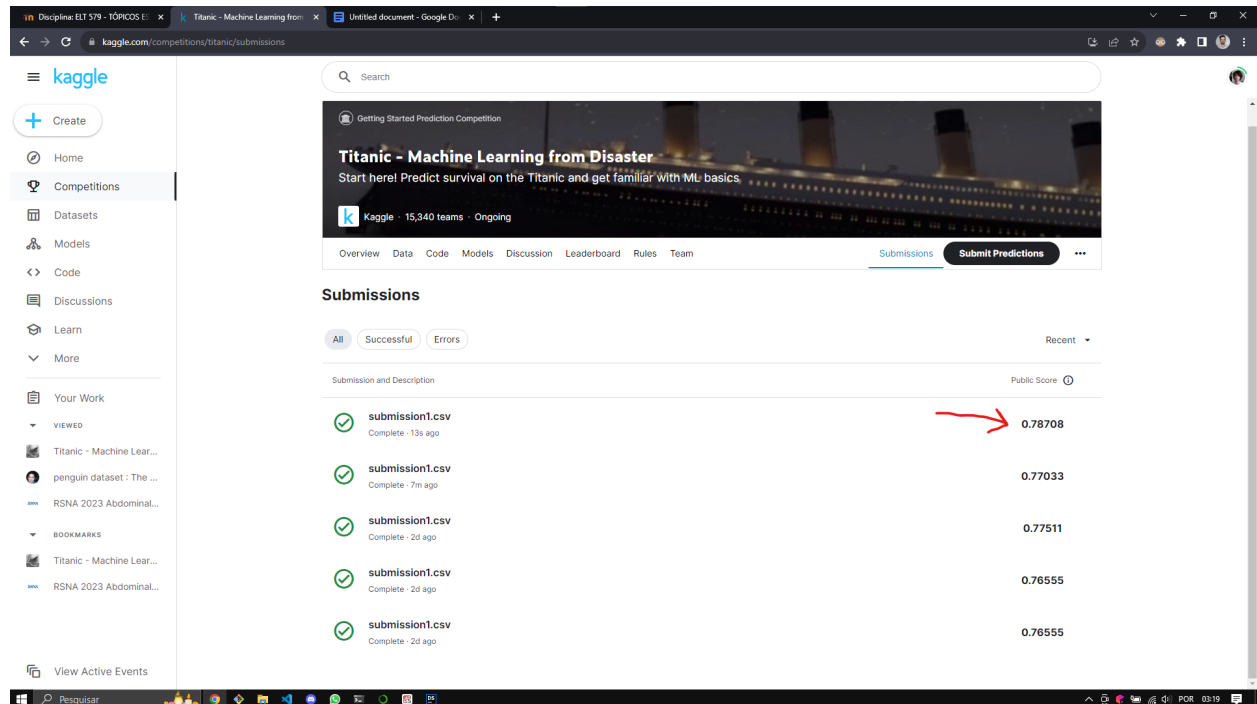


## Relatório do problema da semana 1: Titanic

Aluno: Gustavo Teixeira

Score atingido: 0.78708



Submission and Description	Public Score
submission1.csv Complete · 13s ago	0.78708
submission1.csv Complete · 7m ago	0.77033
submission1.csv Complete · 2d ago	0.77511
submission1.csv Complete · 2d ago	0.76555
submission1.csv Complete · 2d ago	0.76555

Atingi o score acima adicionando as seguintes features no meu modelo:

- Se é idoso
- Se é criança
- Se a pessoa tinha uma cabine
- Se o ticket dela continha apenas números
- Se o ticket dela continha apenas letras
- Quantidade de caracteres no ticket

O modelo foi o Random Forest.

Tentei usar o scikit optimize, mas tive muitos problemas com a versão, (aquele problema do int64 que o Sr abordou em aula). Não consegui resolver após horas de tentativas. Acabei fazendo sem esse optimize. Acredito que com ele eu poderia ter alcançado um resultado ainda melhor.

Abaixo segue o código que eu utilizei. Disponível também no meu Github através do link:

<https://github.com/gustavoteixeirah/PosGraduacao-Inteligencia-Artificial-e-Computacional/blob/main/ELT%20579%20-%20T%C3%93PICOS%20ESPECIAIS%20EM%20INTELIG%C3%8ANCIA%20ARTIFICIAL/semana%201/titanic1.py>

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-
```

```
"""
```

```
Created on Sat Oct 28 21:09:47 2023
```

```
@author: teixeira
```

```
"""
```

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import cross_val_score
```

```
# baseline 76%
```

```
### criar dados de treino e de teste
```

```
train = pd.read_csv('train.csv')
```

```
test = pd.read_csv('test.csv')
```

```
### Pre processamento dos dados
```

```
# descricao estatisticas das features numericas
```

```
est = train.describe()
```

```
print(train.info())
```

```
# verificar valores nulos ou NAN
```

```
print(train.isnull().sum())
```

```
print(test.isnull().sum())
```

```
### Mapear colunas
```

```
col = pd.Series(list(train.columns))
```

```
X_train = train.drop(['PassengerId', 'Survived'], axis = 1)
```

```
X_test = test.drop(['PassengerId'], axis = 1)
```

```
### Criar feature
```

```
import re
```

```
def criar_features(X):
```

```
    subs = {'female':1, 'male':0}
```

```

X['mulher'] = X['Sex'].replace(subs)

X['Age'] = X['Age'].fillna(X['Age'].mean())

X['Fare'] = X['Fare'].fillna(X['Fare'].mean())

X['Embarked'] = X['Embarked'].fillna('S')

subs = {'S':1, 'C':2, 'Q':3}
X['porto'] = X['Embarked'].replace(subs)

X['ehCrianca'] = 1
X['ehCrianca'] = np.where(X['Age'] < 15, 1, 0)

X['ehIdoso'] = 0
X['ehIdoso'] = np.where(X['Age'] > 55, 1, 0)

X['temCabine'] = np.where(pd.notnull(X['Cabin']), 1, 0)

X['contemApenasNumerosNoTicket'] = X['Ticket'].apply(lambda x: 1 if x.isdigit() else 0)

X['contemLetrasNoTicket'] = X['Ticket'].apply(lambda x: 1 if re.search('[a-zA-Z]', x) else 0)

X['tamanhoDoTicket'] = X['Ticket'].apply(lambda x: len(x))

return X

```

```

X_train = criar_features(X_train)
X_test = criar_features(X_test)

```

### Selecionar as features

```

features = ['Pclass', 'Age', 'SibSp',
            'Parch', 'Fare', 'mulher', 'porto', 'ehCrianca', 'ehIdoso', 'temCabine',
            'contemApenasNumerosNoTicket', 'contemLetrasNoTicket', 'tamanhoDoTicket']
X_train = X_train[features]
X_test = X_test[features]

y_train = train['Survived']

```

```
### Padronizacao das variaveis
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

### Random Forest
from sklearn.ensemble import RandomForestClassifier

model_rf = RandomForestClassifier(n_estimators=80, max_depth=3, random_state=0)

score = cross_val_score(model_rf, X_train, y_train, cv = 10)

model_rf_score = np.mean(score)
print(" Random Forest : ", model_rf_score)

### Modelo final

model_rf.fit(X_train, y_train)

y_pred = model_rf.predict(X_train)

### matrix de confusao
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_train, y_pred)

print(cm)

score = model_rf.score(X_train, y_train)
print(score)

### Previsao final

y_pred = model_rf.predict(X_test)

### Criar arquivo de submissao

submission = pd.DataFrame(test["PassengerId"])
submission["Survived"] = y_pred

submission.to_csv('submission1.csv', index = False)
```

