes, n_iter=50, cv=5, scoring='accuracy', random_state=42)
bayes_search.fit(X_treino, y_treino)
print("\nMelhores parâmetros encontrados com BayesSearchCV:")
print(bayes_search.best_params_)
print("Melhor pontuação de validação cruzada: {:.2f}".format(bayes_search.best_score_))
```

✓ 36.6s

```
Melhores parâmetros encontrados com BayesSearchCV:
OrderedDict({'criterion': 'gini', 'max_depth': 8, 'max_features': 2, 'min_samples_leaf': 6, 'min_samples_split': 18, 'n_estimators': 53})
Melhor pontuação de validação cruzada: 0.83
```

Naive Bayes:

```
bayes_search = BayesSearchCV(GaussianNB(), param_bayes, n_iter=50, cv=5, scoring='accuracy', random_state=42)
bayes_search.fit(X_treino, y_treino)
print("\nMelhores parâmetros encontrados com BayesSearchCV:")
print(bayes_search.best_params_)
print("Melhor pontuação de validação cruzada: {:.2f}".format(bayes_search.best_score_))
```

✓ 12.7s

```
/usr/local/lib/python3.12/dist-packages/skopt/optimizer/optimizer.py:517: UserWarning: The objective has been evaluated at point [3.303653629394814e-08] before, using
  warnings.warn(

Melhores parâmetros encontrados com BayesSearchCV:
OrderedDict({'var_smoothing': 1.9091131576909445e-06})
Melhor pontuação de validação cruzada: 0.80
```

```
param_bayes = {
    'var_smoothing': Real(1e-9, 1e-1, prior='log-uniform')
}
```

✓ 0.0s

a) O modelo que obteve a melhor pontuação, olhando pelo valor do Accuracy, foi o Random Forest. Os valores das métricas de avaliação podem ser observados nos resultados abaixo.
Árvore de decisão:

```
print(classification_report(y_teste, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.80	0.87	0.84	134
1	0.78	0.67	0.72	89
accuracy			0.79	223
macro avg	0.79	0.77	0.78	223
weighted avg	0.79	0.79	0.79	223

RF:

```
print(classification_report(y_teste, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.79	0.90	0.84	134
1	0.81	0.65	0.72	89
accuracy			0.80	223
macro avg	0.80	0.77	0.78	223
weighted avg	0.80	0.80	0.79	223

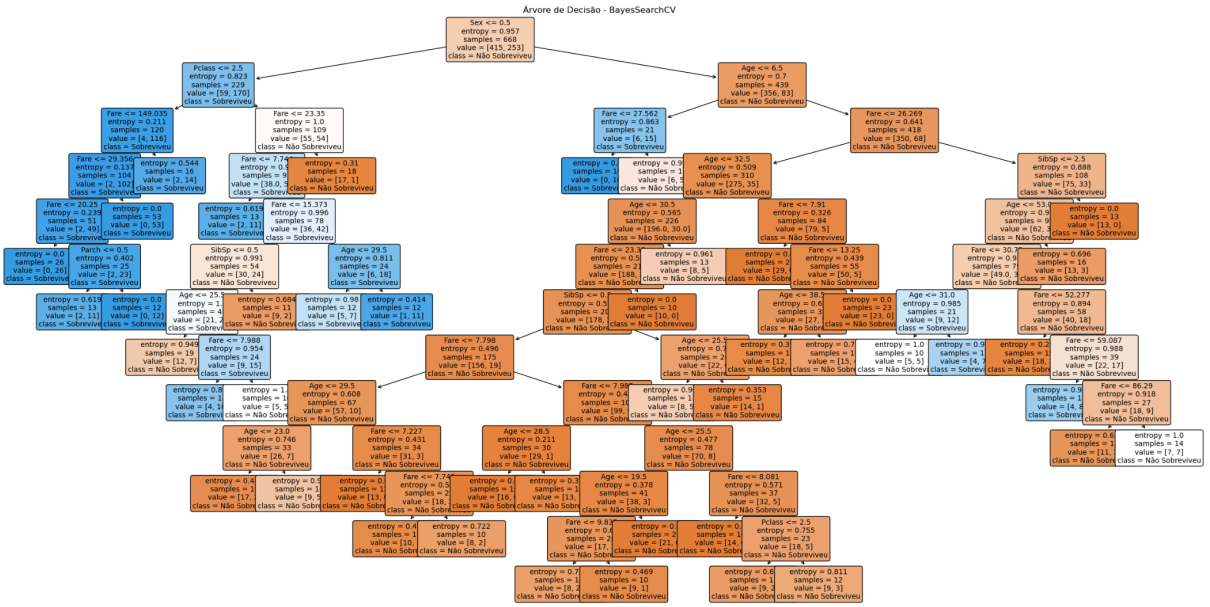
Naive Bayes:

```
print(classification_report(y_teste, previsoes_bayes))
✓ 0.0s
```

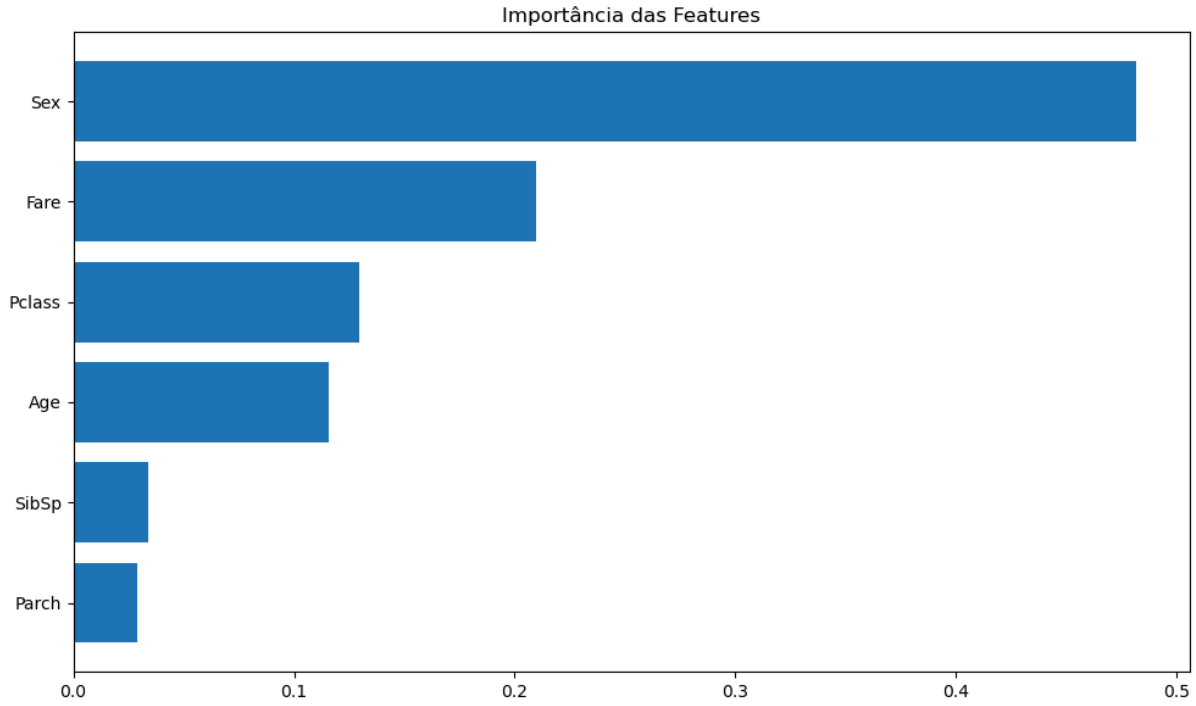
	precision	recall	f1-score	support
0	0.82	0.81	0.82	134
1	0.72	0.73	0.73	89
accuracy			0.78	223
macro avg	0.77	0.77	0.77	223
weighted avg	0.78	0.78	0.78	223

b) O atributo mais relevante foi o Sexo do individuo para todos os 3 modelos, isso se deve, provavelmente, pela diferença fisica entra os sexos masculino e feminino que impactaram na sobrevivencia do individuo. Quanto ao restante dos atributos houve uma similaridade entre a importância deles. Já no sentido dos algoritmos, essa diferença se deve de como eles definem os atributos mais importantes.

Árvore:



RF:

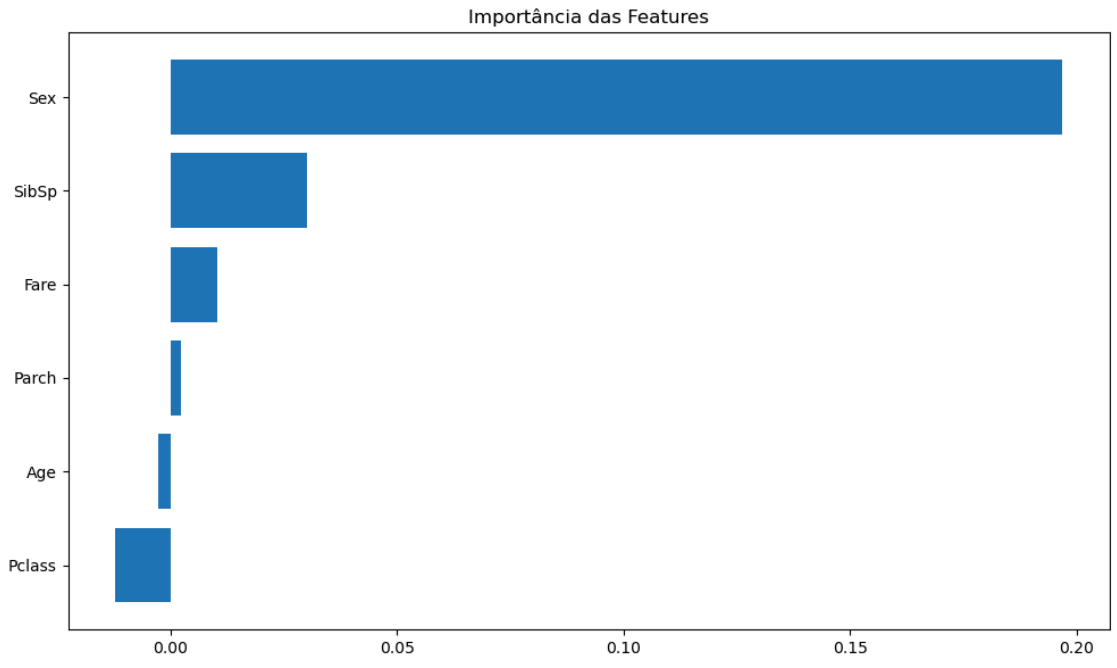


Bayes: Para o cálculo da importância das features deste modelo foi utilizado a função permutation_importance da biblioteca sklearn.inspection

```python
from sklearn.inspection import permutation_importance

# Calcular importância por permutação
result = permutation_importance(best_model_bayes, X_teste, y_teste, n_repeats=10, random_state=42)

# Obter as importâncias médias
importancias = result.importances_mean
```



Q2 - Nesta questão foram implementados métodos de balanceamento diferentes na base de dados. Esta implementação foi feita após decidir os conjuntos de treino e teste.

```
from imblearn.over_sampling import SMOTE
```
✓ 0.0s

```
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_treino, y_treino)
```
✓ 0.0s

```
from imblearn.under_sampling import TomekLinks
```
✓ 0.0s

```
balanceamento_under = TomekLinks(sampling_strategy='auto')
X_under, y_under = balanceamento_under.fit_resample(X_treino, y_treino)
```
✓ 0.0s

```
from imblearn.under_sampling import RandomUnderSampler
```
✓ 0.0s

```
undersample = RandomUnderSampler(random_state=42)
X_resampled, y_resampled = undersample.fit_resample(X_treino, y_treino)
```

```
from dsto_gan import DSTO_GAN
```
✓ 0.0s

```
dsto_gan = DSTO_GAN()
```
✓ 0.0s

```
X_resampled, y_resampled = dsto_gan.fit_resample(X_treino.values, y_treino.values)
```
✓ 1.9s

As bases balanceadas foram aplicadas em um modelo de Árvore de Decisão. Segue os resultados de cada modelo:

OverSampling:

```
print(classification_report(y_teste_over, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.79	0.81	0.80	134
1	0.70	0.67	0.69	89
accuracy			0.75	223
macro avg	0.74	0.74	0.74	223
weighted avg	0.75	0.75	0.75	223

UnderSampling:

```
print(classification_report(y_teste_under, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.84	0.81	0.82	134
1	0.72	0.76	0.74	89
accuracy			0.79	223
macro avg	0.78	0.79	0.78	223
weighted avg	0.79	0.79	0.79	223

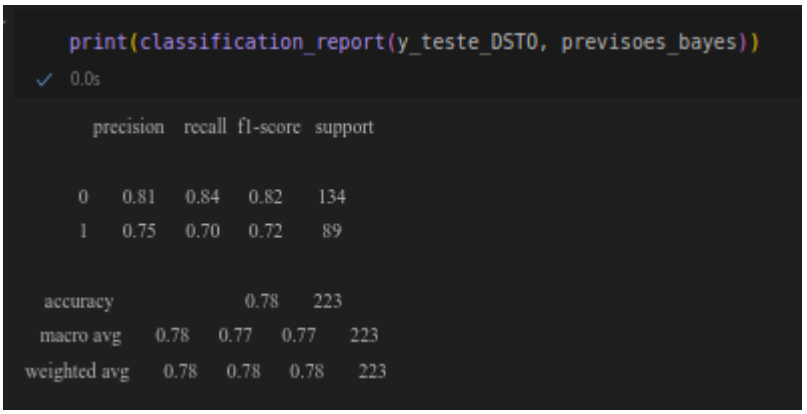
Random UnderSampling:

```
print(classification_report(y_teste_rd_under, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.81	0.82	0.82	134
1	0.73	0.72	0.72	89
accuracy			0.78	223
macro avg	0.77	0.77	0.77	223
weighted avg	0.78	0.78	0.78	223

DSTO:



```
print(classification_report(y_teste_DSTO, previsoes_bayes))
```

✓ 0.0s

	precision	recall	f1-score	support
0	0.81	0.84	0.82	134
1	0.75	0.70	0.72	89
accuracy			0.78	223
macro avg	0.78	0.77	0.77	223
weighted avg	0.78	0.78	0.78	223

Observando os valores médios de precisão, recall, e f1-score de cada base podemos concluir que:

Precisão:
maior = UnderSampling e DSTO
menor = OverSampling

Recall:
maior = UnderSampling
menor = OverSampling

f1-score:
maior = UnderSampling
menor = OverSampling

Com base nessas observações podemos concluir que a o método de UnderSampling foi o que obteve melhor resultados na árvore de decisão e o modelo de OverSampling foi o que teve os piores resultados.

Q3 - Nesta questão foi utilizado, novamente, o modelo de árvore de decisão para avaliar os métodos de imputação de dados. Vale lembrar que o único atributo que tinha valores vazios era a idade do indivíduo.

A imputação pela média teve o seguinte resultado

```
print(classification_report(y_teste_DSTO, previsoes_bayes))
```
✓ 0.0s

	precision	recall	f1-score	support
0	0.81	0.84	0.82	134
1	0.75	0.70	0.72	89
accuracy			0.78	223
macro avg	0.78	0.77	0.77	223
weighted avg	0.78	0.78	0.78	223

A moda apresentou os seguintes resultados

```
print(classification_report(y_teste, previsoes_bayes))
```
✓ 0.0s

	precision	recall	f1-score	support
0	0.81	0.87	0.84	134
1	0.78	0.70	0.74	89
accuracy			0.80	223
macro avg	0.80	0.78	0.79	223
weighted avg	0.80	0.80	0.80	223

O MissForest houve erros de implementação que impossibilitaram a execução do notebook, isso ocorreu por causa de uma diferença de versão entre ele e o Sklearn, o MissForest necessita de uma versão antiga do Sklearn.

Já o KNNImputer obteve os seguintes resultados

```
print(classification_report(y_teste, previsoes_bayes))
```
✓ 0.0s

	precision	recall	f1-score	support
0	0.80	0.87	0.84	134
1	0.78	0.67	0.72	89
accuracy			0.79	223
macro avg	0.79	0.77	0.78	223
weighted avg	0.79	0.79	0.79	223

Tendo em vista esses resultados podemos observar que imputar pela moda foi o que obteve uma melhora acurácia.

Q4 - Letra A

P(B) = 0,07 + 0,005 = 0,075

Gostar: P(A) = 9/17 = 0,529

Experiência (Alta): 4/9 = 0,444
interesse (Alto): 6/9 = 0,667
Horas (Baixas): 4/9 = 0,444
P(B | A) = 0,529 * 0,444* 0,667 * 0,444 = 0,07

P(A | B) = (0,07 / 0,075) * 100 = 93,33%

~Gostar: 8/17 = 0,471

Experiência: 1/8 = 0,125
interesse: 1/8 = 0,125
Horas: 5/8 = 0,625
P(B | A) = 0,471 * 0,125 * 0,125 * 0,625 = 0,005

P(A | B) = (0,005 / 0,075) * 100 = 6,67%

Q5 -

Um atributo que não aparece nenhuma vez em um conjunto de dados assumirá o valor 0 (zero) de ocorrência nas probabilidades calculadas. Isso causará porque aparecerão produtos ou divisões por zero no decorrer do fluxo do algoritmo.

Para corrigir este problema, podemos usar a correção Laplaciana, que se resume na adição de registros fictícios no conjunto de dados, de forma que não apareçam atributos de frequência zero. Ressalta-se que a adição de registros irá causar a mudança dos valores de probabilidade calculados inicialmente.