Importar base de treino

In [211…

```
import pandas as pd


def import_table(file_name: str, include_survived: bool = True) -> pd.DataFrame:
    cols = [
        "Survived",
        "Pclass",
        "Sex",
        "Age",
        "Fare",
        "Embarked",
    ]

    if not include_survived:
        cols.pop(0)

    table = pd.read_csv(
        file_name,
        usecols=cols,
    )

    table = pd.get_dummies(
        table,
        columns=["Sex", "Embarked"],
        prefix="",
        prefix_sep="",
    )

    table.loc[:, "Age"] = table["Age"].fillna(table["Age"].mean())
    table.loc[:, "Fare"] = table["Fare"].fillna(table["Fare"].mean())

    return table


train_table = import_table("train.csv")
display(train_table)
```

	Survived	Pclass	Age	Fare	female	male	C	Q	S
0	0	3	22.000000	7.2500	False	True	False	False	True
1	1	1	38.000000	71.2833	True	False	True	False	False
2	1	3	26.000000	7.9250	True	False	False	False	True
3	1	1	35.000000	53.1000	True	False	False	False	True
4	0	3	35.000000	8.0500	False	True	False	False	True
…	…	…	…	…	…	…	…	…	…
886	0	2	27.000000	13.0000	False	True	False	False	True
887	1	1	19.000000	30.0000	True	False	False	False	True
888	0	3	29.699118	23.4500	True	False	False	False	True
889	1	1	26.000000	30.0000	False	True	True	False	False
890	0	3	32.000000	7.7500	False	True	False	True	False

891 rows × 9 columns

Remover outliers

In [212…

```
def remove_outliers(df: pd.DataFrame):
    for col in df.select_dtypes(include=["float64", "int64"]).columns:
        # Calculate Q1 (25th percentile) and Q3 (75th percentile) for each numeric column
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Remove rows that contain outliers
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

    return df

train_table = remove_outliers(train_table)
display(train_table)
```

	Survived	Pclass	Age	Fare	female	male	C	Q	S
0	0	3	22.000000	7.2500	False	True	False	False	True
2	1	3	26.000000	7.9250	True	False	False	False	True
3	1	1	35.000000	53.1000	True	False	False	False	True
4	0	3	35.000000	8.0500	False	True	False	False	True
5	0	3	29.699118	8.4583	False	True	False	True	False
…	…	…	…	…	…	…	…	…	…
886	0	2	27.000000	13.0000	False	True	False	False	True
887	1	1	19.000000	30.0000	True	False	False	False	True
888	0	3	29.699118	23.4500	True	False	False	False	True
889	1	1	26.000000	30.0000	False	True	True	False	False
890	0	3	32.000000	7.7500	False	True	False	True	False

718 rows × 9 columns

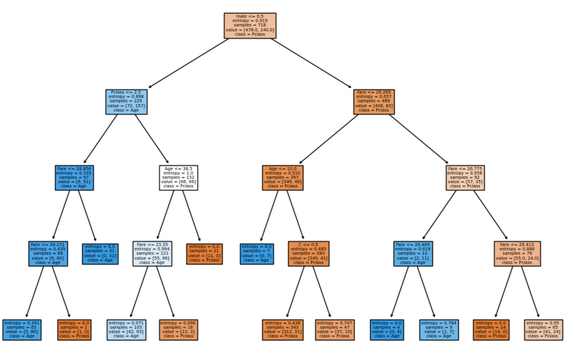
Criar Árvore de decisão

In [213…

```
from sklearn.tree._classes import DecisionTreeClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt


decision_tree = DecisionTreeClassifier(criterion='entropy', max_depth=4)
x = train_table.iloc[:, 1:]
decision_tree.fit(X=x, y=train_table["Survived"])

plt.figure(figsize=(12, 8))
plot_tree(decision_tree, feature_names=x.columns, class_names=x.columns, filled=True)
plt.show()
```



Importação tabela de teste

In [214…

```
test_table = import_table("test.csv", include_survived=False)
display(test_table)
```

	Pclass	Age	Fare	female	male	C	Q	S
0	3	34.50000	7.8292	False	True	False	True	False
1	3	47.00000	7.0000	True	False	False	False	True
2	2	62.00000	9.6875	False	True	False	True	False
3	3	27.00000	8.6625	False	True	False	False	True
4	3	22.00000	12.2875	True	False	False	False	True
…	…	…	…	…	…	…	…	…
413	3	30.27259	8.0500	False	True	False	False	True
414	1	39.00000	108.9000	True	False	True	False	False
415	3	38.50000	7.2500	False	True	False	False	True
416	3	30.27259	8.0500	False	True	False	False	True
417	3	30.27259	22.3583	False	True	True	False	False

418 rows × 8 columns

Importação resultados

In [215…

```
from sklearn.metrics import accuracy_score

test_table = import_table("test.csv", include_survived=False)
results = pd.read_csv("gender_submission.csv")
display(results)
```

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
…	…	…
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

Método Naive Bayes

In [216…

```
from sklearn.naive_bayes import GaussianNB

nb_model = GaussianNB()
nb_model.fit(x, train_table["Survived"])

print(f"Acurácia Naive Bayes: {accuracy_score(results["Survived"], nb_model.predict(test_table)) * 100:.2f}%")
```

```
Acurácia Naive Bayes: 88.52%
```

Método Random Forest

In [217…

```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier()
rf_model.fit(x, train_table["Survived"])

print(f"Acurácia RandomForest: {accuracy_score(results["Survived"], rf_model.predict(test_table)) * 100:.2f}%")
```

```
Acurácia RandomForest: 80.86%
```