Who Survived the Titanic?

派遣会社(Agency)

Introduction

In this project, we will work on a comprehensive classification example using the dataset from the Titanic: Machine Learning from Disaster competition hosted on Kaggle.

The Titanic competition aims to predict the survival outcome of passengers aboard the Titanic ship based on various features such as age, gender, ticket class, and more. By analyzing this dataset and building a classification model, we can gain insights into the factors that influenced survival rates during the tragic event.

To accomplish this, we will utilize a combination of data analysis, feature engineering, and machine learning techniques. Our goal is to develop a robust classification model that can accurately predict whether a passenger survived or not.

Titanic

Through this project, we will delve into the entire data science pipeline, starting from data exploration and preprocessing to model selection, training, and evaluation. We will also employ various techniques to handle missing values, handle categorical variables, and optimize our model's performance.

Loading the libraries we need and libraries we require:

- %matplotlib inline: This is a magic command in Jupyter Notebook that allows the generated plots to be displayed inline within the notebook itself.
- numpy (np): This library provides support for efficient numerical operations and array manipulation in Python.
- pandas (pd): Pandas is a powerful data manipulation library that provides data structures and functions for working with structured data, such as CSV files or database tables. It offers flexible data handling capabilities and enables easy data manipulation and analysis.

- re: This library is the regular expression module in Python. It provides functions and methods for working with regular expressions, allowing pattern matching and text manipulation.

- scipy.stats (from scipy.stats import norm): This module from the SciPy library provides a wide range of statistical functions and distributions. The norm function specifically deals with the normal (Gaussian) distribution.

- seaborn: Seaborn is a data visualization library based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics.

- matplotlib.pyplot (plt): Matplotlib is a widely used plotting library in Python. The pyplot module provides a collection of functions that enable the creation of various types of plots, such as line plots, bar charts, histograms, and scatter plots.

By utilizing these libraries in our project, we can perform data manipulation, statistical analysis, and create visually appealing plots to communicate your findings effectively.

In [2]:

```
%matplotlib inline

import numpy as np
import pandas as pd
import re as re
from scipy.stats import norm
!pip install seaborn
import seaborn as sns
import matplotlib.pyplot as plt
```

```
Collecting seaborn
  Downloading seaborn-0.12.2-py3-none-any.whl (293 kB)
                                              0.0/293.3 kB ? eta -:--:--
     ---------------------------------      256.0/293.3 kB 7.7 MB/s eta 0:00:01
     -------------------------------------- 293.3/293.3 kB 4.5 MB/s eta 0:00:00
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\panasonic\anaconda3\lib
\site-packages (from seaborn) (1.25.0)
Requirement already satisfied: pandas>=0.25 in c:\users\panasonic\anaconda3\lib\site-pac
kages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\panasonic\anaconda3\l
ib\site-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\panasonic\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\panasonic\anaconda3\lib\site-pac
kages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\panasonic\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\panasonic\anaconda3\lib\sit
e-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\panasonic\anaconda3\lib\site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\panasonic\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\panasonic\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\panasonic\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\panasonic\anaconda
3\lib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (5.2.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\panasonic\anaconda3\lib\site-pac
kages (from pandas>=0.25->seaborn) (2022.7)
Requirement already satisfied: zipp>=3.1.0 in c:\users\panasonic\anaconda3\lib\site-pack
ages (from importlib-resources>=3.2.0->matplotlib!=3.6.1,>=3.1->seaborn) (3.11.0)
Requirement already satisfied: six>=1.5 in c:\users\panasonic\anaconda3\lib\site-package
```

```
s (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
Installing collected packages: seaborn
Successfully installed seaborn-0.12.2
```

Loading the dataset

In [3]:
```
df = pd.read_csv('titanic/titanic.csv', header = 0)
```

Data Description

Our dataset consists of 12 columns or variables, out of which 3 (`Age` , `Cabin` , and `Embarked`) have missing values. The variable we want to predict is `Survived` , which indicates whether the passenger survived the tragedy of the Titanic.

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [6]:
```
df.head()
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [7]:
```
df.describe()
```

Out[7]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Data transformation

We will analyse variable by variable in order to elucidate what transformations it will need in order to be used for modelling the data.

`PassengerId`

Passenger identifier. This is a numerical correlative variable, which has no predictive value. We will not include it in our model.

In [9]:
```
df['PassengerId'].head(10)
```

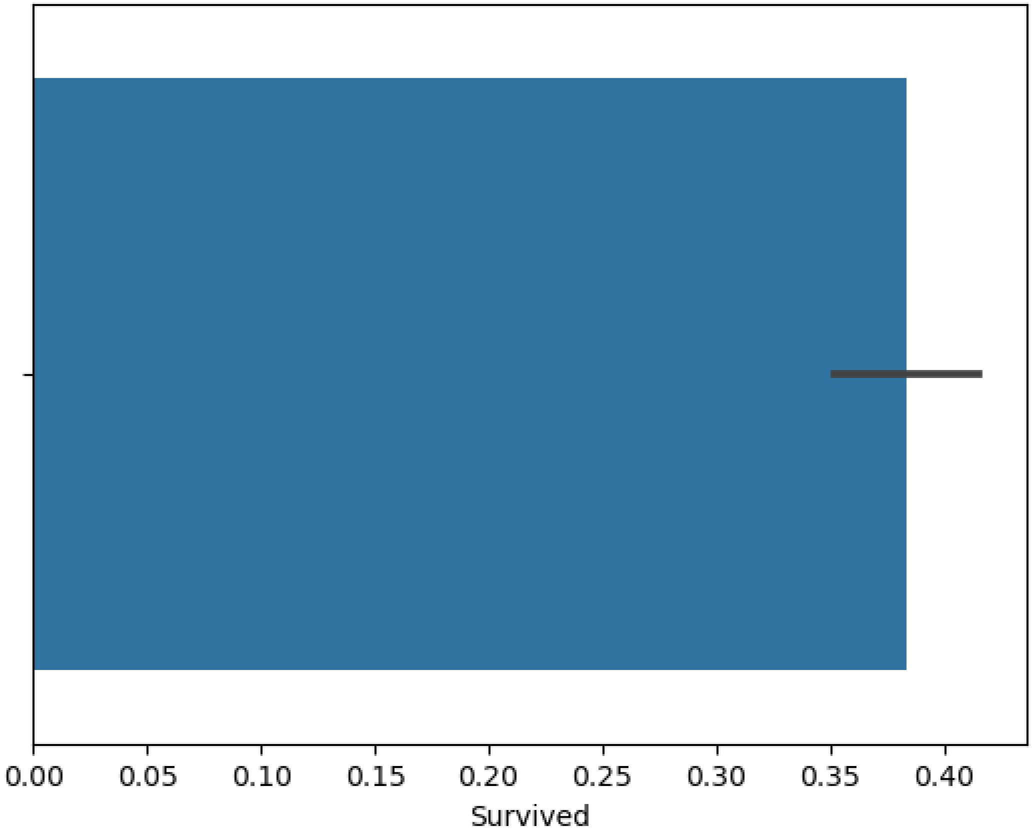
Out[9]:
```
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
Name: PassengerId, dtype: int64
```

In [11]:
```
df.drop(['PassengerId'], axis=1, inplace=True)
```

`Survived` Indicates whether the passenger survived (`1`) or not (`0`). This will be our predictor variable. We see that there are about 38% of passengers who survived.

In [12]:
```
sns.barplot(x="Survived", data=df)
```

Out[12]:
```
<Axes: xlabel='Survived'>
```



In [13]:
```
df.describe()['Survived']
```

Out[13]:
```
count    891.000000
mean       0.383838
std        0.486592
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        1.000000
Name: Survived, dtype: float64
```

`Pclass` Class of the boarding ticket. It is a categorical variable, with three different values corresponding to first, second and third class. We see that it is correlated with survival: the higher the class, the more survivors there are.

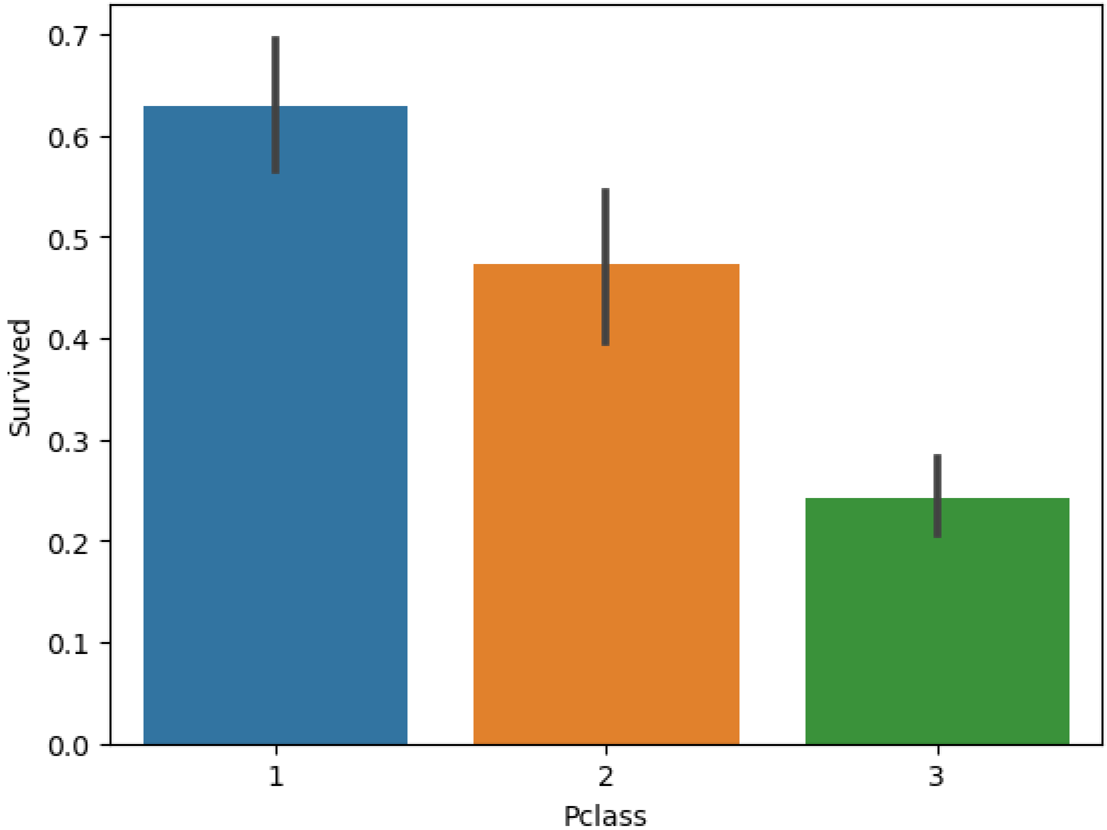
In [18]:
```
df[['Pclass', 'Survived']].groupby(['Pclass'],as_index=True).mean()
```

Out[18]:

Pclass	Survived
1	0.629630
2	0.472826
3	0.242363

In [19]:
```
sns.barplot(x="Pclass", y="Survived", data=df)
```

Out[19]:
```
<Axes: xlabel='Pclass', ylabel='Survived'>
```



Sex

Sex of the passenger. This is a categorical variable with two values, `male` and `female`. We will convert it to binary for our analysis.

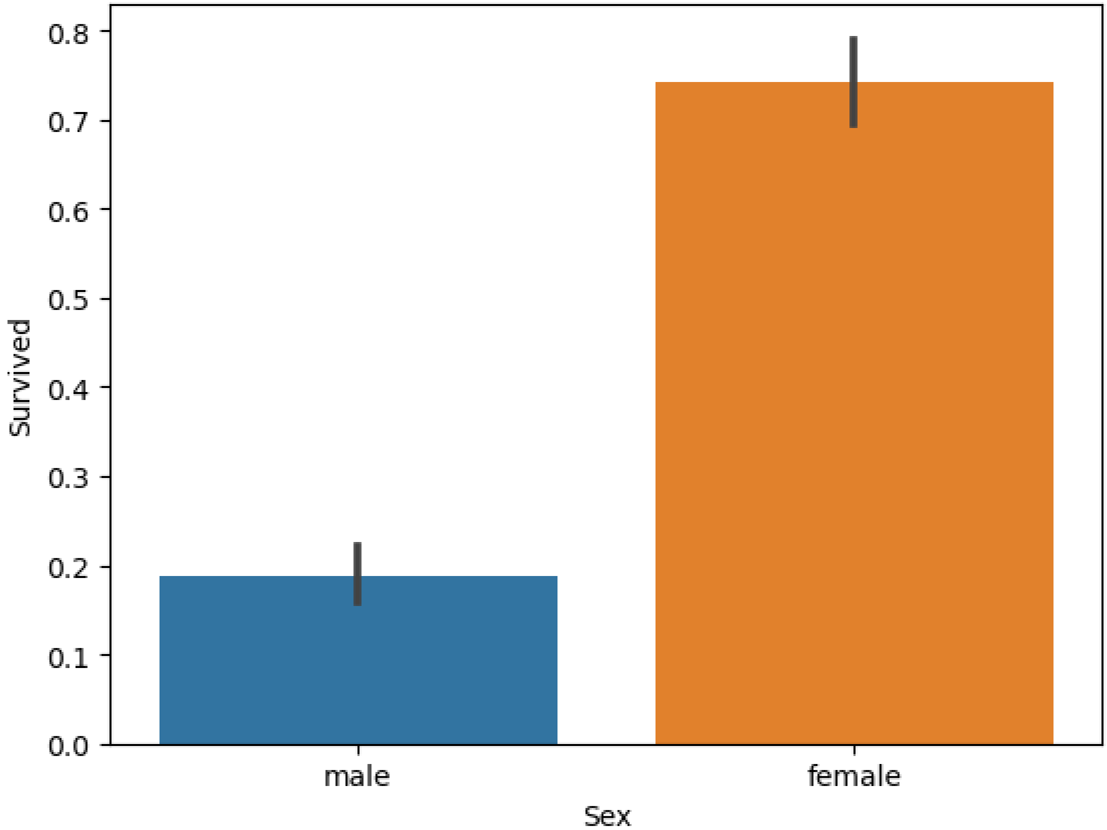
In [23]:
```
df[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean()
```

Out[23]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

In [24]:
```
sns.barplot(x="Sex", y="Survived", data=df)
```

Out[24]:
```
<Axes: xlabel='Sex', ylabel='Survived'>
```



In [ ]:
```
converting the categorical 'Sex' column into a boolean representation for the purpose of
where the sex of a passenger will be represented as True (if male) or False (if female).
```

In [25]:
```
df['Sex'] = df['Sex'] == 'male'
```

In [26]:
```
print(df)
```

```
     Survived  Pclass                                               Name  \
0           0       3                            Braund, Mr. Owen Harris
1           1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2           1       3                             Heikkinen, Miss. Laina
3           1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4           0       3                           Allen, Mr. William Henry
..        ...     ...                                                ...
886         0       2                              Montvila, Rev. Juozas
887         1       1                       Graham, Miss. Margaret Edith
888         0       3           Johnston, Miss. Catherine Helen "Carrie"
889         1       1                              Behr, Mr. Karl Howell
890         0       3                                Dooley, Mr. Patrick

       Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked
0     True  22.0      1      0         A/5 21171   7.2500   NaN        S
1    False  38.0      1      0          PC 17599  71.2833   C85        C
2    False  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3    False  35.0      1      0            113803  53.1000  C123        S
4     True  35.0      0      0            373450   8.0500   NaN        S
..     ...   ...    ...    ...               ...      ...   ...      ...
886   True  27.0      0      0            211536  13.0000   NaN        S
887  False  19.0      0      0            112053  30.0000   B42        S
888  False   NaN      1      2        W./C. 6607  23.4500   NaN        S
889   True  26.0      0      0            111369  30.0000  C148        C
890   True  32.0      0      0            370376   7.7500   NaN        Q

[891 rows x 11 columns]
```

`SibSp`

Number of siblings or spouses of the passenger on the Titanic. This is a numerical variable.

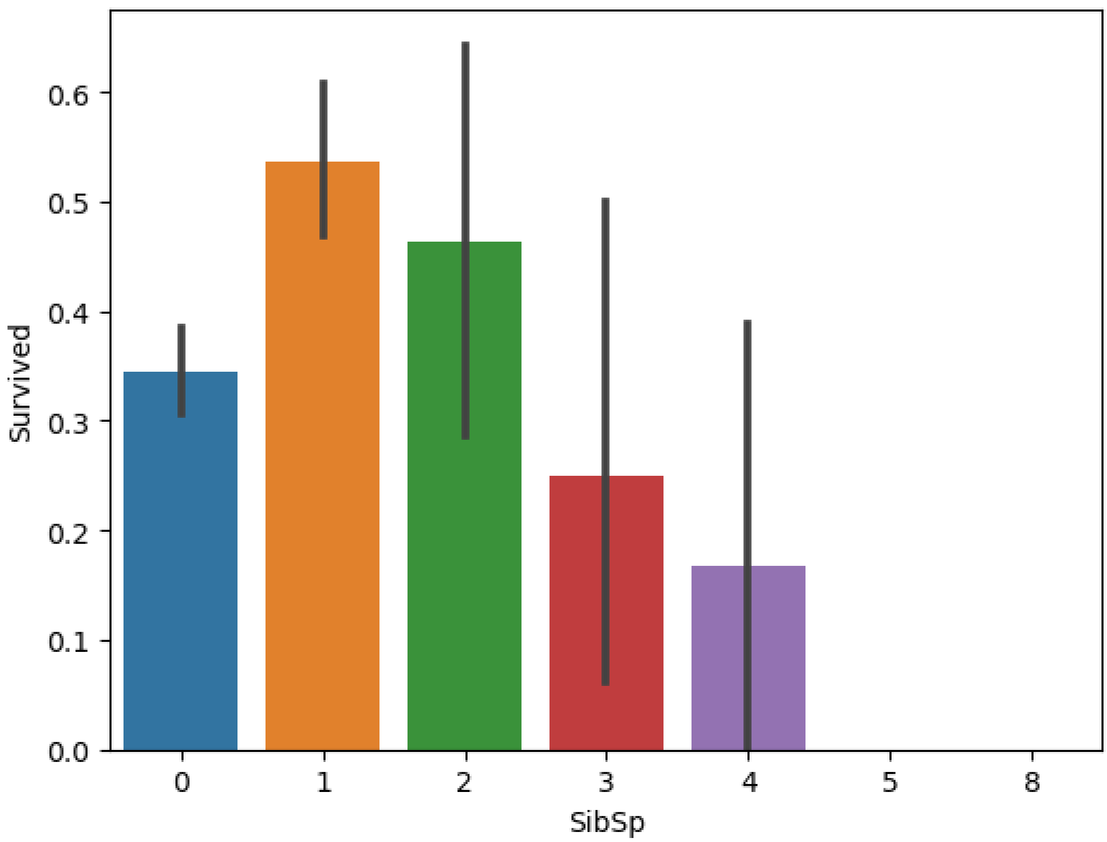
In [28]:
```
df[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean()
```

Out[28]:

	SibSp	Survived
0	0	0.345395
1	1	0.535885
2	2	0.464286
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

In [29]:
```
sns.barplot(x="SibSp", y="Survived", data=df)
```

Out[29]: <Axes: xlabel='SibSp', ylabel='Survived'>



Parch

Parents and children of the passenger inside the Titanic. It is a numerical variable.

In [30]:
```
df[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mean()
```

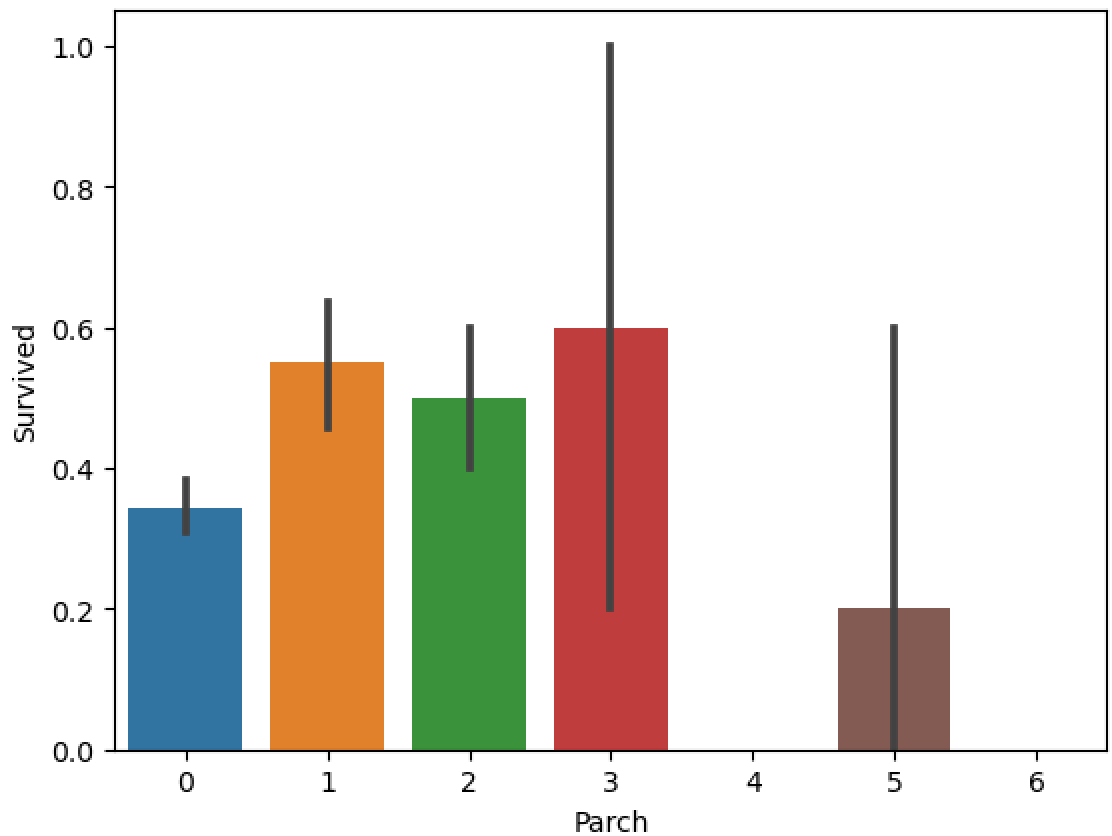
Out[30]:

	Parch	Survived
0	0	0.343658
1	1	0.550847

2	2	0.500000
3	3	0.600000
4	4	0.000000
5	5	0.200000
6	6	0.000000

In [31]:
```
sns.barplot(x="Parch", y="Survived", data=df)
```

Out[31]:
```
<Axes: xlabel='Parch', ylabel='Survived'>
```



`FamilySize`

With the variables `SibSp` and `Parch` we can calculate the size of a passenger's family + 1 member, which we will call `FamilySize`. It will therefore be a numeric variable.

In [32]:
```
df['FamilySize'] = df['SibSp'] + df['Parch'] + 1
df[["FamilySize", "Survived"]].groupby(['FamilySize'], as_index=False).mean()
```

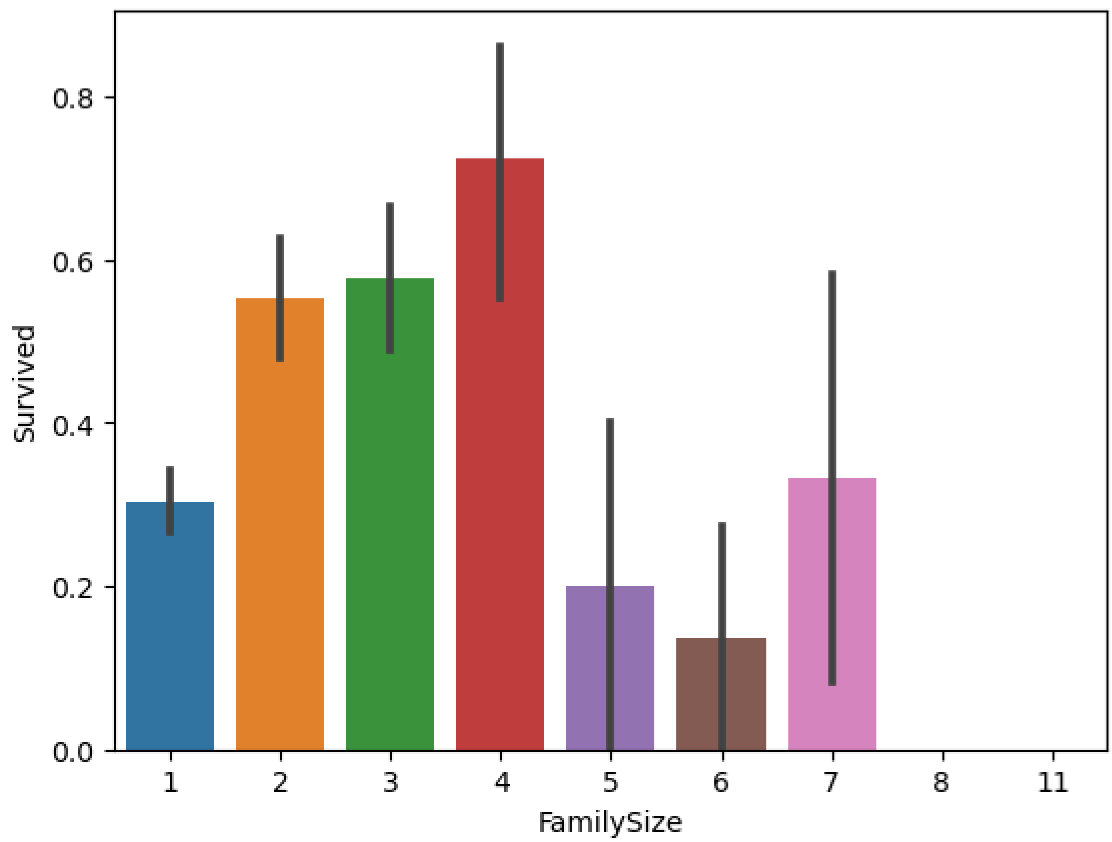
Out[32]:

	FamilySize	Survived
0	1	0.303538
1	2	0.552795
2	3	0.578431
3	4	0.724138
4	5	0.200000
5	6	0.136364
6	7	0.333333

7	8	0.000000
8	11	0.000000

In [33]:
```
sns.barplot(x="FamilySize", y="Survived", data=df)
```

Out[33]:
```
<Axes: xlabel='FamilySize', ylabel='Survived'>
```



`IsAlone`

It is interesting to characterise whether the passenger has no family inside the Titanic. Assuming that the one-member families on the titanic are in fact one person alone. We will calculate the binary variable `IsAlone`.

In [34]:
```
df['IsAlone'] = 0
df.loc[df['FamilySize'] == 1, 'IsAlone'] = 1
df[['IsAlone', 'Survived']].groupby(['IsAlone'], as_index=False).mean()
```

Out[34]:

	IsAlone	Survived
0	0	0.505650
1	1	0.303538

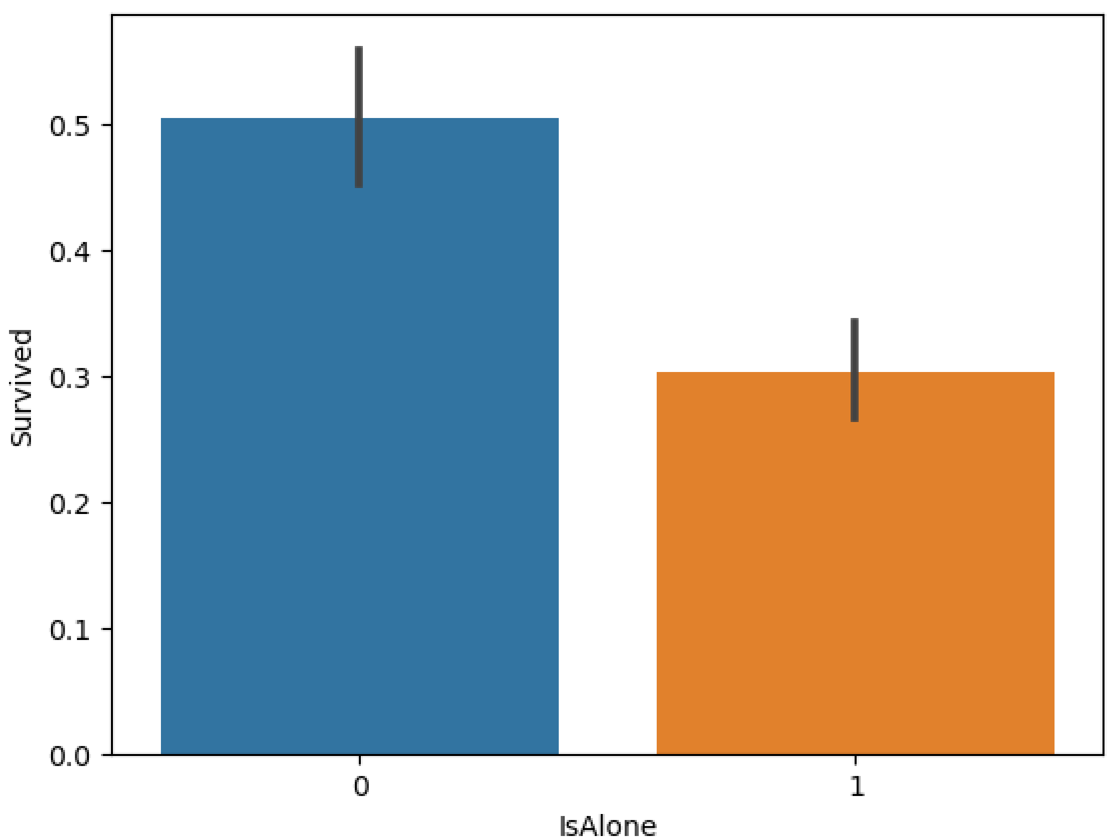
In [35]:
```
print(df)
```

```
     Survived  Pclass                                               Name  \
0           0       3                            Braund, Mr. Owen Harris
1           1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...
2           1       3                             Heikkinen, Miss. Laina
3           1       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)
4           0       3                           Allen, Mr. William Henry
..        ...     ...                                                ...
```

```
886         0       2                                      Montvila, Rev. Juozas
887         1       1                               Graham, Miss. Margaret Edith
888         0       3                   Johnston, Miss. Catherine Helen "Carrie"
889         1       1                                      Behr, Mr. Karl Howell
890         0       3                                        Dooley, Mr. Patrick

       Sex   Age  SibSp  Parch            Ticket     Fare Cabin Embarked  \
0     True  22.0      1      0         A/5 21171   7.2500   NaN        S
1    False  38.0      1      0          PC 17599  71.2833   C85        C
2    False  26.0      0      0  STON/O2. 3101282   7.9250   NaN        S
3    False  35.0      1      0            113803  53.1000  C123        S
4     True  35.0      0      0            373450   8.0500   NaN        S
..     ...   ...    ...    ...               ...      ...   ...      ...
886   True  27.0      0      0            211536  13.0000   NaN        S
887  False  19.0      0      0            112053  30.0000   B42        S
888  False   NaN      1      2        W./C. 6607  23.4500   NaN        S
889   True  26.0      0      0            111369  30.0000  C148        C
890   True  32.0      0      0            370376   7.7500   NaN        Q

     FamilySize  IsAlone
0             2        0
1             2        0
2             1        1
3             2        0
4             1        1
..          ...      ...
886           1        1
887           1        1
888           4        0
889           1        1
890           1        1

[891 rows x 13 columns]
```

In [36]:
```python
sns.barplot(x="IsAlone", y="Survived", data=df)
```

Out[36]:
```
<Axes: xlabel='IsAlone', ylabel='Survived'>
```



`Ticket`

Passenger ticket number. It is alphanumeric and a priori does not offer information that could help us in the predictive model. We will not include it.

In [37]:
```
df.drop(['Ticket'], axis=1, inplace=True)
```

`Embarked`

Indicates the passenger's port of embarkation. It is a categorical variable where `C` indicates embarkation at Cherbourg, `Q` at Queenstown and `S` at Southampton. We will complete the infinite values by imputing the port of departure as Southampton.

In [38]:
```
df['Embarked'] = df['Embarked'].fillna('S')
df[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean()
```

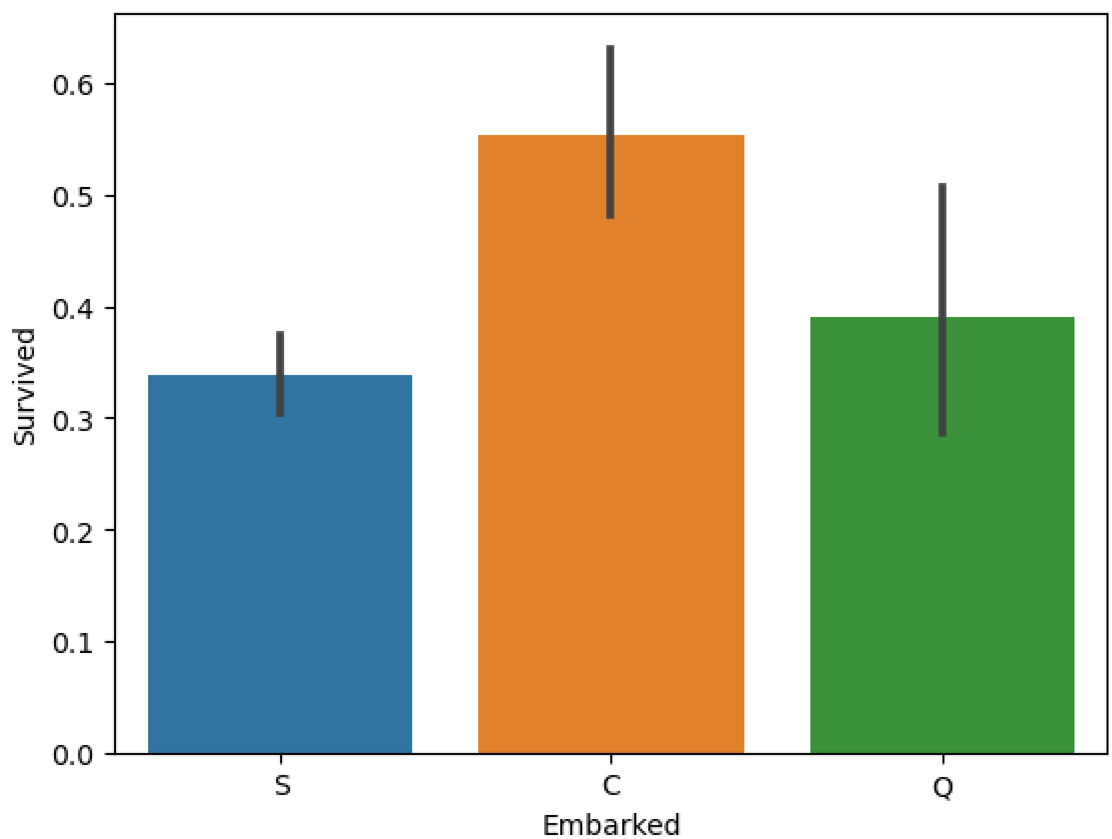
Out[38]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

In [39]:
```
sns.barplot(x="Embarked", y="Survived", data=df)
```

Out[39]:

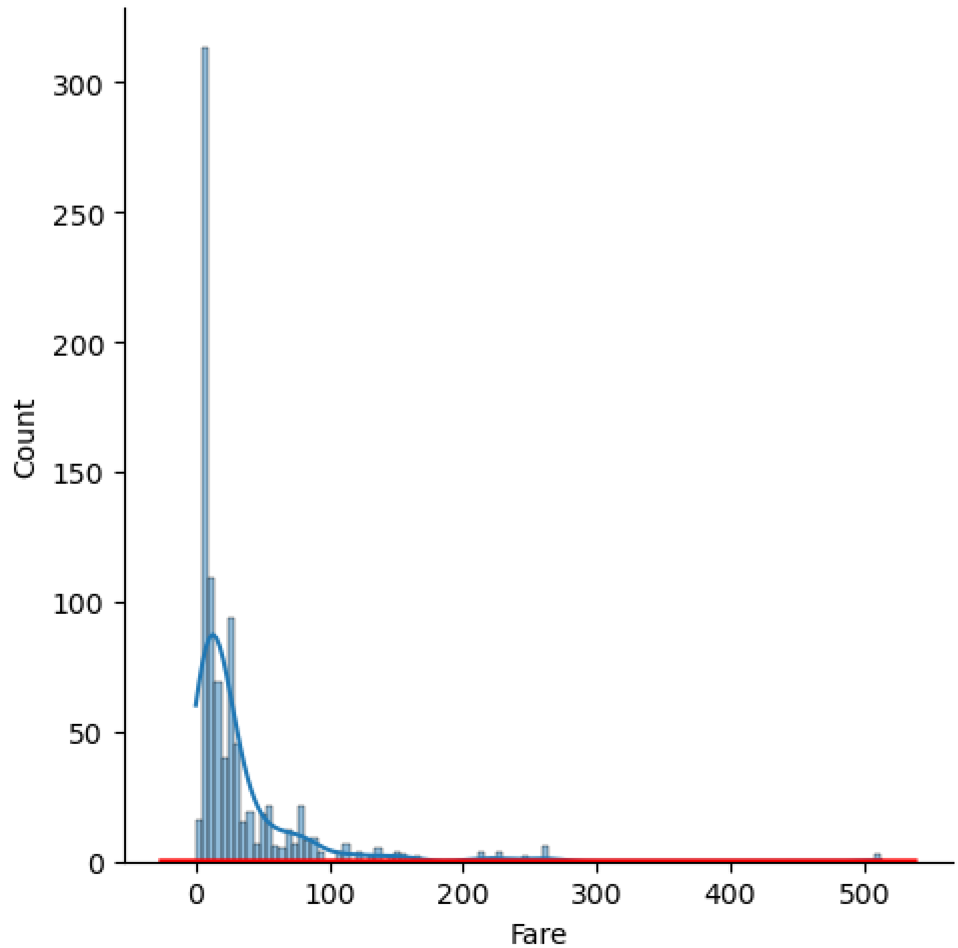
```
<Axes: xlabel='Embarked', ylabel='Survived'>
```



`Fare`

It indicates the price of the ticket. It is therefore a continuous numerical variable.

In [52]:
```
from scipy.stats import norm

# Ploting the distribution of 'Fare'
sns.displot(df['Fare'], kde=True)

# Fiting a normal distribution to the data
mu, std = norm.fit(df['Fare'])
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'r', linewidth=2)

# Show the plot
plt.show()
```



We transform the variable into a categorical variable.In other words, we are going to discretise them

In [54]:
```
df['FareGroup'] = pd.qcut(df['Fare'], 7, labels=['A', 'B', 'C', 'D', 'E', 'F', 'G'])
df[['FareGroup', 'Survived']].groupby(['FareGroup'], as_index=False).mean()
```

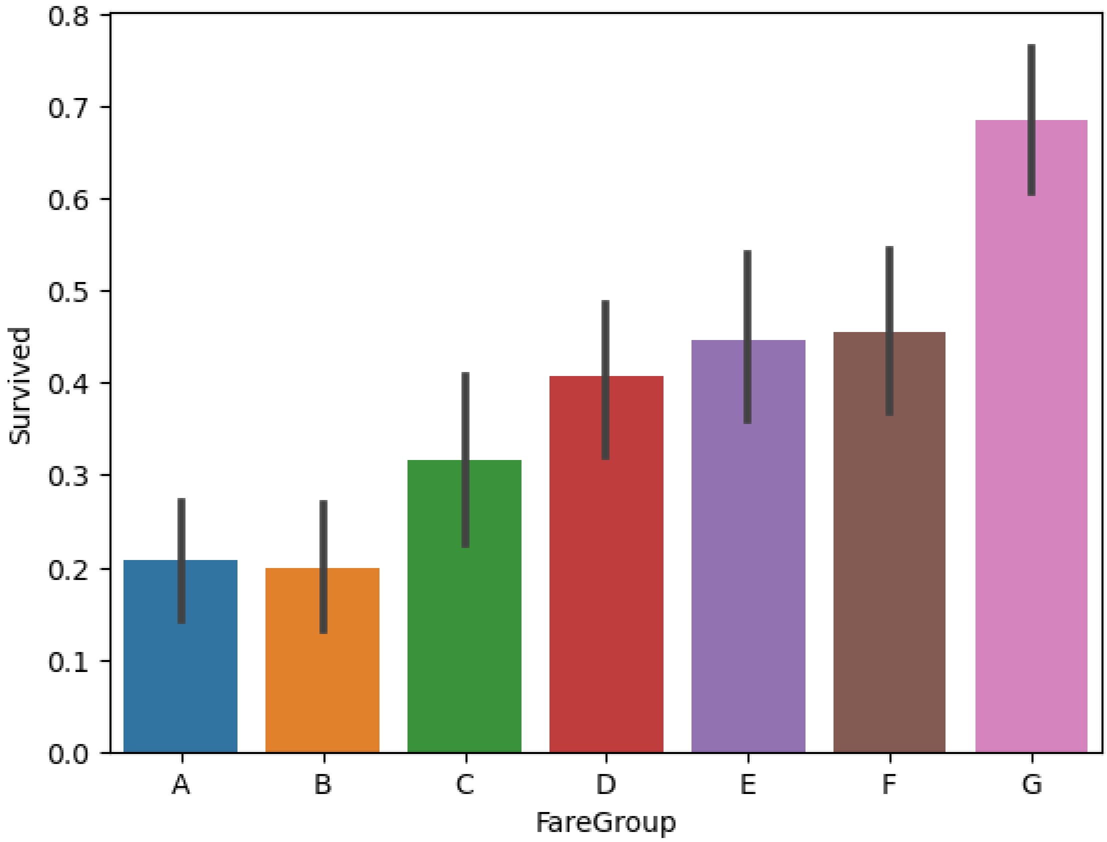
Out[54]:

	FareGroup	Survived
0	A	0.207143
1	B	0.200000
2	C	0.316327
3	D	0.406250
4	E	0.445312
5	F	0.456000

6	G	0.685039

In [55]:
```
sns.barplot(x="FareGroup", y="Survived", data=df)
```

Out[55]:
```
<Axes: xlabel='FareGroup', ylabel='Survived'>
```



We eliminate the variable we do not need

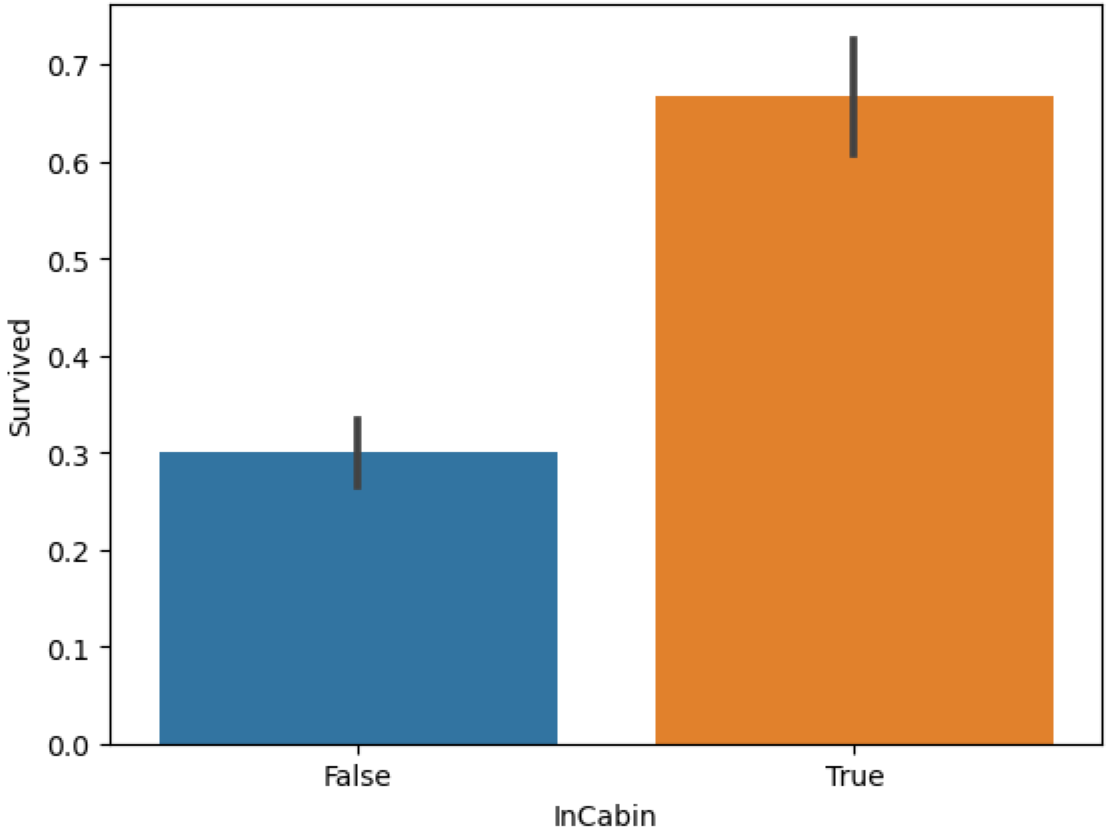
In [56]:
```
df.drop(['Fare'], axis=1, inplace=True)
```

`Cabin`

Passenger cabin number. This is a text variable, indicating which cabin the passenger was in. We will transform the variable into a binary variable depending on whether the passenger was in a cabin or not.

In [57]:
```
df['InCabin'] = ~df['Cabin'].isnull()
```

In [58]:
```
sns.barplot(x="InCabin", y="Survived", data=df)
plt.show()
```



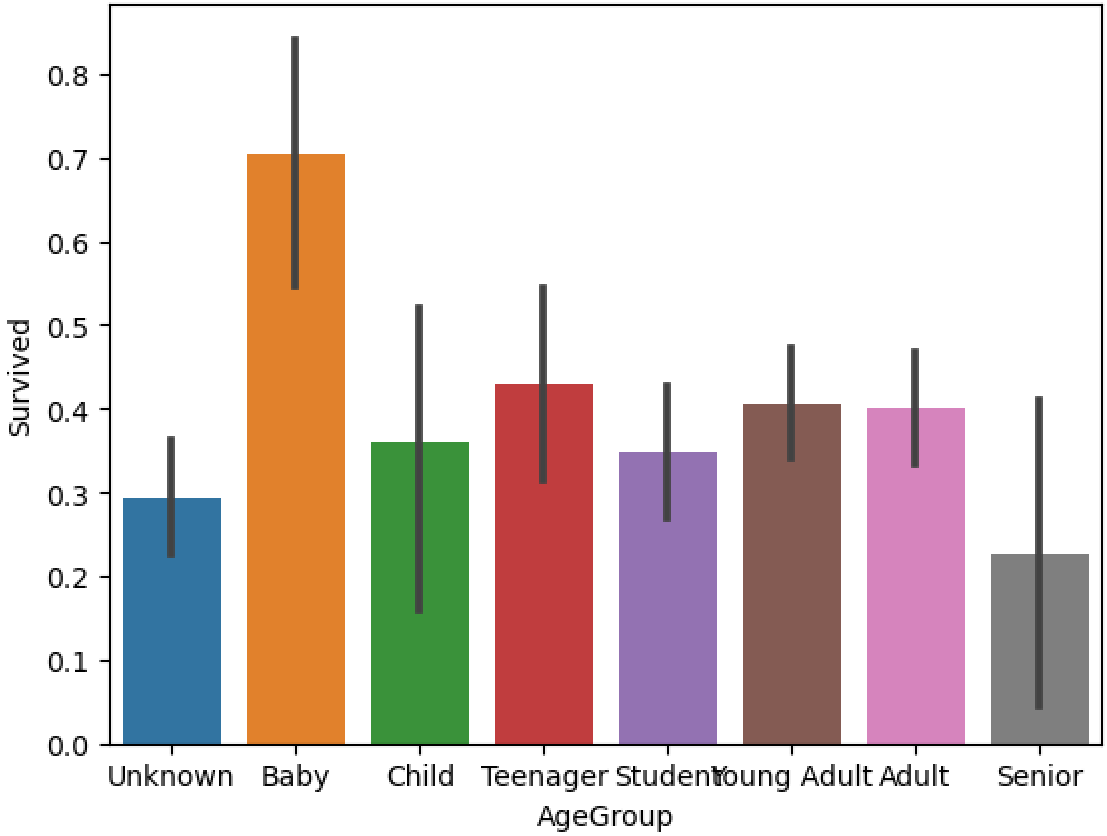
In [59]:
```
df.drop(['Cabin'], axis=1, inplace=True)
```

Age

Edad del pasajero. Es una variable numerica que tiene bastantes valores vacios. Categorizaremos la variable en 8 categorías según la edad que tenga. La categoría `Unknown` será la de aquellos pasajeros que no tengamos la edad.

In [60]:
```
df["Age"] = df["Age"].fillna(-0.5)
bins = [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Child', 'Teenager', 'Student', 'Young Adult', 'Adult', 'Se
df['AgeGroup'] = pd.cut(df["Age"], bins, labels = labels)
```

In [61]:
```
sns.barplot(x="AgeGroup", y="Survived", data=df)
plt.show()
```



In [62]:
```
df.drop(['Age'], axis=1, inplace=True)
```

`Name`.

Name of the passenger. From this variable we can extract the titles included with the name, such as `Mr`, `Miss` or `Master`, which can have a predictive value. The rest of the name, we will ignore.

In [64]:
```
df['Name'].head(10)
```

Out[64]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

In [65]:
```
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    if title_search:
        return title_search.group(1)
    return ""

df['Title'] = df['Name'].apply(get_title)

pd.crosstab(df['Title'], df['Sex'])
```

Out[65]:

Sex	False	True
Title		

Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

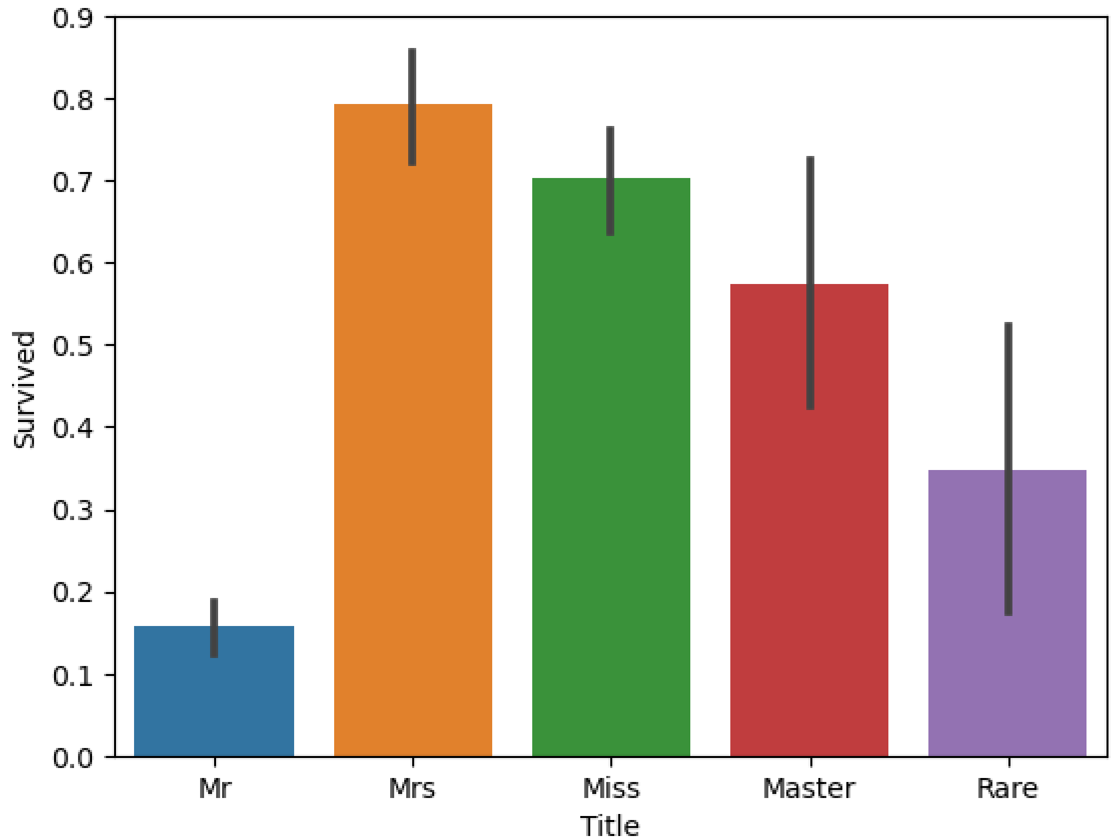
Categorizaremos aquellos títulos que sean mas usados, y el resto los agruparemos a una categoría `Rare`.

In [66]:
```
df['Title'] = df['Title'].replace(['Lady', 'Countess','Capt', 'Col', 'Don', 'Dr', 'Major
df['Title'] = df['Title'].replace('Mlle', 'Miss')
df['Title'] = df['Title'].replace('Ms', 'Miss')
df['Title'] = df['Title'].replace('Mme', 'Mrs')

df[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()
```

Out[66]:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Rare	0.347826

In [67]:
```
sns.barplot(x="Title", y="Survived", data=df)
plt.show()
```

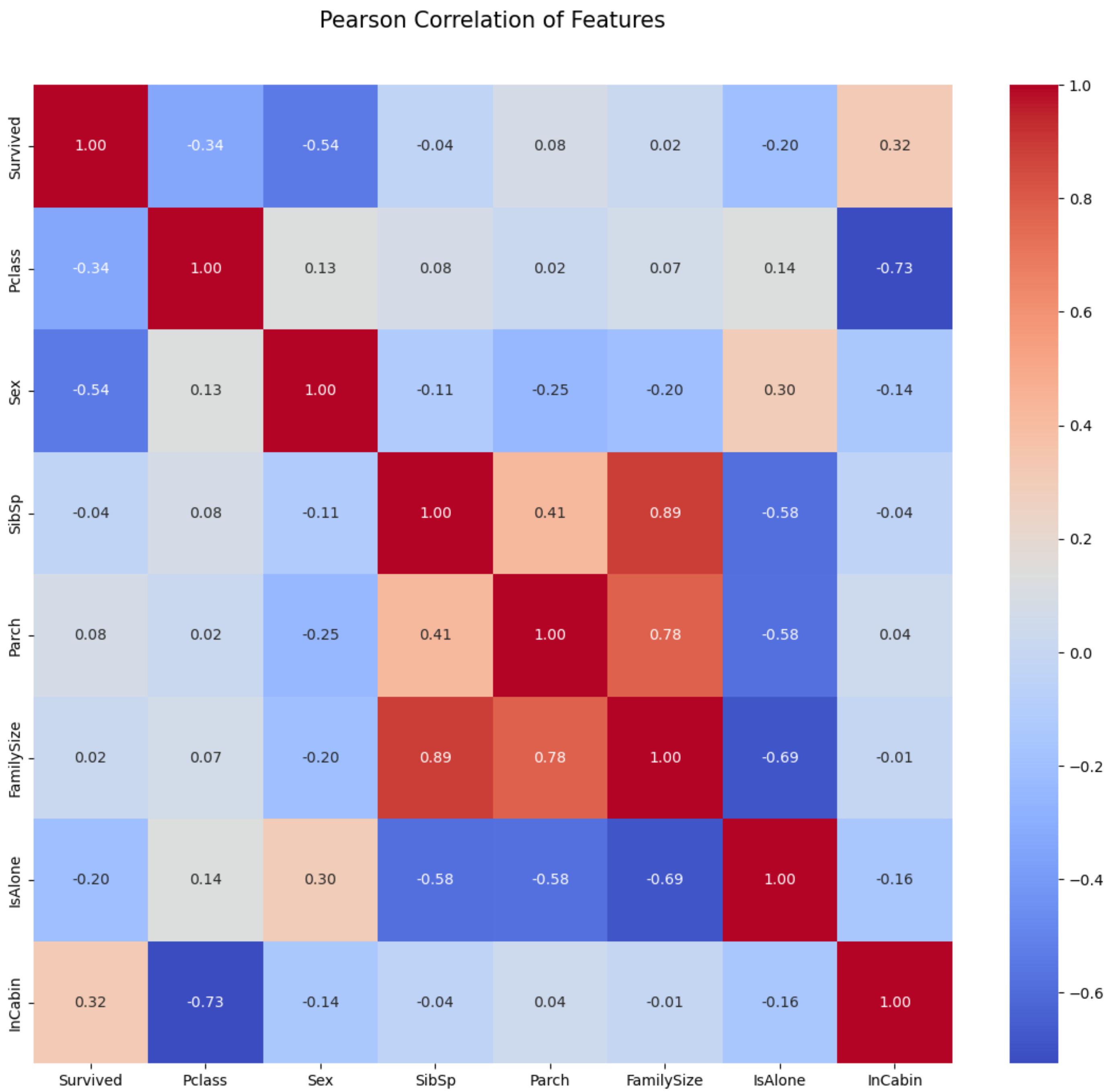


In [68]:
```
df.drop(['Name'], axis=1, inplace=True)
```

Correlation of the variables

If we visualise the correlation matrix between the variables, we see that the most correlated variables with the one we want to predict are `Sex`, `Pclass` and `isAlone`.

In [70]:
```
correlation_matrix = df.corr(numeric_only=True)
correlation_matrix

plt.figure(figsize=(14, 12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.show()
```



Binarization of categorical variables

For each categorical variable we will make `m` binary variables, where `m` is the number of categories of the variable.

In [71]:

```
cols = ['Pclass', 'Embarked', 'FareGroup', 'AgeGroup', 'Title']
titanic_categorical = df[cols]
titanic_categorical = pd.concat([pd.get_dummies(titanic_categorical[col], prefix=col) fo
titanic_categorical.head()
df = pd.concat([df[df.columns[~df.columns.isin(cols)]], titanic_categorical], axis=1)
df.head()
```

Out[71]:

	Survived	Sex	SibSp	Parch	FamilySize	IsAlone	InCabin	Pclass_1	Pclass_2	Pclass_3	...	AgeGroup_Teenag
0	0	True	1	0	2	0	False	0	0	1	...	
1	1	False	1	0	2	0	True	1	0	0	...	
2	1	False	0	0	1	1	False	0	0	1	...	

3	1	False	1	0	2	0	True	1	0	0	...
4	0	True	0	0	1	1	False	0	0	1	...

5 rows × 33 columns

Training and test sets

We divide the data into two sets, training and test. With the training set we will create the predictive model, and with the test set, we will evaluate it to see how it performs.

In [74]:
```
from sklearn.model_selection import train_test_split
X = df.drop(labels='Survived', axis=1)
y = df.Survived

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=4
```

Predictive Models

Now that we have the dataset prepared for analysis, we will create different prediction models and compare them. The models we will create are as follows:

- Logistic Regression
- Decision Trees
- Random Forests
- k-nearest neighbors
- Support Vector Machines We will evaluate them using the area under the ROC curve metric for each prediction model concerning the outcomes. We will then select the model that yields the best results.

In [75]:
```
performance_auc = {} #Empty Dictionary
```

In [77]:
```
%who
```

```
X          X_test   X_train          bins     cols     correlation_matrix       df       get_tit
le         labels
mu         norm     np       p       pd       performance_auc          plt      re       sns

std        titanic_categorical      train_test_split         x        xmax     xmin     y
y_test     y_train
```

Logistic regression

In [85]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression().fit(X_train, y_train)

# Get the coefficients and intercept
coefficients = model.coef_
intercept = model.intercept_

print("Coefficients:", coefficients)
print("Intercept:", intercept)
```

```
Coefficients: [[-1.35093421e+00 -2.30998129e-01 -9.61107751e-02 -3.26199013e-01
```

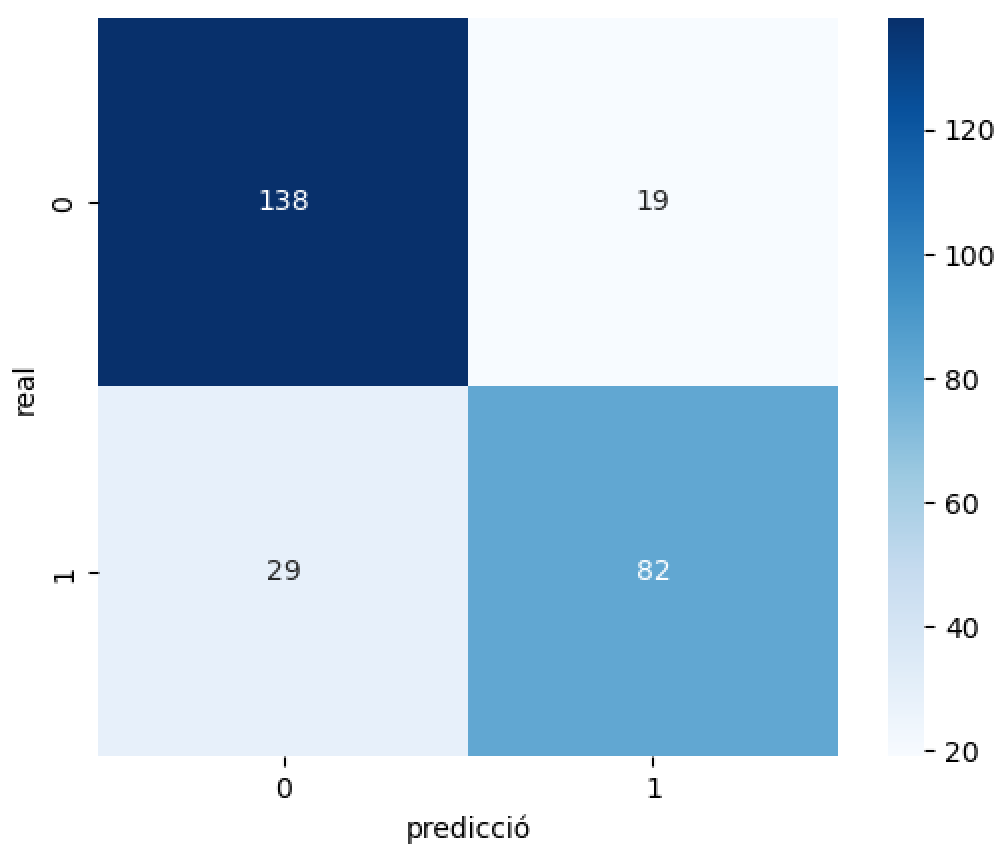
```
  -3.21498085e-01  1.01739670e+00  1.44435174e-01  5.21441605e-01
  -6.64966888e-01  3.14281953e-01  1.92576772e-01 -5.05948834e-01
  -1.26718215e-01  3.62192784e-01 -5.36835619e-01 -1.29147416e-01
   6.28042101e-04  1.90343256e-02  4.11755990e-01 -3.50507359e-01
   1.12805866e+00 -2.53885004e-02 -3.74621675e-01  7.96068642e-02
   1.11450427e-01 -4.46954751e-01 -1.20733770e-01  1.43144305e+00
   4.70447016e-02 -1.34614864e+00  9.15582612e-01 -1.04701183e+00]]
Intercept: [2.12903885]
```

In [80]:
```python
predicted = model.predict(X_test)
predicted
```

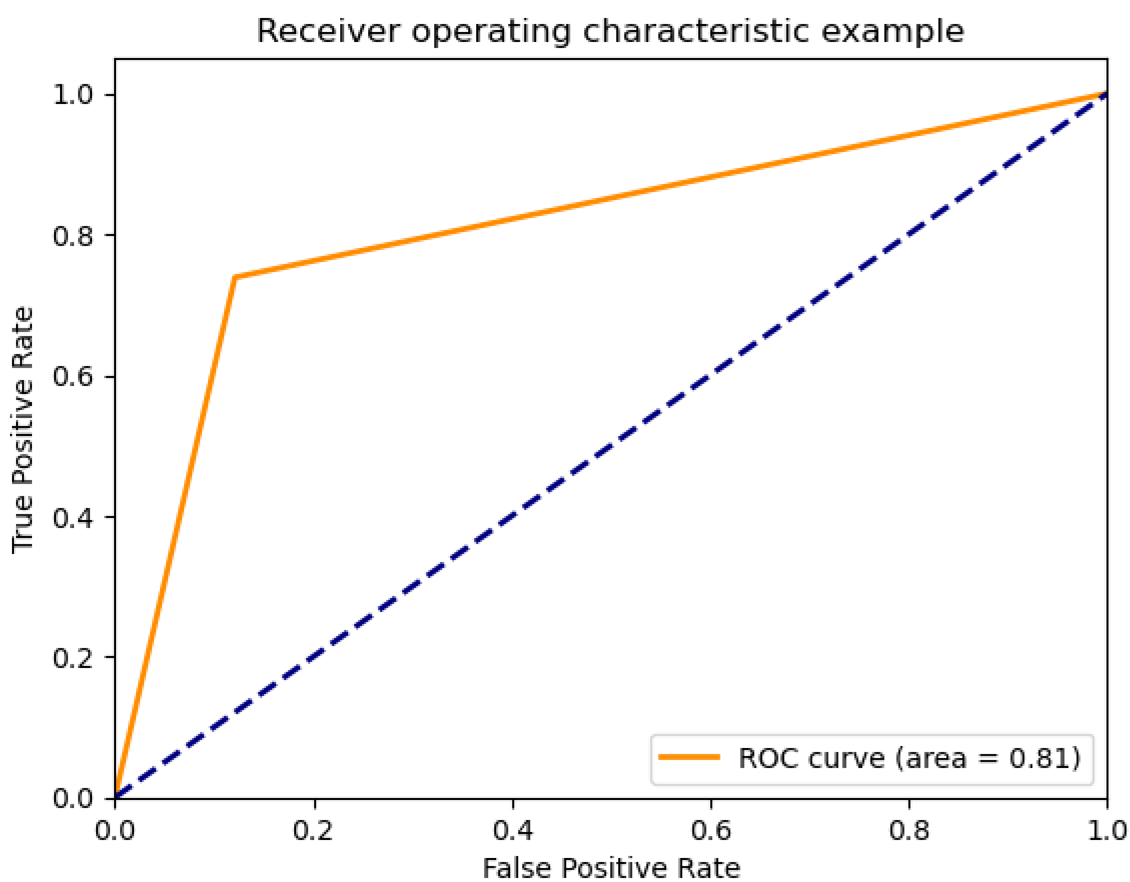
Out[80]:
```
array([1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0], dtype=int64)
```

In [81]:
```python
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, predicted)
sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', square=True)
plt.xlabel("predicció")
plt.ylabel("real")
plt
```

Out[81]:
```
<module 'matplotlib.pyplot' from 'C:\\Users\\Panasonic\\anaconda3\\lib\\site-packages\\m
atplotlib\\pyplot.py'>
```



In [82]:
```python
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, predicted)
roc_auc = auc(fpr, tpr)
performance_auc['Logistic Regression'] = roc_auc

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Decision trees

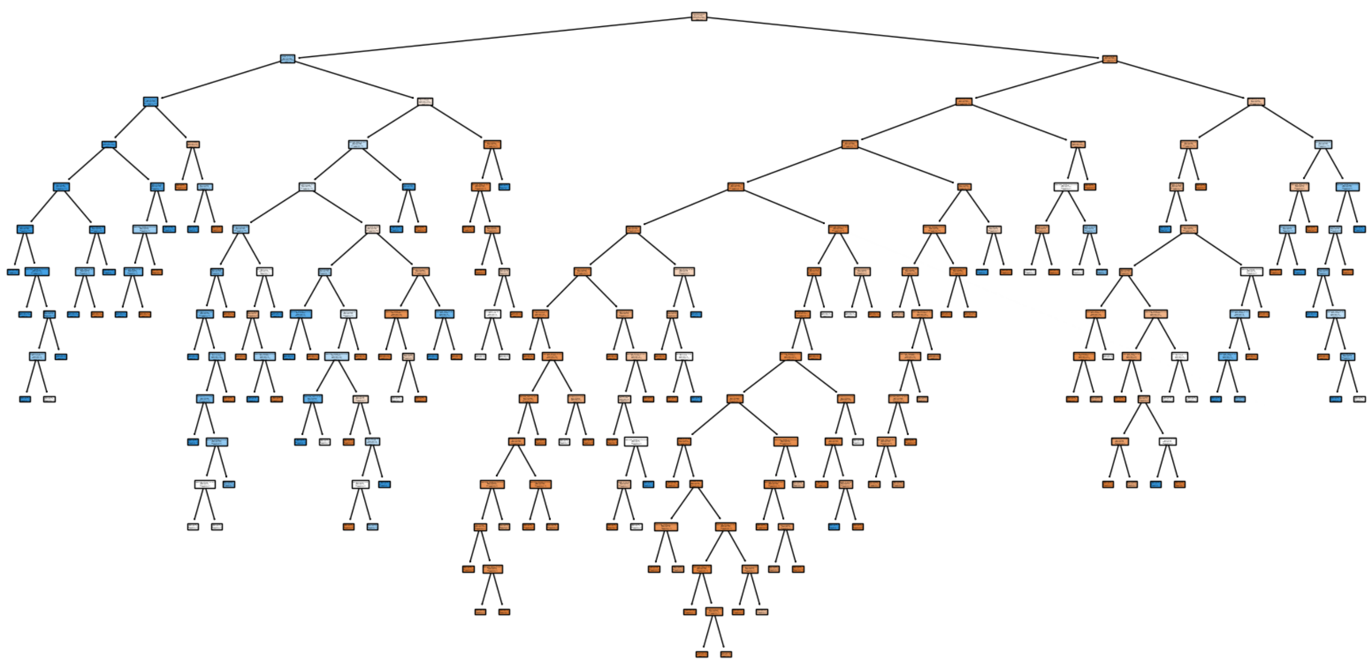
In [86]:
```python
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier().fit(X_train, y_train)

# Get feature importances
feature_importances = model.feature_importances_

# Get the number of nodes in the tree
num_nodes = model.tree_.node_count

# Print the feature importances and number of nodes
print("Feature Importances:", feature_importances)
print("Number of Nodes in the Tree:", num_nodes)
```

```
Feature Importances: [0.00000000e+00 1.65102621e-02 1.42790931e-02 1.05916813e-01
```

```
 6.02407922e-03 4.44315676e-02 1.43650387e-02 5.96850732e-03
 1.16344312e-01 1.41462060e-02 1.84415059e-02 1.34725036e-02
 2.54672481e-02 8.97146481e-03 1.49435277e-02 6.71661556e-03
 9.37467780e-03 9.97837229e-03 1.00094869e-02 2.51572864e-02
 2.15985571e-02 0.00000000e+00 2.66646417e-04 7.73671585e-03
 2.95822489e-02 2.89778266e-02 9.57777049e-03 0.00000000e+00
 1.37694543e-02 3.63960512e-01 0.00000000e+00 4.40117003e-02]
Number of Nodes in the Tree: 255
```

In [87]:

```
from sklearn.tree import plot_tree

# Visualizing the decision tree
plt.figure(figsize=(20, 10))
plot_tree(model, filled=True, feature_names=X_train.columns, class_names=['0', '1'])
plt.show()
```

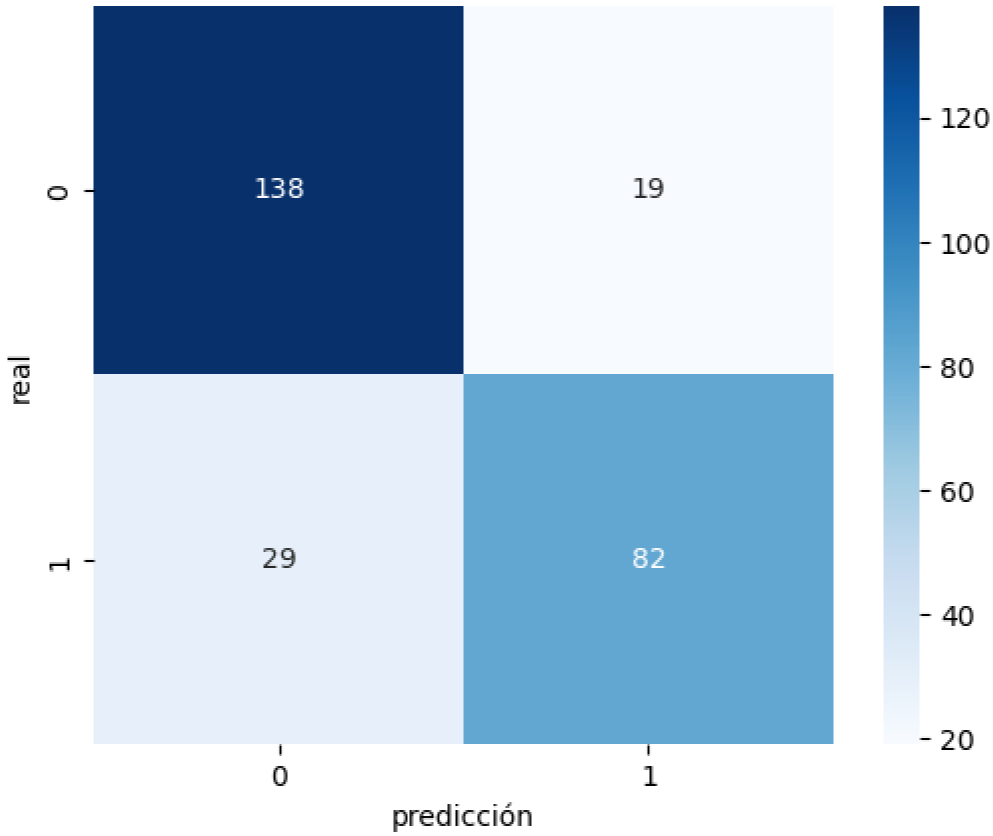


In [89]:

```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, predicted)
sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', square=True)
plt.xlabel("predicción")
plt.ylabel("real")
plt
```

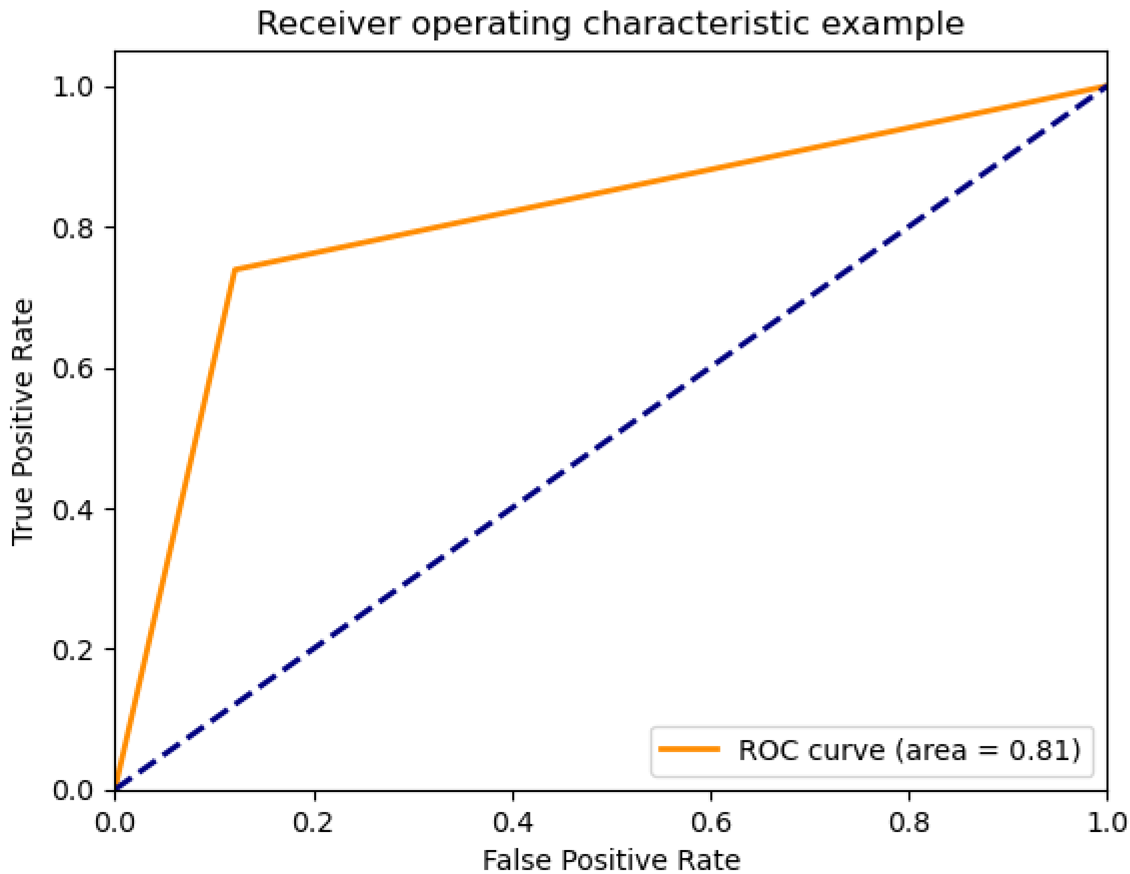
Out[89]:

```
<module 'matplotlib.pyplot' from 'C:\\Users\\Panasonic\\anaconda3\\lib\\site-packages\\m
atplotlib\\pyplot.py'>
```



In [90]:

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, predicted)
roc_auc = auc(fpr, tpr)
performance_auc['Decision Tree'] = roc_auc

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



In [91]:
```
pd.concat((pd.DataFrame(X_train.iloc[:, 1:].columns, columns = ['variable']),
           pd.DataFrame(model.feature_importances_, columns = ['importance'])),
          axis = 1).sort_values(by='importance', ascending = False)[:20]
```

Out[91]:

	variable	importance
29	Title_Mrs	0.363961
8	Embarked_C	0.116344
3	IsAlone	0.105917
5	Pclass_1	0.044432
31	NaN	0.044012
24	AgeGroup_Adult	0.029582
25	AgeGroup_Senior	0.028978
12	FareGroup_B	0.025467
19	AgeGroup_Baby	0.025157
20	AgeGroup_Child	0.021599
10	Embarked_S	0.018442
1	Parch	0.016510
14	FareGroup_D	0.014944
6	Pclass_2	0.014365
2	FamilySize	0.014279
9	Embarked_Q	0.014146
28	Title_Mr	0.013769

11	FareGroup_A	0.013473
18	AgeGroup_Unknown	0.010009
17	FareGroup_G	0.009978

Random Forest

In [99]:
```
from sklearn.ensemble import RandomForestClassifier


model = RandomForestClassifier(n_estimators=1000).fit(X_train, y_train)

# Get the feature importances
feature_importances = model.feature_importances_

# Predict on new data (X_test)
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = model.score(X_test, y_test)  # Accuracy on the test set
# We can use other evaluation metrics, e.g., confusion matrix, ROC AUC, etc.

# Print the feature importances and accuracy
print("Feature Importances:", feature_importances)
print("Model Accuracy:", accuracy)
```

```
Feature Importances: [0.12797438 0.04080987 0.02958903 0.05981278 0.01510198 0.04983048
 0.02114992 0.02429846 0.05575202 0.02335158 0.01358702 0.02415359
 0.02226135 0.01879795 0.01681573 0.01768928 0.01482299 0.02648228
 0.01849042 0.02464705 0.02285737 0.00539538 0.01324031 0.01757225
 0.02831117 0.02300574 0.00755807 0.01204701 0.04093541 0.12958053
 0.04491414 0.00916446]
Model Accuracy: 0.8097014925373134
```

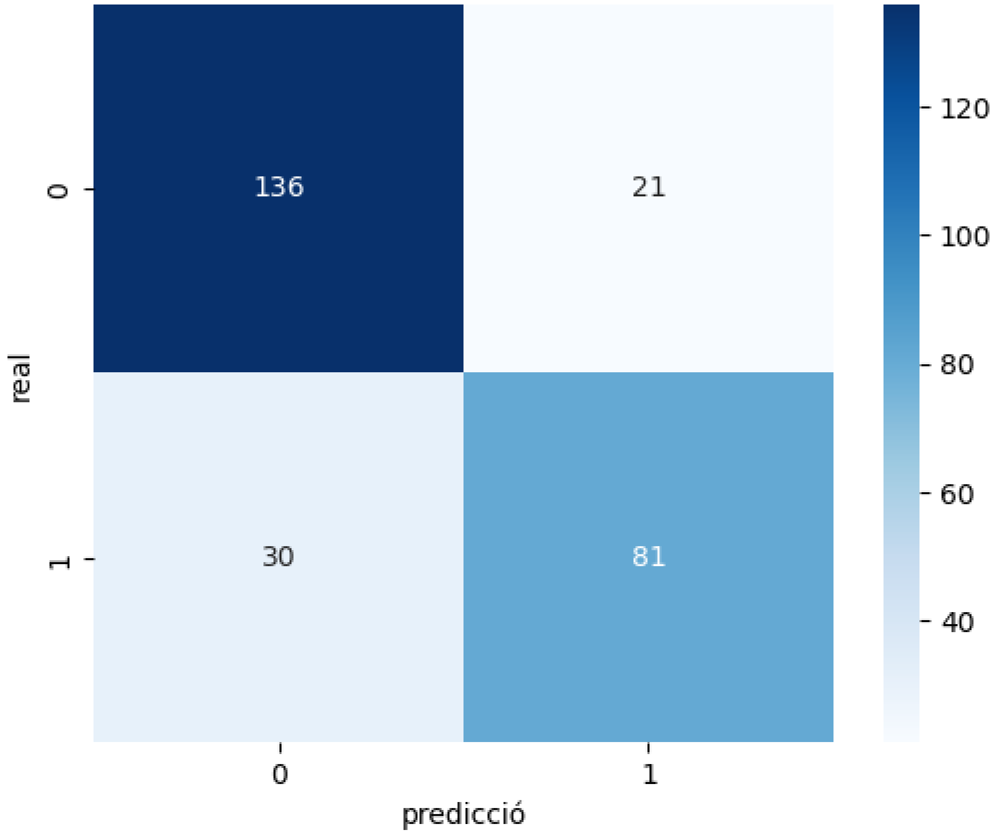
In [94]:
```
predicted = model.predict(X_test)
predicted
```

Out[94]:
```
array([1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 1], dtype=int64)
```

In [95]:
```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, predicted)
sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', square=True)
plt.xlabel("predicción")
plt.ylabel("real")
plt
```

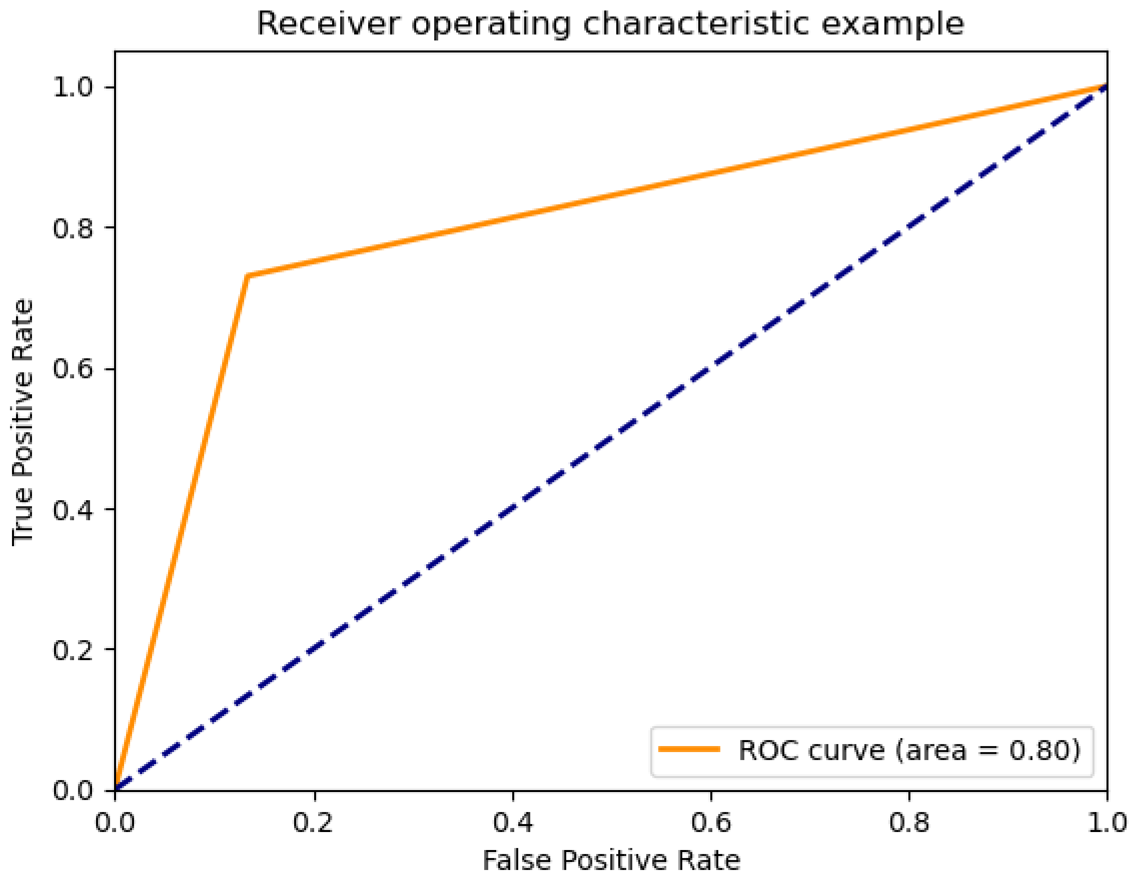
Out[95]:
```
<module 'matplotlib.pyplot' from 'C:\\Users\\Panasonic\\anaconda3\\lib\\site-packages\\m
```

```
atplotlib\\pyplot.py'>
```



In [96]:

```python
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, predicted)
roc_auc = auc(fpr, tpr)
performance_auc['Random Forests'] = roc_auc

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



In [97]:
```
pd.concat((pd.DataFrame(X_train.iloc[:, 1:].columns, columns = ['variable']),
           pd.DataFrame(model.feature_importances_, columns = ['importance'])),
          axis = 1).sort_values(by='importance', ascending = False)[:20]
```

Out[97]:

	variable	importance
29	Title_Mrs	0.130578
0	SibSp	0.128849
3	IsAlone	0.061739
8	Embarked_C	0.053645
5	Pclass_1	0.049275
30	Title_Rare	0.043787
28	Title_Mr	0.041868
1	Parch	0.038882
2	FamilySize	0.029711
24	AgeGroup_Adult	0.028501
17	FareGroup_G	0.025635
7	Pclass_3	0.025290
11	FareGroup_A	0.024434
19	AgeGroup_Baby	0.024359
6	Pclass_2	0.024000
9	Embarked_Q	0.023262
25	AgeGroup_Senior	0.022734

20	AgeGroup_Child	0.022654
12	FareGroup_B	0.021511
13	FareGroup_C	0.018444

k-nearest neighbors

In [103…

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3).fit(X_train, y_train)

# Predict on new data (X_test)
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = model.score(X_test, y_test)  # Accuracy on the test set


# Print the model's accuracy
print("Model Accuracy:", accuracy)
# Displaying a detailed representation of the model
print(model)
# or 'print(repr(model))'
```

```
Model Accuracy: 0.8059701492537313
KNeighborsClassifier(n_neighbors=3)
```

In [104…

```
predicted = model.predict(X_test)
predicted
```

Out[104]:

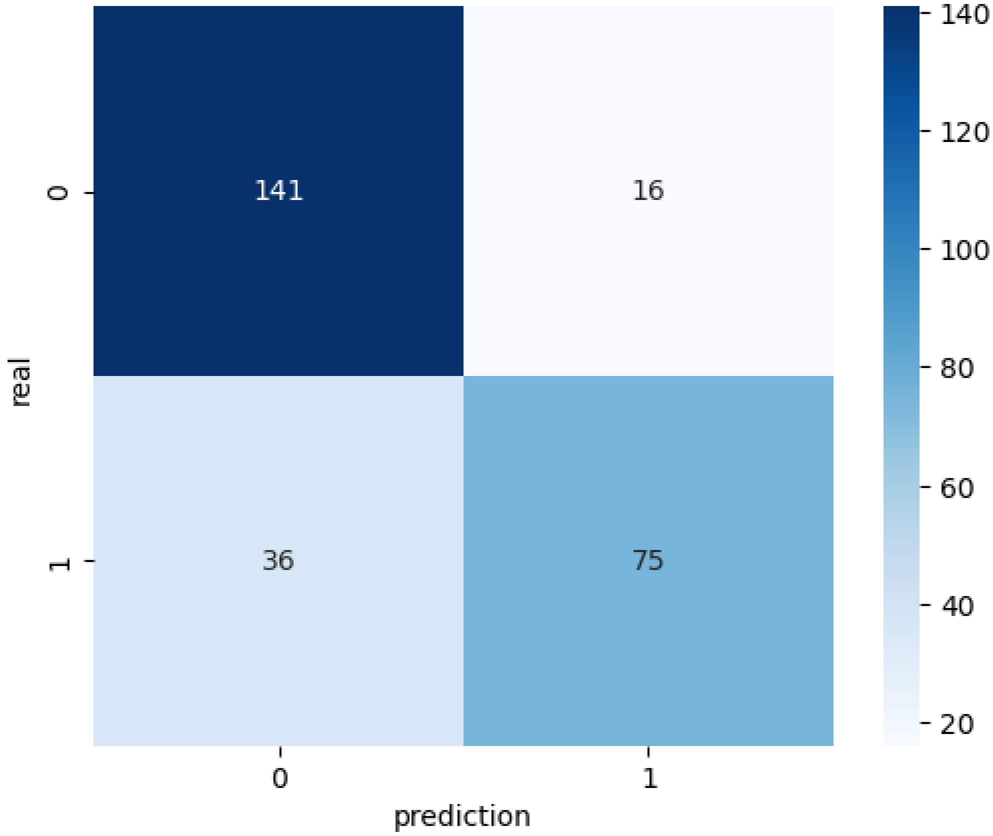
```
array([1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0], dtype=int64)
```

In [106…

```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, predicted)
sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', square=True)
plt.xlabel("prediction")
plt.ylabel("real")
plt
```

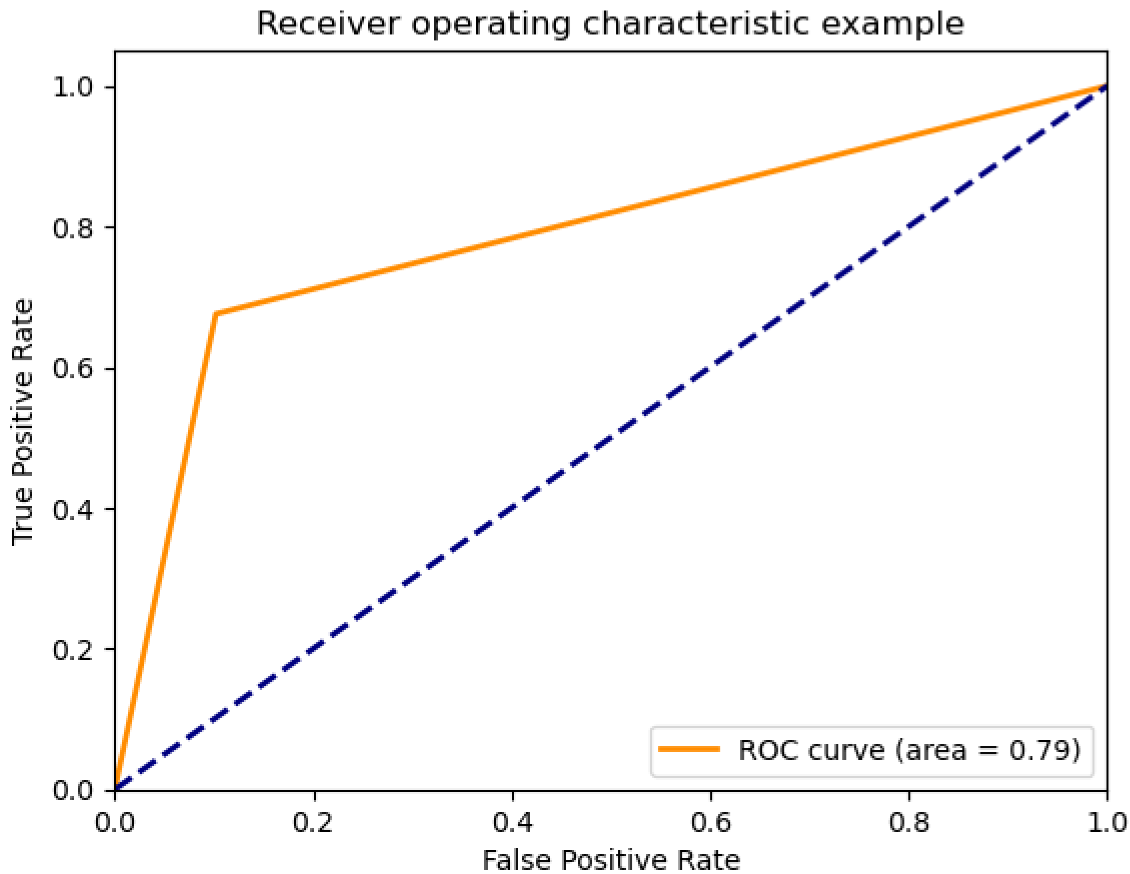
Out[106]:

```
<module 'matplotlib.pyplot' from 'C:\\Users\\Panasonic\\anaconda3\\lib\\site-packages\\m
atplotlib\\pyplot.py'>
```



In [107…

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, predicted)
roc_auc = auc(fpr, tpr)
performance_auc['k-nearest neighbours'] = roc_auc

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Support Vector Machines

In [109…

```
from sklearn.svm import SVC


model = SVC(probability=True).fit(X_train, y_train)

# Get the support vectors
support_vectors = model.support_vectors_

# Get the support indices
support_indices = model.support_

# Get the number of support vectors for each class
n_support_vectors = model.n_support_

# Predict on new data (X_test)
y_pred = model.predict(X_test)

# Evaluate the model's performance
accuracy = model.score(X_test, y_test)  # Accuracy on the test set

# Print the number of support vectors and accuracy
print("Number of Support Vectors:", n_support_vectors)
print("Model Accuracy:", accuracy)
```

```
Number of Support Vectors: [168 153]
Model Accuracy: 0.835820895522388
```

In [110…

```
predicted = model.predict(X_test)
predicted
```

Out[110]:

```
array([1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
```

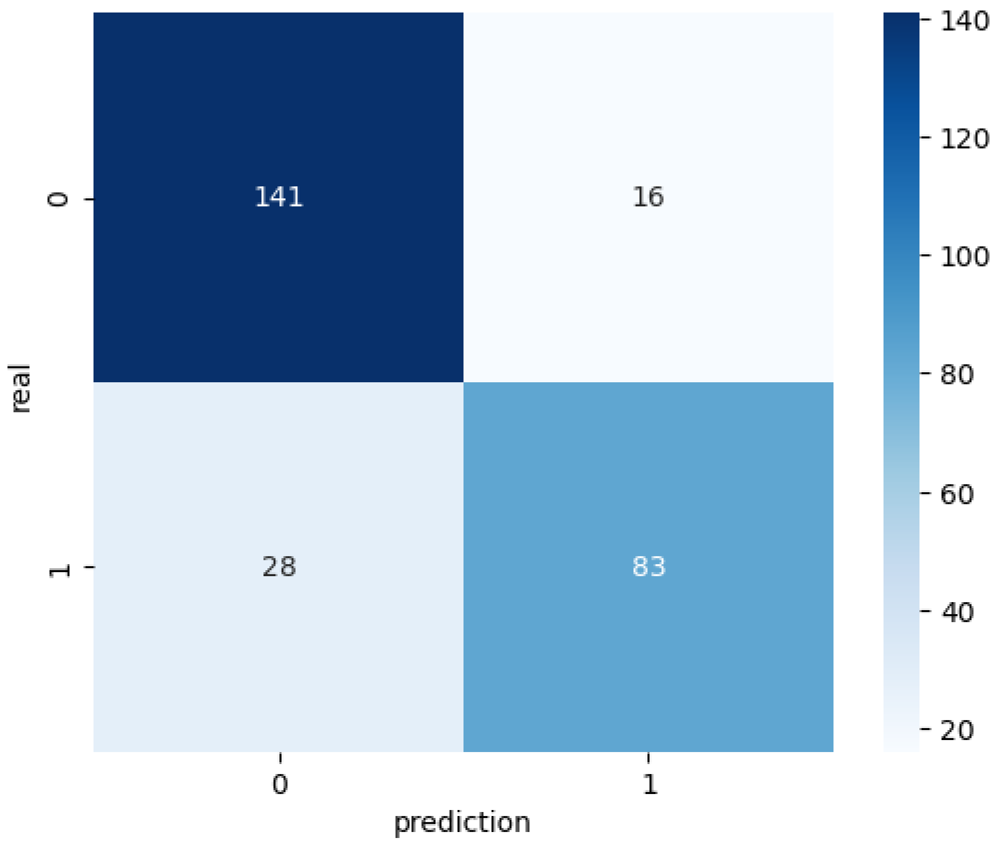
```
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0], dtype=int64)
```

In [112…

```
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, predicted)
sns.heatmap(matrix, annot=True, fmt="d", cmap='Blues', square=True)
plt.xlabel("prediction")
plt.ylabel("real")
plt
```

Out[112]:

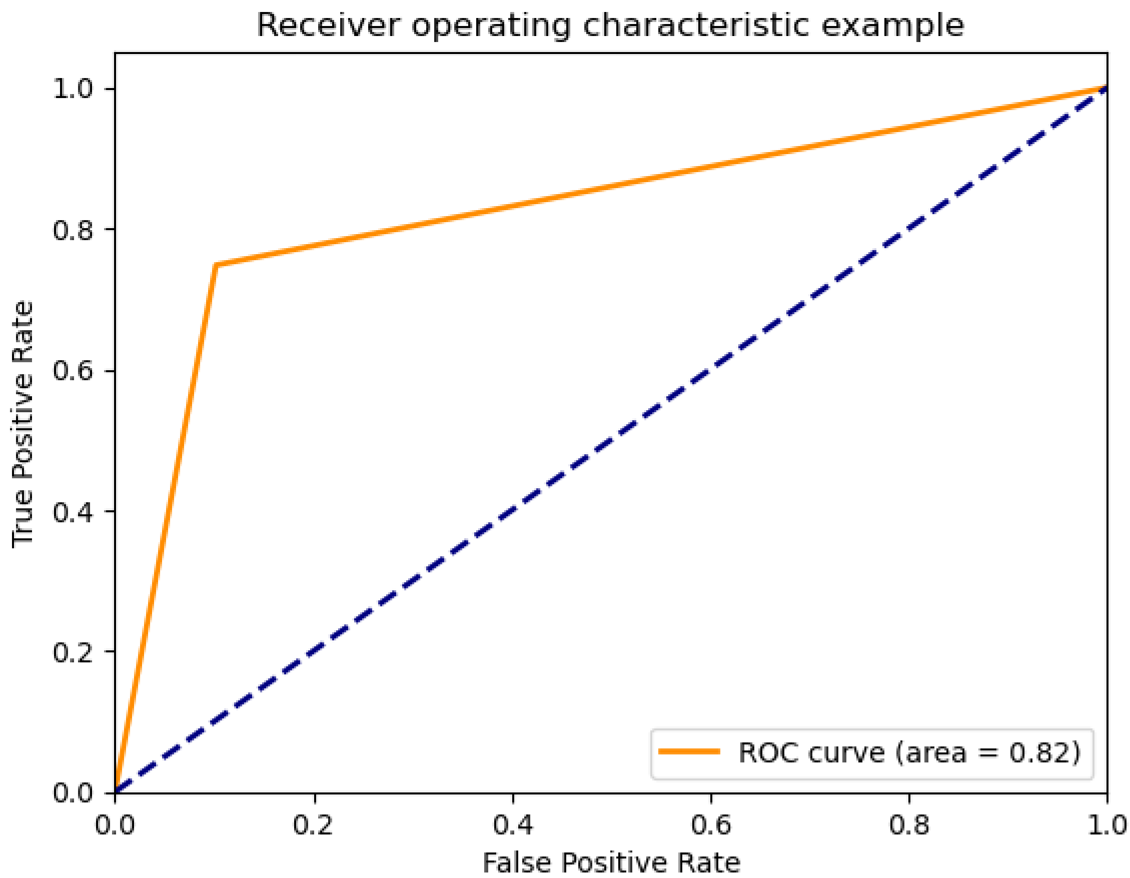
```
<module 'matplotlib.pyplot' from 'C:\\Users\\Panasonic\\anaconda3\\lib\\site-packages\\matplotlib\\pyplot.py'>
```



In [113…

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, thresholds = roc_curve(y_test, predicted)
roc_auc = auc(fpr, tpr)
performance_auc['SVM'] = roc_auc

plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc_auc
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
```

```python
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



Conclusion

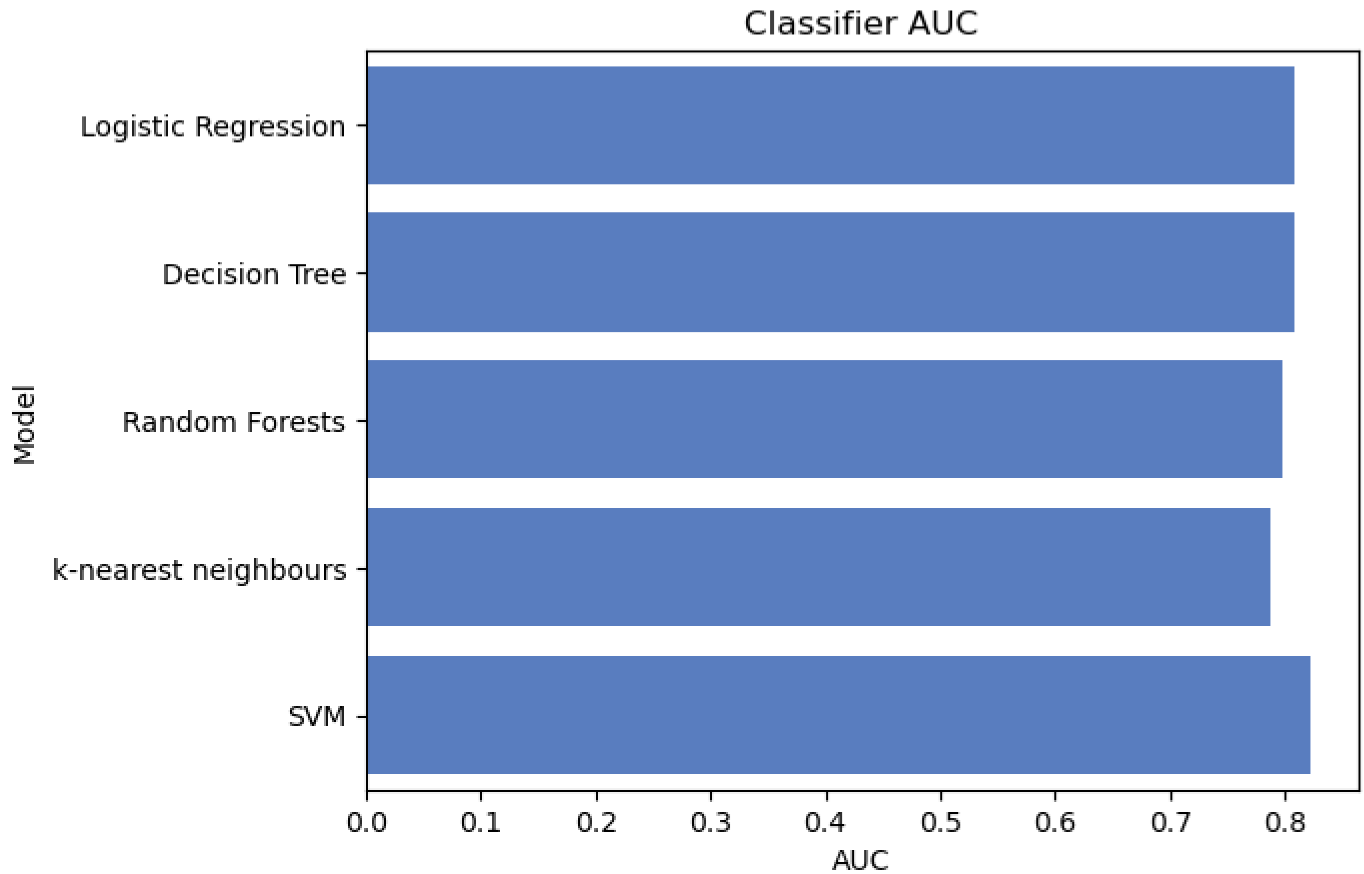
After applying different classification models, and seeing that there are not too many differences in performance between them, the one that has given us the best result with the metric we have chosen has been the Support Vector Machines model.

In [114…

```python
perf = pd.DataFrame.from_dict(performance_auc, orient='index')
perf['Model'] = perf.index
perf['AUC'] = perf[0]
plt.xlabel('AUC')
plt.title('Classifier AUC')
sns.set_color_codes("muted")
sns.barplot(x='AUC', y='Model', data=perf, color="b")
```

Out[114]:

```
<Axes: title={'center': 'Classifier AUC'}, xlabel='AUC', ylabel='Model'>
```



In [118…
```
print(performance_auc)
```

```
{'Logistic Regression': 0.808859815229242, 'Decision Tree': 0.808859815229242, 'Random Forests': 0.7979858839731452, 'k-nearest neighbours': 0.786882423825099, 'SVM': 0.8229184598611351}
```

Clasification models evaluation

Who survived the Titanic?

August 2022