Survival prediction of titanic passengers

Set up

Load modules

In [1]:
```
# libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

#local modules
from barplot import plot_barplot
```

Set display options

In [2]:
```
# allow multiple outputs per cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Plot the Figures Inline
%matplotlib inline

# Prevent label cut off from figures
from matplotlib import rcParams
rcParams.update({'figure.autolayout': True})
```

Data loader

In [3]:
```
# get metadata
meta_data = pd.read_csv("data/metadata.csv")
meta_data
```

Out[3]:

	Variable	Definition	Key
0	survival	Survival	0 = No 1 = Yes
1	pclass	Ticket class	1 = 1st 2 = 2nd 3 = 3rd
2	sex	Sex	NaN
3	Age	Age in years	NaN
4	sibsp	# of siblings / spouses aboard the Titanic	NaN
5	parch	# of parents / children aboard the Titanic	NaN
6	ticket	Ticket number	NaN
7	fare	Passenger fare	NaN
8	cabin	Cabin number	NaN
9	embarked	Port of Embarkation	C = Cherbourg Q = Queenstown S = Southampton

In [4]:
```
# load train data
train_data = pd.read_csv("data/titanic-train.csv")
print("Shape: ", train_data.shape)
train_data.head()
```

```
Shape:  (891, 12)
```

Out[4]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
0	1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1
2	3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
4	5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0

In [5]:
```
# load test data
test_data = pd.read_csv("data/titanic-test.csv")
print("Shape: ", test_data.shape)
test_data.head()
```

```
Shape:  (418, 11)
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

Data exploration

Check if the datasets contain missing values

In [6]:
```
missing_values = pd.DataFrame({'Training set': train_data.isna().sum(),
                               'Test set': test_data.isna().sum()})
missing_values
```

Out[6]:

	Training set	Test set
Age	177	86.0
Cabin	687	327.0
Embarked	2	0.0
Fare	0	1.0
Name	0	0.0
Parch	0	0.0
PassengerId	0	0.0
Pclass	0	0.0
Sex	0	0.0
SibSp	0	0.0
Survived	0	NaN
Ticket	0	0.0

Conclusion: There are many missing values for the age of passengers and the cabin type. Therefore, these features will be excluded from the following analyses.

Count the number of unique values of features of interest

In [7]:
```
train_data["Sex"].nunique()
train_data["SibSp"].nunique()
train_data["Parch"].nunique()
train_data["Fare"].nunique()
```

Out[7]: 2

Out[7]: 7

Out[7]: 7

Out[7]: 248

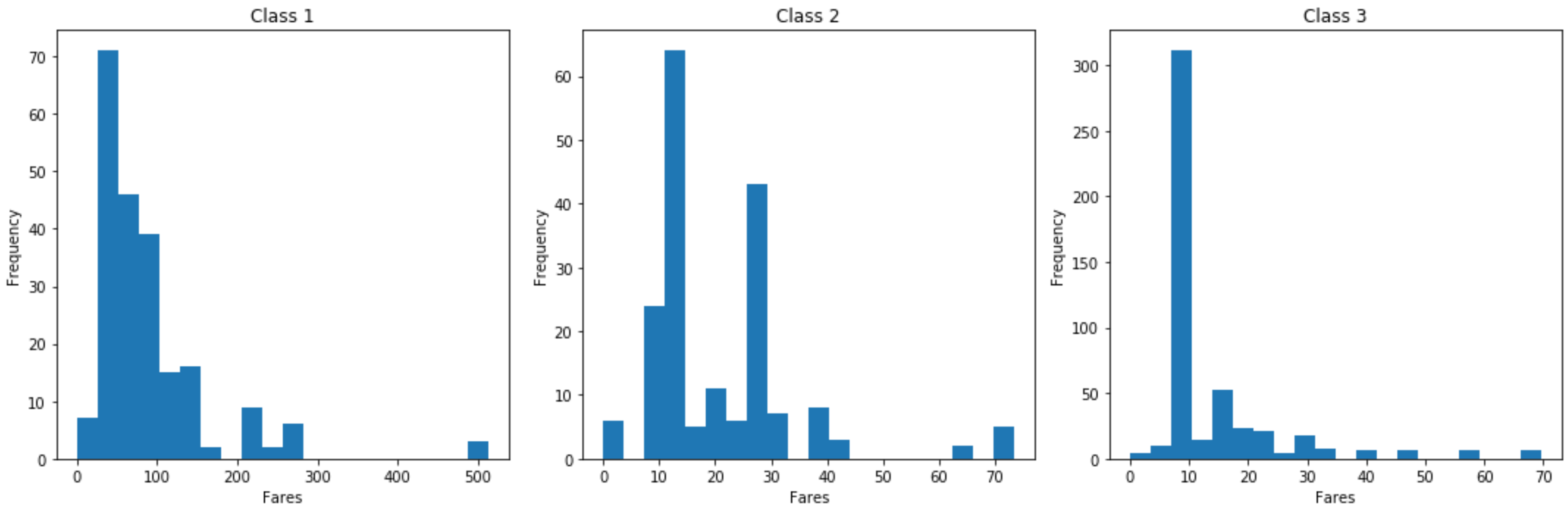
Conclusion: there are many different fares that are assumably associated with the ticket class. Let's check this:

Investigate fares

In [8]:
```
# check min and max prices of fares per class

# divide training dataset per class
class1 = train_data.loc[train_data['Pclass'] == 1]
class2 = train_data.loc[train_data['Pclass'] == 2]
class3 = train_data.loc[train_data['Pclass'] == 3]
# save classes in list
classes = [class1, class2, class3]

# print fare ranges
for i, pclass in enumerate(classes):
    print(f"Max fare class {i+1}: ", pclass["Fare"].max())
    print(f"Min fare class {i+1}: ",pclass["Fare"].min())
    print()
```

```
Max fare class 1:  512.3292
Min fare class 1:  0.0

Max fare class 2:  73.5
Min fare class 2:  0.0

Max fare class 3:  69.55
Min fare class 3:  0.0
```

In [9]:
```
# plot fares per class as histograms

# save fares in numpy array
fares_per_class = [class1["Fare"].to_numpy(),
             class2["Fare"].to_numpy(),
             class3["Fare"].to_numpy()]

# plot fares
fig, ax = plt.subplots(1,len(fares_per_class), figsize=(15, 5))
for i, data in enumerate(fares_per_class):
    _ = ax[i].hist(data, bins=20)
    _ = ax[i].set_title(f"Class {i+1}")
    _ = ax[i].set_xlabel("Fares")
    _ = ax[i].set_ylabel("Frequency")
```



Conclusion: The fares of the 3 different classes overlap, especially the fares of class 2 and 3. It might therefore be more useful to predict survival rates depending on passenger class rather than fare. Let's check among the categorical features if there are categories that are (strongly) associated with survival rate.

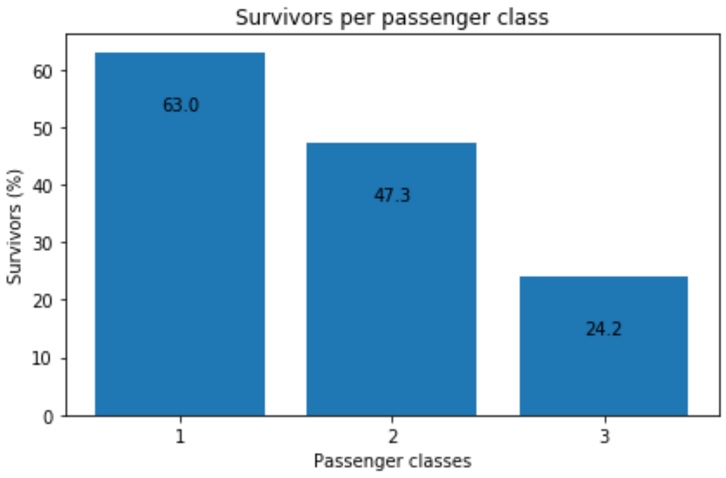
Investigate survival rates per categories

Passenger class:

In [10]:
```
# save categories in list and convert them to string variables for plotting
categories_class = list(map(str, train_data["Pclass"].unique()))
categories_class.sort()
categories_class # check result

# calculate percentage of survivors per passenger class
survivors_per_class = []
for pclass in classes:
    surv = round(pclass["Survived"].sum()/len(pclass["Survived"])*100, 1)
    survivors_per_class.append(surv)
```

Out[10]:
```
['1', '2', '3']
```

In [11]:
```
# plot survivors per class
plot_barplot(categories_class,
             survivors_per_class,
             title="Survivors per passenger class",
             xlabel="Passenger classes")
```



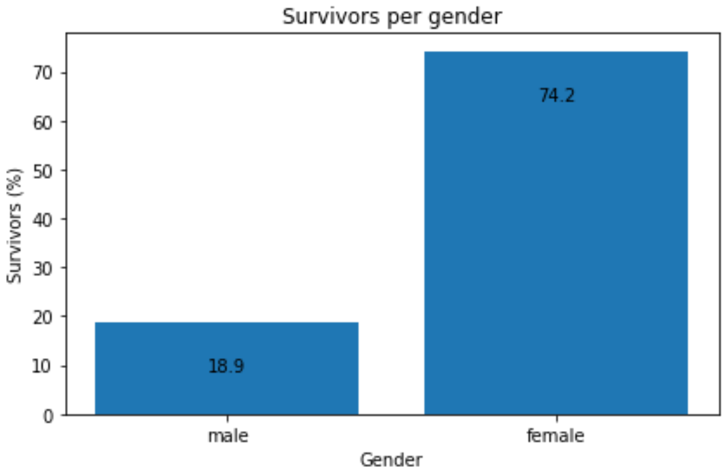
Conclusion: the survival rate seems to be correlated to the passenger class and therefore likely influences the prediction of survival.

Gender:

In [12]:
```
# save categories in list
categories_gender = list(map(str, train_data["Sex"].unique()))
categories_gender # check result

# calculate percentage of survivors per gender
men = train_data.loc[train_data.Sex == 'male']["Survived"].to_numpy()
women = train_data.loc[train_data.Sex == 'female']["Survived"].to_numpy()
men_surv = round(sum(men)/len(men)*100, 1)
women_surv = round(sum(women)/len(women)*100, 1)

# store results in list
survivors_per_gender = [men_surv, women_surv]
```

Out[12]:
```
['male', 'female']
```

In [13]:
```
# plot survivors per gender
plot_barplot(categories_gender,
             survivors_per_gender,
             title="Survivors per gender",
             xlabel="Gender")
```



Conclusion: the survival rate of women is much higher than the survival rate of men. Therefore, the gender likely has a strong influence on the prediction of survival.

Number of siblings/ spouses aboard

In [14]:
```
# save categories in list
categories_sibsp = list(train_data["SibSp"].unique())
categories_sibsp.sort()

# calculate percentage of survivors per number of siblings/ spouses aboard
# and save results in list
survivors_per_sibsp = []
for i in categories_sibsp:
    sibsp = train_data.loc[train_data.SibSp == i]["Survived"].to_numpy()
    survivors_per_sibsp.append(round(sum(sibsp)/len(sibsp)*100, 1))

# convert categories to string variables for plotting
categories_sibsp = list(map(str, categories_sibsp))
categories_sibsp # check result
```

Out[14]:
```
['0', '1', '2', '3', '4', '5', '8']
```

In [15]:
```
# plot survivors per number of siblings/ spouses aboard
plot_barplot(categories_sibsp,
             survivors_per_sibsp,
             title="Survivors per number of siblings/ spouses aboard",
             xlabel="Number of siblings/ spouses aboard")
```

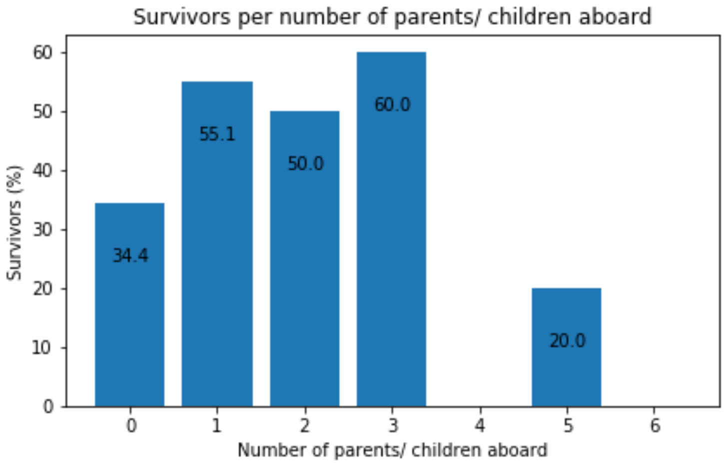
Conclusion: The people with 1 or 2 siblings/ spouses aboard had the highest rate of survival. This could mean that these people had support from family members with getting a spot in one of the lifeboats. Therefore, the number of siblings/ spouses might be associated with the chance of survival.

Number of parents/ children aboard

In [16]:
```
# save categories in list
categories_parch = list(train_data["Parch"].unique())
categories_parch.sort()

# calculate percentage of survivors per number of parents/ children aboard
# and save results in list
survivors_per_parch = []
for i in categories_parch:
    parch = train_data.loc[train_data.Parch == i]["Survived"].to_numpy()
    survivors_per_parch.append(round(sum(parch)/len(parch)*100, 1))

# convert categories to string variables for plotting
categories_parch = list(map(str, categories_parch))
categories_parch # check result
```

Out[16]:
```
['0', '1', '2', '3', '4', '5', '6']
```

In [17]:
```
# plot survivors per number of parents/ children aboard
plot_barplot(categories_parch,
             survivors_per_parch,
             title="Survivors per number of parents/ children aboard",
             xlabel="Number of parents/ children aboard")
```



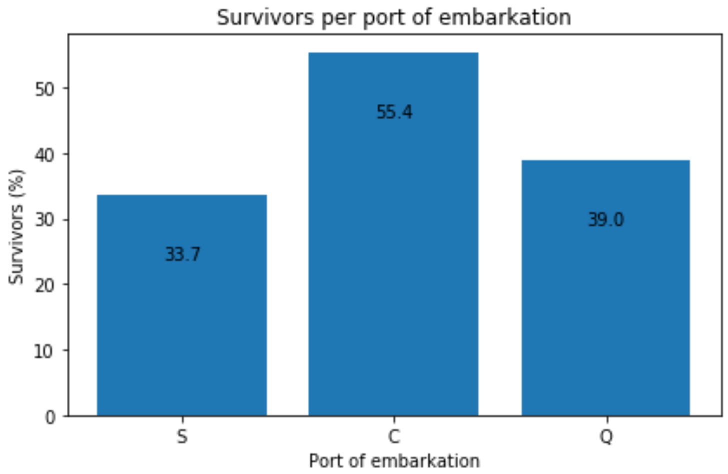
Conclusion: The people who had between 1 and 3 parents/ children aboard had the highest rate of survival. As above, this could mean that these people had support from family members with getting a spot in one of the lifeboats. Therefore, the number of parents/ children might be associated with the chance of survival.

Port of embarkation

In [18]:
```
# save categories in list
categories_embarked = list(map(str, train_data["Embarked"].unique()))
categories_embarked

# calculate percentage of survivors per port of embarkation
# note: leave out the two passengers of unknown port of embarkation
survivors_per_port = []
for i in categories_embarked[:3]:
    port = train_data.loc[train_data.Embarked == i]["Survived"].to_numpy()
    survivors_per_port.append(round(sum(port)/len(port)*100, 1))
```

Out[18]:
```
['S', 'C', 'Q', 'nan']
```

In [19]:
```
# plot survivors per port of embarkation
plot_barplot(categories_embarked[:3],
             survivors_per_port,
             title="Survivors per port of embarkation",
             xlabel="Port of embarkation")
```



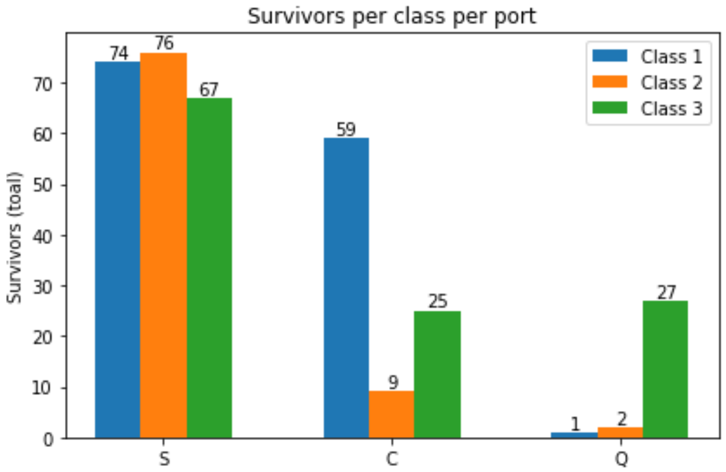
Conclusion: the percentage of people who embarked in Cherbourg is higher compared to Southampton and Queenstown. This could be due to many first class passengers having embarked here. Let's check this:

Passengers per class per port

In [20]:
```
# calculate percentage of survivors per class and port of embarkation
# note: leave out the two passengers of unknown port of embarkation

survivors_class_port = []
# loop over classes
for pclass in classes:
    survivors_per_port_pclass = []
    # loop over ports
    for cat in categories_embarked[:3]:
        port = pclass.loc[pclass.Embarked == cat]["Survived"].to_numpy().sum()
        survivors_per_port_pclass.append(port)
    survivors_class_port.append(survivors_per_port_pclass)

survivors_class_port
```

Out[20]:
```
[[74, 59, 1], [76, 9, 2], [67, 25, 27]]
```

In [21]:
```
# plot survivors per class per port of embarkation

# set variables
x = np.arange(len(categories_embarked[:3]))  # the label locations
width = 0.2  # the width of the bars

# set up plot
fig, ax = plt.subplots()
_ = ax.set_title("Survivors per class per port")
_ = ax.set_ylabel("Survivors (toal)")
_ = ax.set_xticks(x)
_ = ax.set_xticklabels(categories_embarked[:3])

# plot barplot
for i,j in zip(survivors_class_port,range(-1,2)):
    _ = ax.bar(x=x+width*j, height=i, width=width, label=f'Class {j+2}')
    # annotate barplot
    for k, data in enumerate(i):
        _ = ax.annotate(s=data, xy=(k+width*j, data+0.7), ha='center')
_ = ax.legend()
```



Conclusion: Most passengers, irrespective of class, embarked in Southampton. However, in Cherbourg a higher number of first class passengers embarked compared to second and thrid class passengers. Additionally, in Queenstown a higher number of third class passengers embarked compared to first and second class passengers. Therefore, the port of embarkation might have a weak influence on the prediction of survival.

Summary

Based on this data exploration, the features that likely influence the prediction of survival are in presumed descending order of strength:

- gender
- passenger class
- siblings/ spouses aboard, children/ parents aboard
- port of embarkation/ fare

Models

Random forest model

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

y = train_data["Survived"]

features = ["Pclass", "Sex", "SibSp", "Parch"]
X = pd.get_dummies(train_data[features])
X_test = pd.get_dummies(test_data[features])

model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

output = pd.DataFrame({'PassengerId': test_data.PassengerId, 'Survived': predictions})
output.to_csv('my_submission.csv', index=False)
```

In [ ]:
```
output
```

In [ ]:

In [ ]:

In [ ]: