In [4]:

```python
# Importamos las librerías necesarias
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from matplotlib import style
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import cross_val_score
import warnings
```

In [5]:

```python
# Leyendo los datos
test = pd.read_csv("/content/test.csv")
train = pd.read_csv("/content/train.csv")
```

In [6]:

```python
# Quitamos Passenger Id del dataset de train
train = train.drop(['PassengerId'], axis=1)
```

In [7]:

```python
# Rellenar los valores faltantes en la columna Embarked con el valor más común ('S')
common_value = 'S'
data = [train, test]

for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna(common_value)
```

In [8]:

```python
# Asignar un número a cada valor de la columna Embarked
ports = {"S": 0, "C": 1, "Q": 2}
for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].map(ports)
```

In [9]:

```python
# Mapear la columna Sex (0 para mujer, 1 para hombre)
gender = {"female": 0, "male": 1}
for dataset in data:
    dataset['Sex'] = dataset['Sex'].map(gender)
```

In [10]:

```python
# Extraer el titulo de los nombres
train['Title'] = train['Name'].apply(lambda x: x.split(',')[1].split('.')[0].strip())
test['Title'] = test['Name'].apply(lambda x: x.split(',')[1].split('.')[0].strip())
```

In [11]:

```python
# Rellenar valores faltantes de edad en el dataset de train

# DataFrame Mr
df_mr = train[train["Title"] == "Mr"]
# Interpolación
df_mr.loc[:, "Age"] = df_mr["Age"].interpolate(method = "linear")

# DataFrame Miss
df_miss = train[train["Title"] == "Miss"]
# Interpolación
df_miss.loc[:, "Age"] = df_miss["Age"].interpolate(method = "linear")

# DataFrame Mrs
df_mrs = train[train["Title"] == "Mrs"]
# Interpolación
```

```
df_mrs.loc[:, "Age"] = df_mrs["Age"].interpolate(method = "linear")

# DataFrame Master
df_master = train[train["Title"] == "Master"]
# Interpolación
df_master.loc[:, "Age"] = df_master["Age"].interpolate(method = "linear")

# DataFrame de todo lo demás
df_extra = train[~train["Title"].isin(["Mr", "Miss", "Mrs", "Master"])]
# Parchar el único NaN con la media de las personas con el título Dr
mean_age_dr = df_extra[df_extra["Title"] == "Dr"]["Age"].mean()
df_extra.loc[(train["Title"] == "Dr") & (train["Age"].isnull()), "Age"] = mean_age_dr

# Juntar todos los DF
df_list = [df_mr, df_miss, df_mrs, df_master, df_extra]
train = pd.concat(df_list, ignore_index = True)

# Rellenar valores faltantes de edad en el dataset de test

# DataFrame Mr
df_mr = test[test["Title"] == "Mr"]
# Interpolación
df_mr.loc[:, "Age"] = df_mr["Age"].interpolate(method = "linear")

# DataFrame Miss
df_miss = test[test["Title"] == "Miss"]
# Interpolación
df_miss.loc[:, "Age"] = df_miss["Age"].interpolate(method = "linear")

# DataFrame Mrs
df_mrs = test[test["Title"] == "Mrs"]
# Interpolación
df_mrs.loc[:, "Age"] = df_mrs["Age"].interpolate(method = "linear")

# DataFrame Master
df_master = test[test["Title"] == "Master"]
# Interpolación
df_master.loc[:, "Age"] = df_master["Age"].interpolate(method = "linear")

# DataFrame de todo lo demás
df_extra = test[~test["Title"].isin(["Mr", "Miss", "Mrs", "Master"])]

# Calcular la media de la edad para títulos "Ms" en df_extra
mean_age_ms = df_mrs[df_mrs["Title"] == "Mrs"]["Age"].mean()

# Rellenar los valores faltantes en df_extra para las filas donde el título es "Ms"
df_extra.loc[(test["Title"] == "Ms") & (test["Age"].isnull()), "Age"] = mean_age_ms

# Juntar todos los DF
df_list = [df_mr, df_miss, df_mrs, df_master, df_extra]
test = pd.concat(df_list, ignore_index = True)
```

In [12]:

```
# Vemos que el dataset de Fare tiene un valor faltante este valor simplemente lo relle
mean_fare_pclass_3 = test[test['Pclass'] == 3]['Fare'].mean()

test.loc[(test['Pclass'] == 3) & (test['Fare'].isna()), 'Fare'] = mean_fare_pclass_3
```

In [13]:

```
# Agrupar titulos
def transform_title(title):
    rango_alto = ['Col','Major','Sir','Dr']
    hombres = ['Mr','Don','Jonkheer','Capt','Rev']
    mujeres = ['Lady','Miss','Mlle','Mme','Mrs','Ms','the Countess']
    unchanged_titles = ['Master']

    if title in rango_alto:
        return 'Ra'
    elif title in hombres:
        return 'Mr'
    elif title in mujeres:
        return 'Mrs'
    elif title in unchanged_titles:
        return title
    else:
        return 'Other'

train['Title'] = train['Title'].apply(transform_title)
test['Title'] = test['Title'].apply(transform_title)
```

In [14]:

```
# Vemos que las primeras dos letras de los tickets tienen algo que decirnos, por ejemp
train['Ticket_2letter'] = train['Ticket'].apply(lambda x: x[:2])
test['Ticket_2letter'] = test['Ticket'].apply(lambda x: x[:2])

train['Ticket_len'] = train['Ticket'].apply(lambda x: len(x))
test['Ticket_len'] = test['Ticket'].apply(lambda x: len(x))
```

In [15]:

```
# Asignar un numero a cada titulo
titulo = {"Mr": 0, "Mrs": 1, "Ra": 2, "Master":3,"Other":4}
for dataset in data:
    dataset['Title'] = dataset['Title'].map(titulo)
```

In [16]:

```
# Calcular el tamaño de la familia y clasificarlo
train['Tam_Fam'] = train['SibSp'] + train['Parch'] + 1
test['Tam_Fam'] = test['SibSp'] + test['Parch'] + 1

train['Tipo_Fam'] = pd.cut(train['Tam_Fam'], [0,1,4,7,11], labels=['Solo','Chico','Gra
test['Tipo_Fam'] = pd.cut(test['Tam_Fam'], [0,1,4,7,11], labels=['Solo','Chico','Granc
```

In [17]:

```
# Eliminar columnas que no utilizaremos
train = train.drop(['Name', 'SibSp', 'Parch'], axis=1)
test = test.drop(['Name', 'SibSp', 'Parch'], axis=1)
```

In [18]:

```
# Mapear los valores de Tipo_Fam
data = [train, test]
familia = {"Solo": 0, "Chico": 1, "Grande": 2, "Muy Grande": 3}
for dataset in data:
    dataset['Tipo_Fam'] = dataset['Tipo_Fam'].map(familia)
```

In [19]:

```
# La primera letra de cada cabina representa el piso en el que estaba en el titanic
train['Cabin'] = train['Cabin'].str[0]
test['Cabin'] = test['Cabin'].str[0]

# Sustituir los NaN en cabin con 'N'
train['Cabin'] = train['Cabin'].fillna('N')
test['Cabin'] = test['Cabin'].fillna('N')

# Hay solo un valor de T que no sabemos que significa asi que lo sustituimos por el va
train['Cabin'] = train['Cabin'].str.replace('T', 'A')
```

```
test['Cabin'] = test['Cabin'].str.replace('T', 'A')

# Mapeamos los valores de cabina
cabina = {"N": 0, "A": 1, "B": 2, "C": 3, "D":4, "E":5, "F":6, "G":7}
for dataset in data:
    dataset['Cabin'] = dataset['Cabin'].map(cabina)
```

In [20]:

```
# Convertir a int Tipo_Fam y Age
for dataset in data:
    dataset['Tipo_Fam'] = dataset['Tipo_Fam'].astype(int)
    dataset['Age'] = dataset['Age'].astype(int)
```

In [21]:

```
# Caracteristicas y variable objetivo
y = train['Survived']
features = ['Pclass', 'Fare', 'Title', 'Embarked', 'Tipo_Fam', 'Age','Ticket_len','Tic
X = train[features]
```

In [22]:

```
# Codificación One-Hot para las columnas categóricas
onehot = OneHotEncoder(handle_unknown='ignore', sparse=False)

# Transformamos las columnas categóricas
categorical_cols = ['Pclass', 'Title', 'Embarked', 'Tipo_Fam','Ticket_len','Ticket_2le
X_categorical = pd.DataFrame(onehot.fit_transform(X[categorical_cols]))

# Reiniciamos los indices
X_categorical.columns = onehot.get_feature_names_out(categorical_cols)
X_categorical.index = X.index

# Concatenamos con las columnas numéricas
X_numerical = X[['Fare', 'Age']]
X_final = pd.concat([X_categorical, X_numerical], axis=1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:975: Futur
eWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed
in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [23]:

```
# Entrenar el modelo RandomForest
model = RandomForestClassifier(random_state=0, n_estimators=500, max_depth=5)
model.fit(X_final, y)
```

Out[23]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=5, n_estimators=500, random_state=0)
```

In [24]:

```
# Evaluación con validación cruzada
cross_val_score = cross_val_score(model, X_final, y, cv=10)
print(f'Cross validation score: {cross_val_score.mean():.3f}')
```

```
Cross validation score: 0.771
```

In [25]:

```
# Preparar el conjunto de prueba para predicción
X_test = test[features]

# Transformamos las columnas categóricas de X_test
X_test_categorical = pd.DataFrame(onehot.transform(X_test[categorical_cols]))

# Reiniciamos los indices
```

```
X_test_categorical.columns = onehot.get_feature_names_out(categorical_cols)
X_test_categorical.index = X_test.index

# Unimos los dataframes categoricos y numericos
X_test_numerical = X_test[['Fare', 'Age']]
X_test_final = pd.concat([X_test_categorical, X_test_numerical], axis=1)
```

In [26]:

```
# Hacer predicciones y guardarlas en un archivo .csv
predictions = model.predict(X_test_final)
output = pd.DataFrame({'PassengerId': test.PassengerId, 'Survived': predictions})
output.to_csv("my_submission.csv", index=False)
```

In [27]:

```
X_test_final.head()
```

Out[27]:

	Pclass_1	Pclass_2	Pclass_3	Title_Master	Title_Mr	Title_Mrs	Title_Ra	Embarked_0	Embarked_1	I
0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	
1	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	
3	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	
4	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	

5 rows × 95 columns

In [28]:

```
X_final.head()
```

Out[28]:

	Pclass_1	Pclass_2	Pclass_3	Title_Master	Title_Mr	Title_Mrs	Title_Ra	Embarked_0	Embarked_1	I
0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	
1	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	
2	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	
4	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	

5 rows × 95 columns

In [29]:

```
# Evaluar la importancia de las características
feature_importances = pd.DataFrame({
    'Feature': X_final.columns,
    'Importance': model.feature_importances_
}).sort_values(by='Importance', ascending=False)

print("Importancia de las características:")
print(feature_importances)
```

```
Importancia de las características:
              Feature  Importance
4            Title_Mr    0.238394
5           Title_Mrs    0.192975
93               Fare    0.082485
2            Pclass_3    0.064793
85            Cabin_0    0.056236
..                ...         ...
56  Ticket_2letter_45    0.000011
61  Ticket_2letter_72    0.000001
83  Ticket_2letter_W/    0.000000
72  Ticket_2letter_Fa    0.000000
67  Ticket_2letter_A4    0.000000

[95 rows x 2 columns]
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
%%shell
jupyter nbconvert --to html /content/Port_Imple_WFRAMEWORK_A01571214_Lautaro_Coteja.ip
```