Importar as bibliotecas que serão usadas:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Importar os dados treino e teste já criados:

```
train = pd.read_csv('/content/drive/My Drive/data/Titanic/train.csv')
test = pd.read_csv('/content/drive/My Drive/data/Titanic/test.csv')
```

Criar modelo de Machine Learning. Nesse caso, usamos o RandomForestClassifier:

```
modelo = RandomForestClassifier(n_estimators=100, n_jobs=-1, random_state=0)
```

Juntar as variáveis que serão usadas no modelo:

```
variaveis = ['Sex_bin', 'Age']
```

Criar função para transformar os dados do 'sexo' de cada passageiro para numérico, já que o modelo de ML usado não trabalha com strings:

```
def transformar_sexo(valor) :
  if valor == 'female' :
    return 1
  else:
    return 0

train['Sex_bin'] = train['Sex'].map(transformar_sexo)


train.head()
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far

Separar os dados de variáveis(x) que serão usadas no teste e os resultados esperados(y) para treinar o modelo:

Cumings

```
X = train[variaveis]
Y = train['Survived']
```

Briggs

Foi necessário tirar os dados sem informação (NaN) e para isso, de forma simples, só transformamos eles em um número sem influência, como -1:

```
X = X.fillna(-1)
```

No modelo 3, não vamos comparar nosso treino com o teste, mas sim criar um novo teste (valid) pelo o que temos do treino. Muitas vezes isso é necessário e existe uma função do sklearn que faz isso:

```
np.random.seed(0)
X_treino, X_valid, Y_treino, Y_valid = train_test_split(X, Y, test_size = 0.5)
```

Para garantir que todos tem o mesmo tamanho:

```
X_treino.shape, X_valid.shape, Y_treino.shape, Y_valid.shape
```

```
((445, 2), (446, 2), (445,), (446,))
```

Criar o modelo para os dados de treino:

```
modelo.fit(X_treino,Y_treino)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Criar a previsão e comparar com o teste(valid):

```
prev3 = modelo.predict(X_valid)
```

```
np.mean(Y_valid == prev3)
```

```
0.7690582959641256
```

Para comparar com o resultado mais alto que tivemos antes, que era supondo que todas as mulheres foram salvas:

```
prev1 = (1 == X_valid['Sex_bin'])
np.mean(y_valid == prev1)
```

```
0.7825112107623319
```