Importar as bibliotecas que serão usadas:

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifie
```

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()
```

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

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

1	2	1	1	[illegible]	female	38.0	1	0	PC 17599	71.283

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)
```

Criamos o modelo:

```
modelo.fit(X,Y)
```

```
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)
```

Tivemos que adaptar as informações dos dados do teste, assim como fizemos com os dados do treino:

```
test['Sex_bin'] = test['Sex'].map(transformar_sexo)
```

```
X_prev = test[variaveis]
X_prev = X_prev.fillna(-1)
X_prev.head()
```

	Sex_bin	Age
0	0	34.5
1	1	47.0
2	0	62.0
3	0	27.0
4	1	22.0

Previsão do modelo: (O problema é que ele retorna uma array, então tivemos que transformar em Series do pandas com uma cabeçalho e depois transformar em arquivo csv para enviar ao avaliador automático do Kanggle para saber como nos saimos:

```
prev = modelo.predict(X_prev)
prev
```

```
array([0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0])
```

```
sub = pd.Series(prev, index=test['PassengerId'], name='Survived')
```

```
sub.shape
```

```
(418,)
```

```
sub.to_csv('primeiro_modelo.csv')
```

```
!head -n10 primeiro_modelo.csv
```

```
PassengerId,Survived
892,0
893,1
894,0
895,1
896,1
897,0
898,1
899,0
900,1
```

O modelo alcançou um score de 0.71.. Isso não é muito ruim, mas não muito bom. Usando o modelo que só divide passageiros masculinos como mortos e femininos como vivos, funcionou melhor do que esse :/