Exploratory data analysis exercise. Honors project.

Pablo Hinojosa Lopez

The set of data chosen is about the tragic episode happened on April 15, 1912, when the famous Titanic collided with an iceberg, broken the main structure of the ship. Unfortunately, there were no enough lifeboats on board for all the people, resulting this in the death of 1502 passengers.

The main purpose of this analysis is to find out if there was some passanger characteristics that could lead to have more advantages to survive to this tragedy.

¿Rich people had preferences?

¿Kids and woman had preferences?

¿Depending on the cabin colocation some passenger were able to arrive before to these boats?

I will try to shape and respond all these kind of questions.

In [345…

```
%matplotlib inline
%config InlineBackend.figure_formats = ['retina']

# Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn

from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold
from scipy import stats
from scipy.stats import chi2_contingency
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn import preprocessing

warnings.filterwarnings("ignore")
sns.set(rc={"figure.figsize": (20, 15)})
sns.set_style("whitegrid")
```

Variable description

- PassengerId: unique id number to each passenger
- Survived: passenger survive(1) or died(0)
- Pclass: passenger class
- Name: name
- Sex: gender of passenger
- Age: age of passenger
- SibSp: number of siblings/spouses
- Parch: number of parents/children
- Ticket: ticket number
- Fare: amount of money spent on ticket
- Cabin: cabin category
- Embarked: port where passenger embarked (C = Cherbourg, Q = Queenstown, S = Southampton)

In [346…

```
# Now it is turn to load our train dataset because in this case I will only perform the initial exploratory data

df_train = pd.read_csv("train.csv")

print(f"Train set shape:\n{df_train.shape}\n")
```

```
Train set shape:
(891, 12)
```

Let's analyse the different variables in our data frame and hint the most interesting details about each of them

In [347…
```
# info of each of the variables in our train set
df_train.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
```

As we could imagine, the target variable will be 'Survived' column which determines if the passenger finally survived or not.

Also with the info we can see that there are some variables that have missing data so we will manage this later

In [348…
```
# Lets take a view on thej first data of the set
df_train.head(10)
```

Out[348…

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1	0	PC 17599	71.28	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.92	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.10	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.05	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.46	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.00	0	0	17463	51.86	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.00	3	1	349909	21.07	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0	2	347742	11.13	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1	0	237736	30.07	NaN	C

In [349…
```
# Lets drop the passengerid from our dataset because as we could see is only a number to maintain a counter of th
df_train.drop(columns=["PassengerId"], axis=1, inplace=True)
df_train.head(5)
```

Out[349…

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.25	NaN	S
1	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.00	1	0	PC 17599	71.28	C85	C
2	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.92	NaN	S
3	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.10	C123	S
4	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.05	NaN	S

Explore numerical features

In [350…
```
# Lets select columns with numerical data
df_train_num = df_train.select_dtypes(exclude=["object"])
```

```
df_train_num.head()
```

Out[350…

	Survived	Pclass	Age	SibSp	Parch	Fare
0	0	3	22.00	1	0	7.25
1	1	1	38.00	1	0	71.28
2	1	3	26.00	0	0	7.92
3	1	1	35.00	1	0	53.10
4	0	3	35.00	0	0	8.05

In [351…

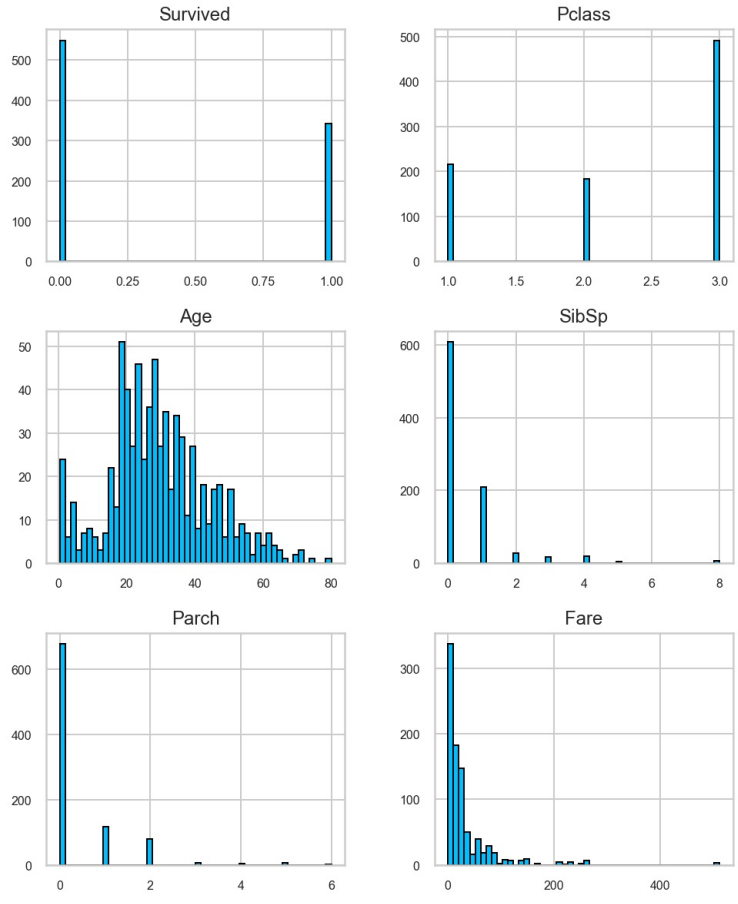
```
# Lets drop columns where the values are mainly constants because they are not to offer information about the dat
sel = VarianceThreshold(threshold=0.05)

# fit finds the features with constant variance
sel.fit(df_train_num.iloc[:, 1:])

# get the number of features that are not constant
print(f"Number of retained features: {sum(sel.get_support())}")

print(f"\nNumber of quasi_constant features: {len(df_train_num.iloc[:, 1:].columns) - sum(sel.get_support())}")

quasi_constant_features_list = [x for x in df_train_num.iloc[:, 1:].columns if x not in df_train_num.iloc[:, 1:].
print(f"\nQuasi-constant features to be dropped: {quasi_constant_features_list}")
```

```
Number of retained features: 5

Number of quasi_constant features: 0

Quasi-constant features to be dropped: []
```

There is no quasi-constant features, so is not needed to drop extra column.

In [352…

```
#Lets plot the distribution of all the numerical data
fig_ = df_train_num.hist(figsize=(8, 10), bins=50, color="deepskyblue",
                         edgecolor="black", xlabelsize=8, ylabelsize=8)
```



Here in these previous plots we can see that, as we could intuit, there are more passenger with low cost fares than passengers with high cost fares.

Also its important to notice that the mean age is about 25 - 30 years old.

In [353…

```
# Lets take a view on the percentage of passengers survived
survived_passengers = df_train.Survived.value_counts()[1]
print(f"\n% Survived passengers: {survived_passengers / len(df_train)}")
```

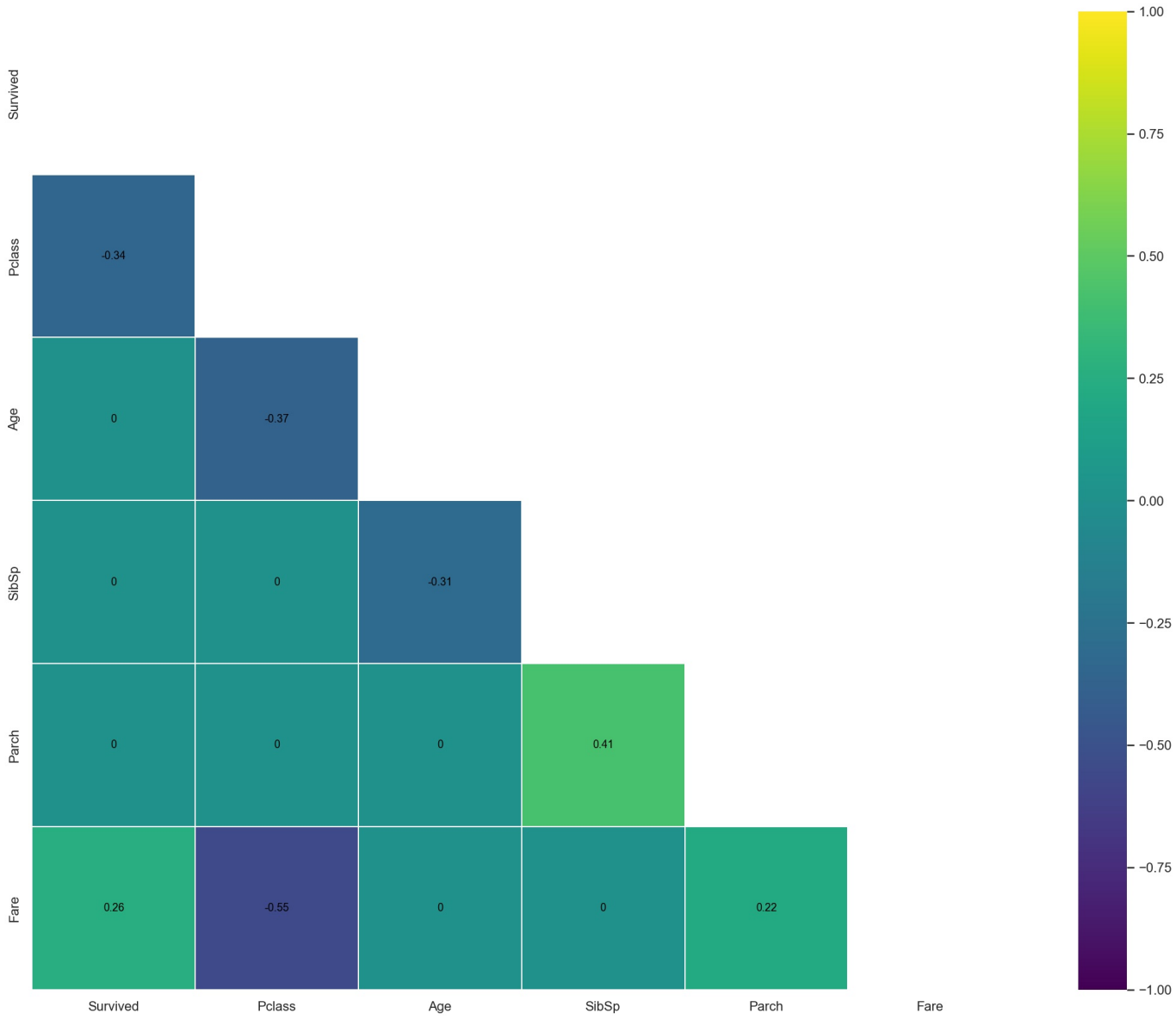
```
% Survived passengers: 0.3838383838383838
```

In [354…

```
# Heatmap for all the remaining numerical data including the target 'Survived'

# This will show us the correlations between variables. I will try to mark the variables which have more than 0.3
pd.options.display.float_format = "{:,.2f}".format

# Define correlation matrix
corr_matrix = df_train_num.corr()

# Replace correlation < |0.20| by 0 for a better visibility
corr_matrix[(corr_matrix < 0.20) & (corr_matrix > -0.20)] = 0

# Mask the upper part of the heatmap
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

# Choose the color map
cmap = "viridis"

# plot the heatmap
sns.heatmap(corr_matrix, mask=mask, vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot_kws={"size": 9, "color": "black"}, square=True, cmap=cmap, annot=True)
```

Out[354…

```
<AxesSubplot:>
```



Apparently the main variables that will affect to the target variable will be in relation with the fare and class of the ticket.

This hint could lead us to think that the money was an important factor to determine if a passenger will survive or not. Also to know that wealth in general is an important parameter to take into account.

Also we note that class and age are stronly correlated, which is logical because when you are older, your monetary capacity increase along the years.

In [355…]
```
# Let's select features where the correlation with 'SalePrice' is higher than |0.2|
# -1 because the latest row is SalePrice
df_num_corr = df_train_num.corr()["Survived"][1:]

# Correlated features (r2 > 0.5)
high_features_list = df_num_corr[abs(
    df_num_corr) >= 0.5].sort_values(ascending=False)
print(f"{len(high_features_list)} strongly correlated values with SalePrice:\n{high_features_list}\n")

# Correlated features (0.2 < r2 < 0.5)
low_features_list = df_num_corr[(abs(df_num_corr) < 0.5) & (
    abs(df_num_corr) >= 0.2)].sort_values(ascending=False)
print(
    f"{len(low_features_list)} slightly correlated values with SalePrice:\n{low_features_list}")
```

```
0 strongly correlated values with SalePrice:
Series([], Name: Survived, dtype: float64)

2 slightly correlated values with SalePrice:
Fare      0.26
Pclass   -0.34
Name: Survived, dtype: float64
```

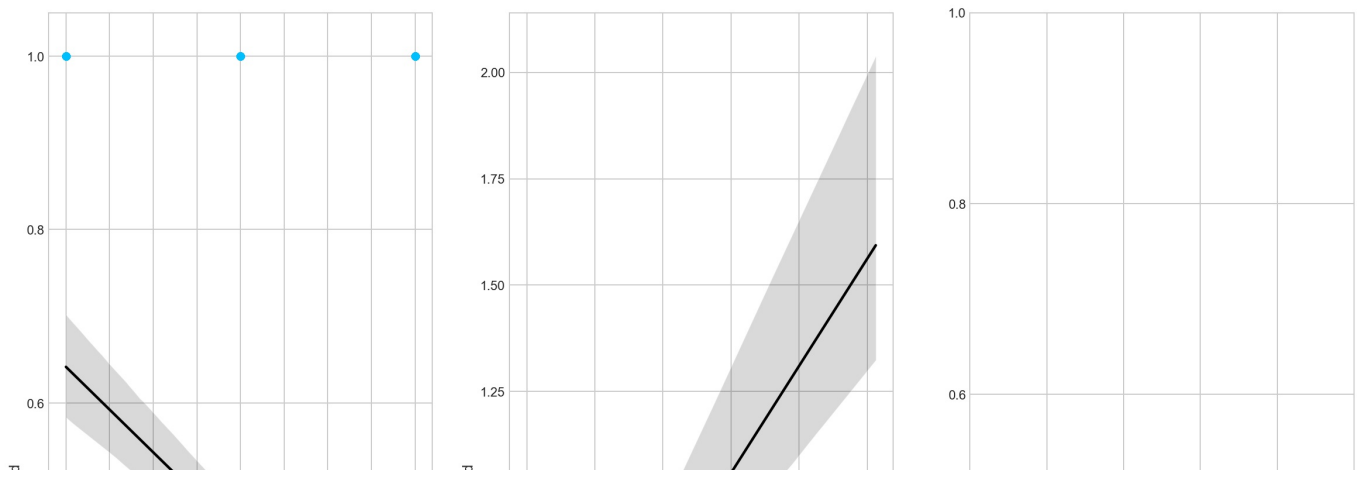
In [356…]
```
# Plcass vs Survived
df_train[["Pclass","Survived"]].groupby(["Pclass"], as_index = False).mean().sort_values(by="Survived",ascending
```

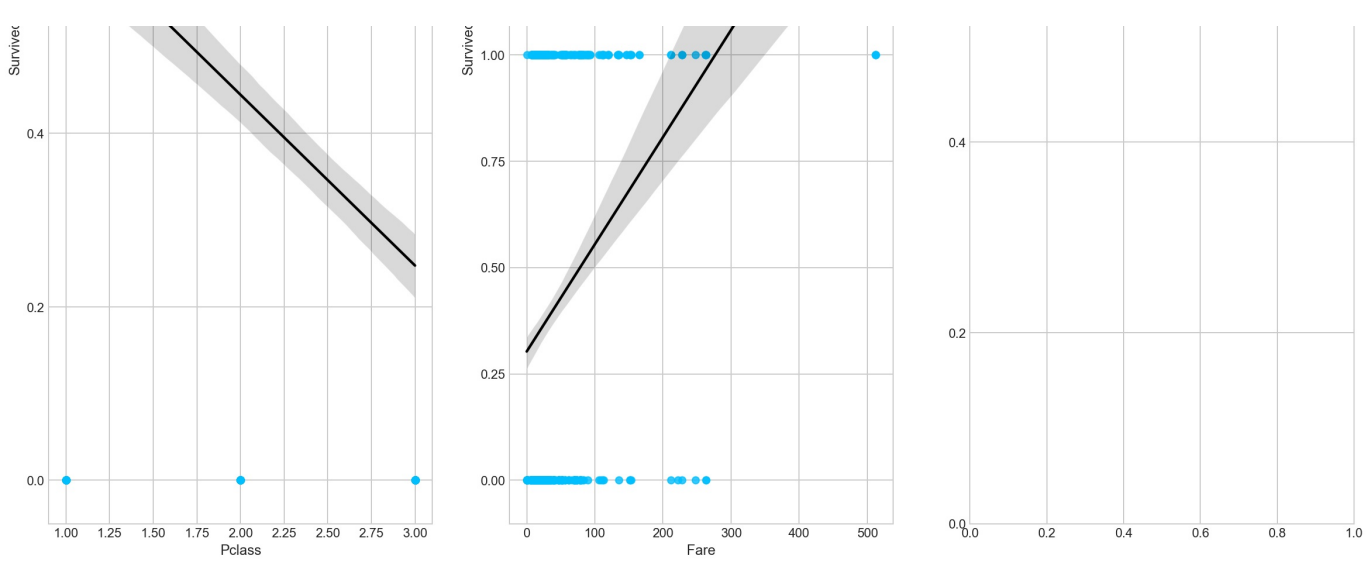
Out[356…]

	Pclass	Survived
0	1	0.63
1	2	0.47
2	3	0.24

Because there is no features highly correlated with the target variable, we could proceed with the representation of the relation with the sightly correlated variables.

In [357…]
```
# Features with low correlation (between 0.2 and 0.5)
low_features = df_num_corr[(abs(df_num_corr) >= 0.2) & (
    abs(df_num_corr) < 0.5)].index.tolist()
low_features.append("Survived")
df_low_features = df_train_num.loc[:, low_features]
df_low_features.head(5)

plt.style.use("seaborn-whitegrid")  # define figures style
fig, ax = plt.subplots(round(len(low_features) / 3), 3)

for i, ax in enumerate(fig.axes):
    # plot the correlation of each feature with SalePrice
    if i < len(low_features) - 1:
        sns.regplot(x=low_features[i], y="Survived", data=df_low_features, ax=ax, scatter_kws={
                    "color": "deepskyblue"}, line_kws={"color": "black"},)
```





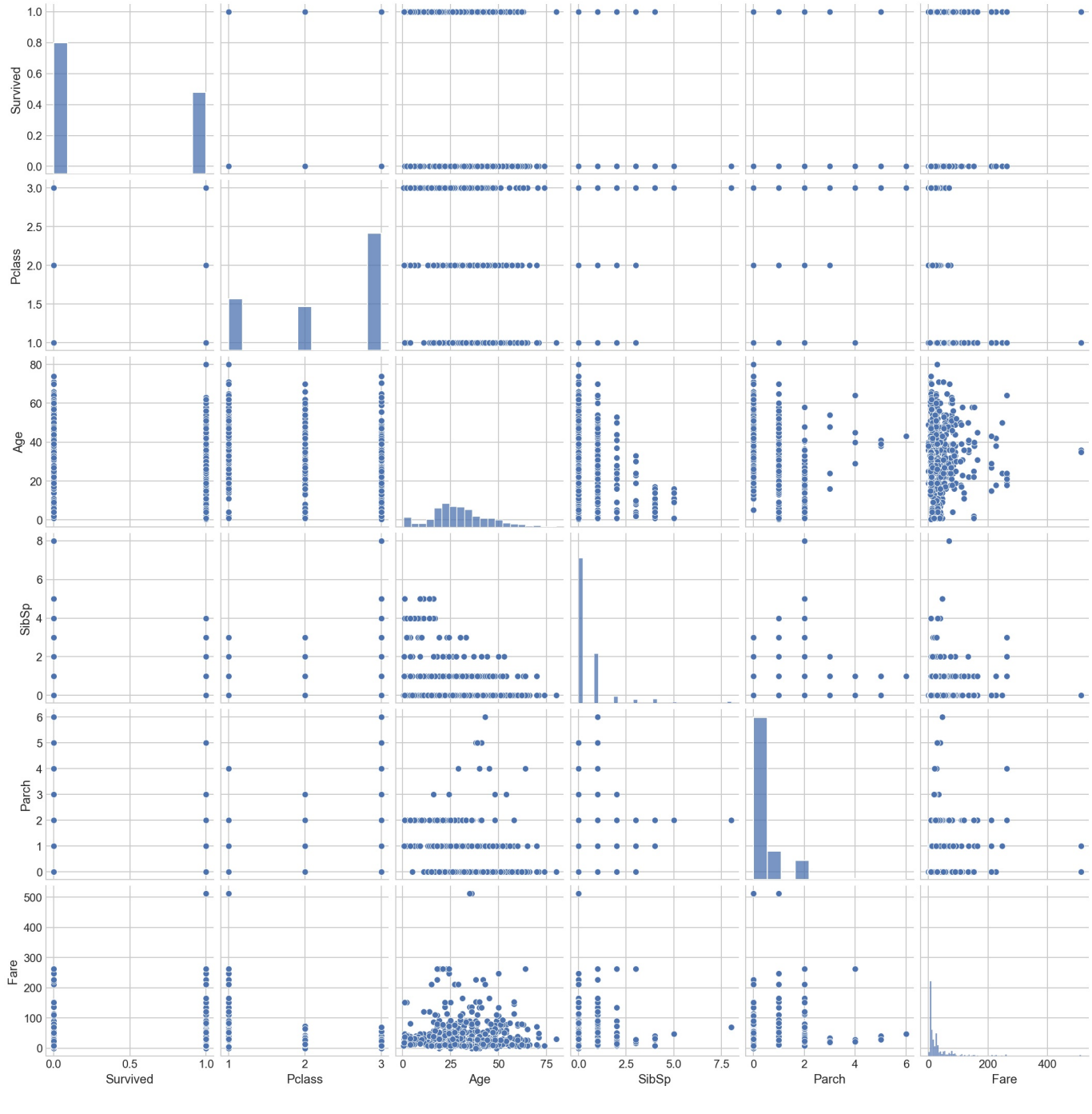
In addition to plot and inspect the correlations, it is interesting to plot the scatter plots for all the other numerical features that are correlated between them.

In [358…

```
sns.pairplot(df_train_num)
```

Out[358…

```
<seaborn.axisgrid.PairGrid at 0x2964307dfc0>
```



As we can see this pairplot does not offer so much information about the relations between variable. Lets try to plot some special relationship that could lead us to have more idea about data relations.

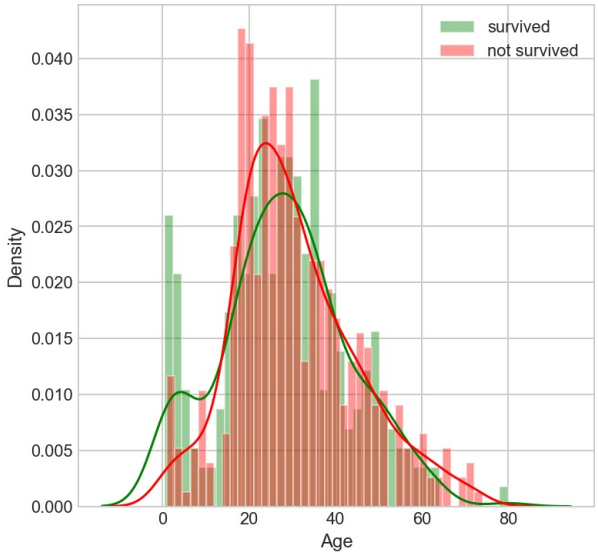
It will be interesting to plot relation between fare and class (which is a natural relation). Also it will be interesting the relation between survived and age and also between survived and number of sons, parents.

In [359…

```
survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=1,figsize=(6, 6))
ax = sns.distplot(df_train_num[df_train_num['Survived']==1].Age.dropna(), bins=40, label = survived, ax = axes, c
ax = sns.distplot(df_train_num[df_train_num['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axe
ax.legend()
```

Out[359…

```
<matplotlib.legend.Legend at 0x2961cd0e6b0>
```



It can be seen how there is not much relationship between age and survival, except for an age range such as babies, in which case it does seem to affect survival. Let's proceed to analyze that age range to check if, according to the story, babies and children had preference.

In [360…

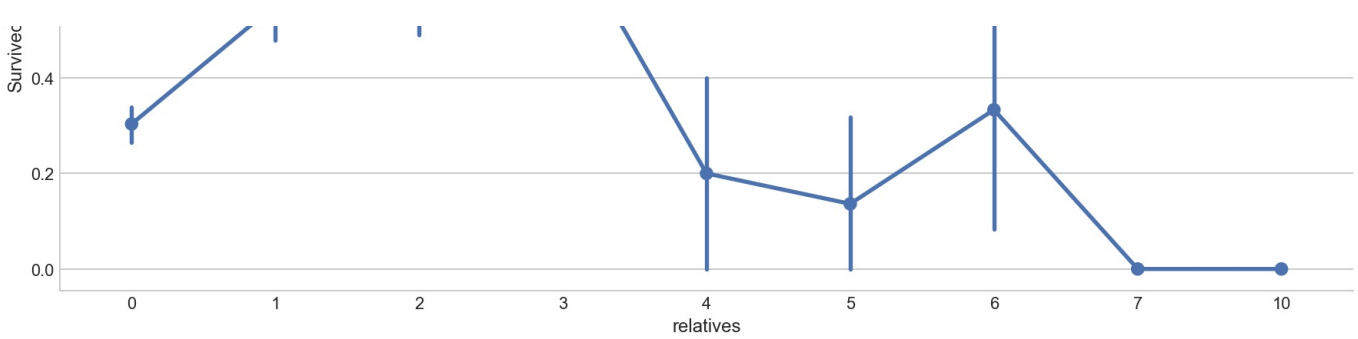
```
child_data = df_train_num[df_train_num['Age'] < 10]

survived_child = len(child_data[child_data["Survived"] == 1])
not_survived_child = len(child_data[child_data["Survived"] == 0])

print(f"\nPercentage of survived child: {survived_child / len(child_data)}")
print(f"\nPercentage of not survived child: {not_survived_child / len(child_data)}")
```

```
Percentage of survived child: 0.6129032258064516

Percentage of not survived child: 0.3870967741935484
```

Let's see how could affect the number of members in the family to the 'Survived' variable. To do this, I am going to do a little of Feature Engineering to transform a little bit the data.

In [361…

```
data = [df_train]
for dataset in data:
    dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
    dataset.loc[dataset['relatives'] > 0, 'travelled_alone'] = 'No'
    dataset.loc[dataset['relatives'] == 0, 'travelled_alone'] = 'Yes'
axes = sns.factorplot('relatives','Survived',
                      data=df_train, aspect = 2.5, )
```



As we can see in the graph, appear to not be very importan to have more family or not.

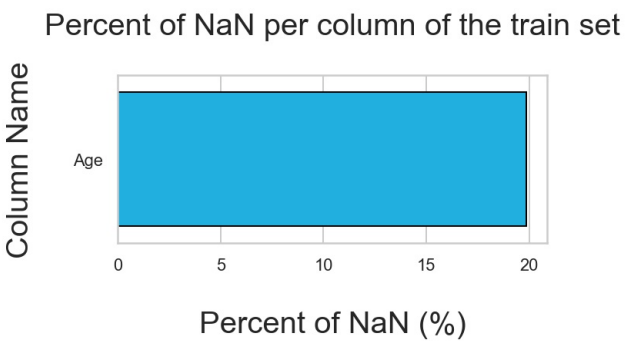
Also as we can see, it appears to be a sightly correlation between age and surviving factors plotted in previous graphs. No it is turn to analyse the missing values in numerical features.

In [362…

```
# Check the NaN of the train set by ploting percent of missing values per column
column_with_nan = df_train_num.columns[df_train_num.isnull().any()]
column_name = []
percent_nan = []

for i in column_with_nan:
    column_name.append(i)
    percent_nan.append(
        round(df_train_num[i].isnull().sum()*100/len(df_train_num), 2))

tab = pd.DataFrame(column_name, columns=["Column"])
tab["Percent_NaN"] = percent_nan
tab.sort_values(by=["Percent_NaN"], ascending=False, inplace=True)

# Define figure parameters
sns.set(rc={"figure.figsize": (5, 2)})
sns.set_style("whitegrid")

# Plot results
p = sns.barplot(x="Percent_NaN", y="Column", data=tab,
                edgecolor="black", color="deepskyblue")

p.set_title("Percent of NaN per column of the train set\n", fontsize=20)
p.set_xlabel("\nPercent of NaN (%)", fontsize=20)
p.set_ylabel("Column Name\n", fontsize=20)
```

Out[362…

```
Text(0, 0.5, 'Column Name\n')
```



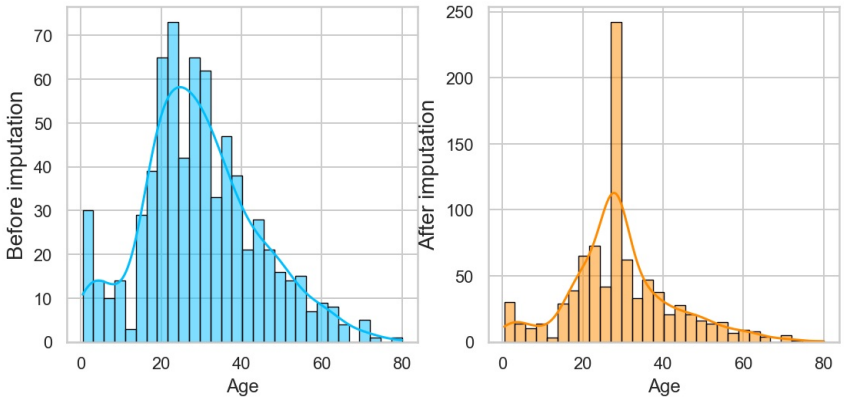
In [363…

```
# Lets fill this missing values with a simpleimputer using median values
# # Imputation of missing values (NaNs) with SimpleImputer
my_imputer = SimpleImputer(strategy="median")
df_train_imputed = pd.DataFrame(my_imputer.fit_transform(df_train_num))
df_train_imputed.columns = df_train_num.columns
```

In [364…

```
# Lets check the distribution of each imputed feature before and after imputation

# Define figure parameters
sns.set(rc={"figure.figsize": (9, 4)})
sns.set_style("whitegrid")
fig, axes = plt.subplots(1, 2)

# Plot the results
for feature, fig_pos in zip(["Age"], [0]):

    """Features distribution before and after imputation"""
```

```
    # before imputation
    p = sns.histplot(ax=axes[0], x=df_train_num[feature],
                     kde=True, bins=30, color="deepskyblue", edgecolor="black")
    p.set_ylabel(f"Before imputation", fontsize=14)

    # after imputation
    q = sns.histplot(ax=axes[1], x=df_train_imputed[feature],
                     kde=True, bins=30, color="darkorange", edgecolor="black")
    q.set_ylabel(f"After imputation", fontsize=14)
```



As we can see, the SimpleImputer is not useful in this case because it only replaces with the same data and provokes a distribution irreal perturbation.

Categorical Features

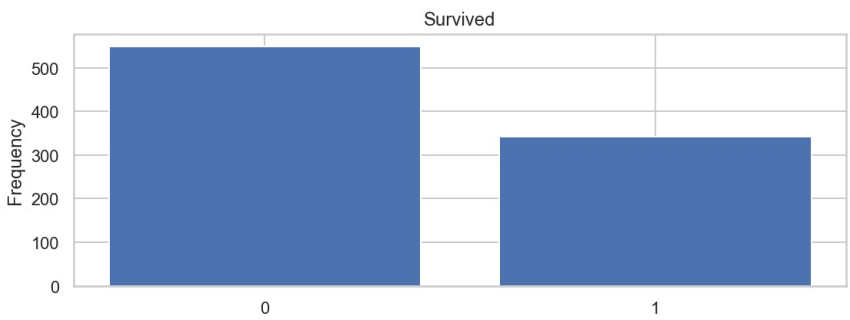
Now I will perform the same analysis for the categorical features in our dataset.

In [365…

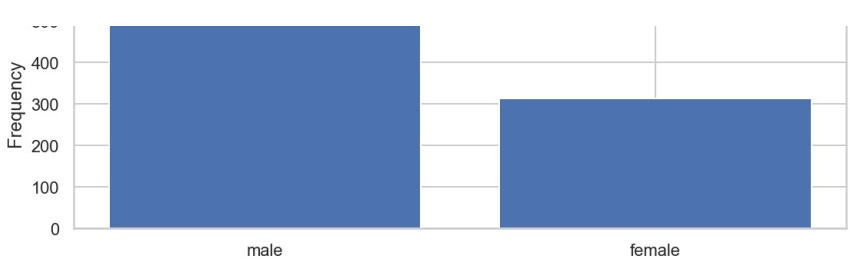
```
# Let's explore a littl bit the count of any of the categorial values.

def bar_plot(variable):
    """
        input: variable ex: "Sex"
        output: bar plot & value count
    """
    # get feature
    var = df_train[variable]
    # count number of categorical variable(value/sample)
    varValue = var.value_counts()

    # visualize
    plt.figure(figsize = (9,3))
    plt.bar(varValue.index, varValue)
    plt.xticks(varValue.index, varValue.index.values)
    plt.ylabel("Frequency")
    plt.title(variable)
    plt.show()
    print("{}: \n {}".format(variable,varValue))
```

In [366…

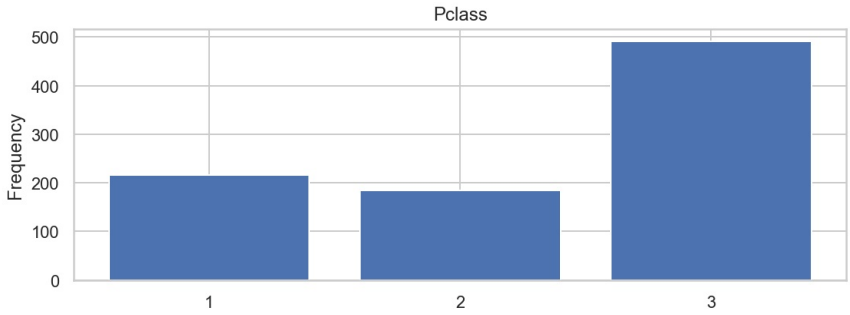
```
category1 = ["Survived","Sex","Pclass","Embarked","SibSp", "Parch"]
for c in category1:
    bar_plot(c)
```



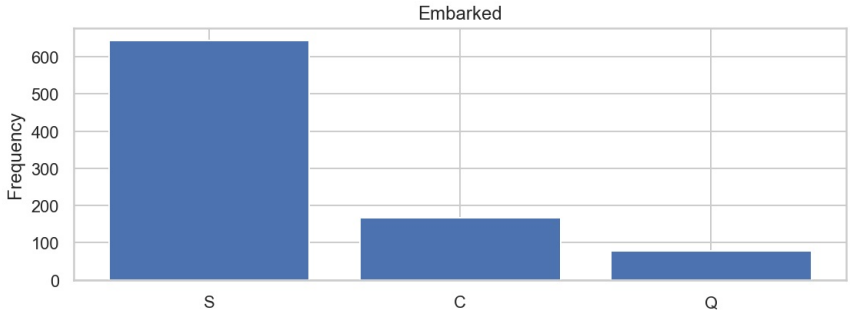

```
Survived:
 0    549
1    342
Name: Survived, dtype: int64
```



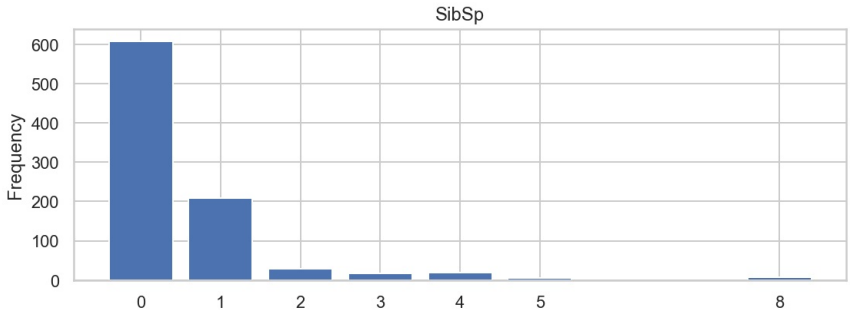

```
Sex:
 male      577
female    314
Name: Sex, dtype: int64
```



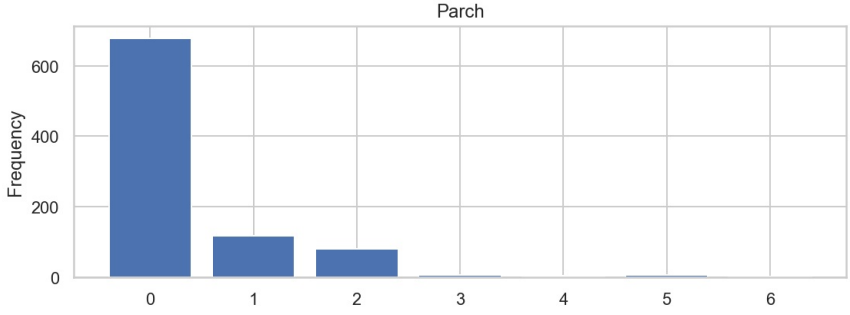

```
Pclass:
 3    491
1    216
2    184
Name: Pclass, dtype: int64
```




```
Embarked:
 S    644
C    168
Q     77
Name: Embarked, dtype: int64
```




```
SibSp:
 0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```




```
Parch:
 0    678
```

```
1    118
2     80
5      5
3      5
4      4
6      1
Name: Parch, dtype: int64
```

In [367…

```
# Let's analyse the categorical variables with more possibilities in their values

category2 = ["Cabin", "Name", "Ticket"]
for c in category2:
    print("{} \n".format(df_train[c].value_counts()))
```

```
B96 B98        4
G6             4
C23 C25 C27    4
C22 C26        3
F33            3
              ..
E34            1
C7             1
C54            1
E36            1
C148           1
Name: Cabin, Length: 147, dtype: int64

Braund, Mr. Owen Harris                     1
Boulos, Mr. Hanna                           1
Frolicher-Stehli, Mr. Maxmillian            1
Gilinski, Mr. Eliezer                       1
Murdlin, Mr. Joseph                         1
                                           ..
Kelly, Miss. Anna Katherine "Annie Kate"    1
McCoy, Mr. Bernard                          1
Johnson, Mr. William Cahoone Jr             1
Keane, Miss. Nora A                         1
Dooley, Mr. Patrick                         1
Name: Name, Length: 891, dtype: int64

347082      7
CA. 2343    7
1601        7
3101295     6
CA 2144     6
           ..
9234        1
19988       1
2693        1
PC 17612    1
370376      1
Name: Ticket, Length: 681, dtype: int64
```

In [368…

```
# Categorical to Quantitative relationship
categorical_features = [
    i for i in df_train.columns if df_train.dtypes[i] == "object"]
categorical_features.append("Survived")

# Train set
df_train_categ = df_train[categorical_features]
df_train_categ.head(5)
```

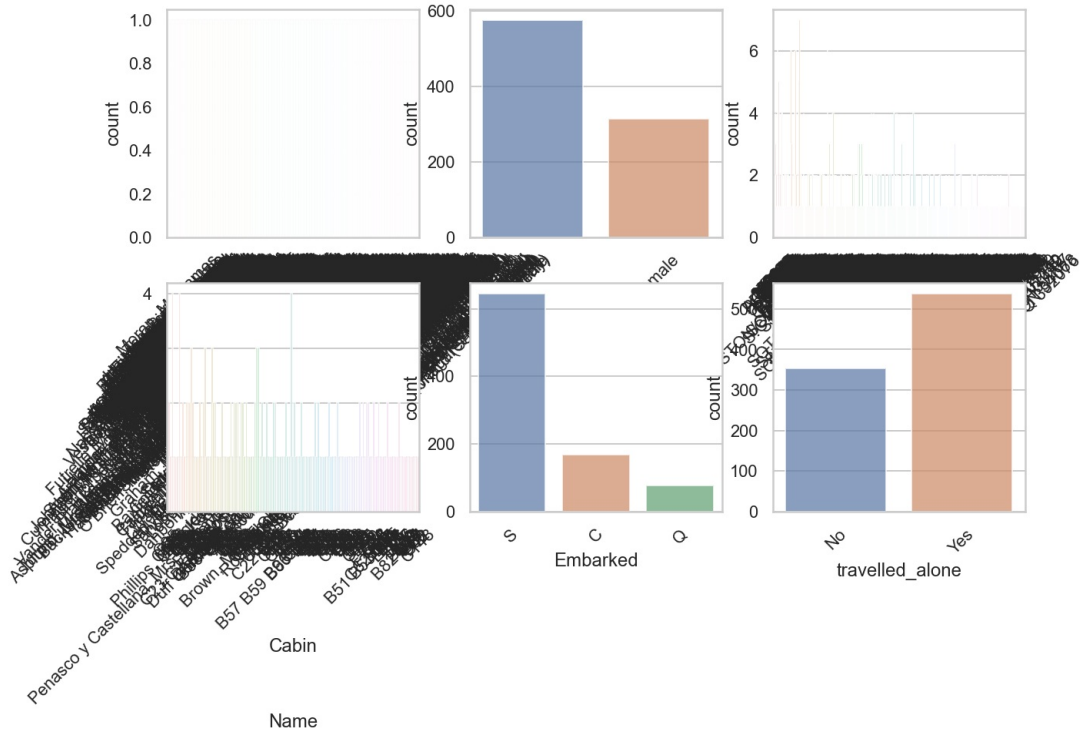
Out[368…

	Name	Sex	Ticket	Cabin	Embarked	travelled_alone	Survived
0	Braund, Mr. Owen Harris	male	A/5 21171	NaN	S	No	0
1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	PC 17599	C85	C	No	1

2	Heikkinen, Miss. Laina	female	STON/O2. 3101282	NaN	S	Yes	1
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	113803	C123	S	No	1
4	Allen, Mr. William Henry	male	373450	NaN	S	Yes	0

In [369…

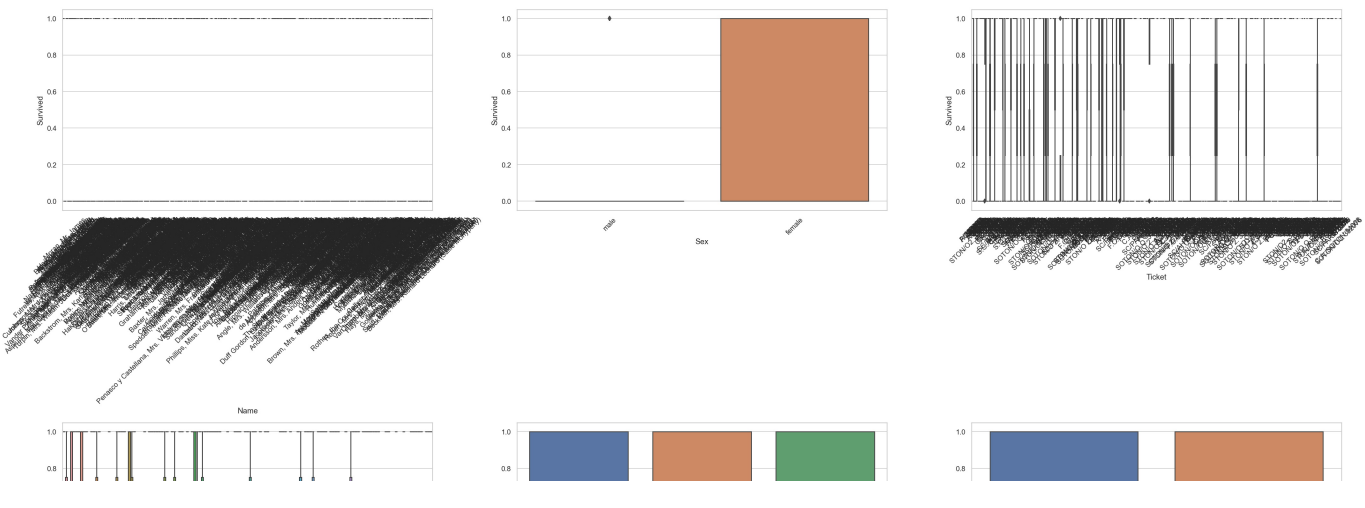
```
# Countplot for each of the categorical features in the train set
fig, axes = plt.subplots(round(len(df_train_categ.columns) / 3), 3, figsize=(10, 6))

for i, ax in enumerate(fig.axes):
    # plot barplot of each feature
    if i < len(df_train_categ.columns) - 1:
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
        sns.countplot(
            x=df_train_categ.columns[i], alpha=0.7, data=df_train_categ, ax=ax)
```

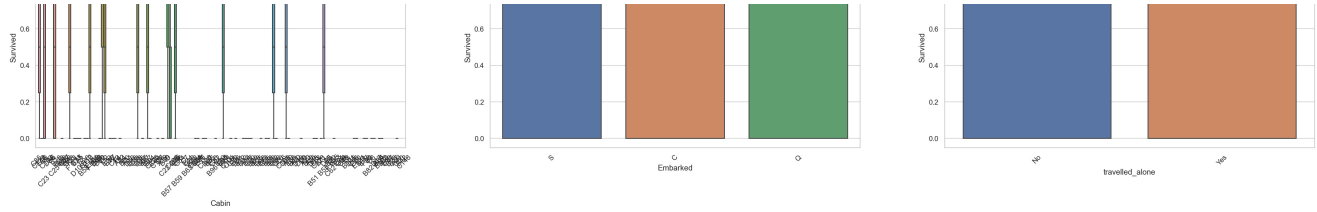


The previous plots show that there is no categorical feature in which only a value is predominant, so that we could delete that feature because not contains any useful information.

In [370…

```
# With the boxplot we can see the variation of the target 'Survived' in each of the categorical features
fig, axes = plt.subplots(
    round(len(df_train_categ.columns)/3), 3, figsize=(30, 15))

for i, ax in enumerate(fig.axes):
    # plot the variation of SalePrice in each feature
    if i < len(df_train_categ.columns) - 1:
        ax.set_xticklabels(ax.xaxis.get_majorticklabels(), rotation=45)
        sns.boxplot(
            x=df_train_categ.columns[i], y="Survived", data=df_train_categ, ax=ax)

fig.tight_layout()
```

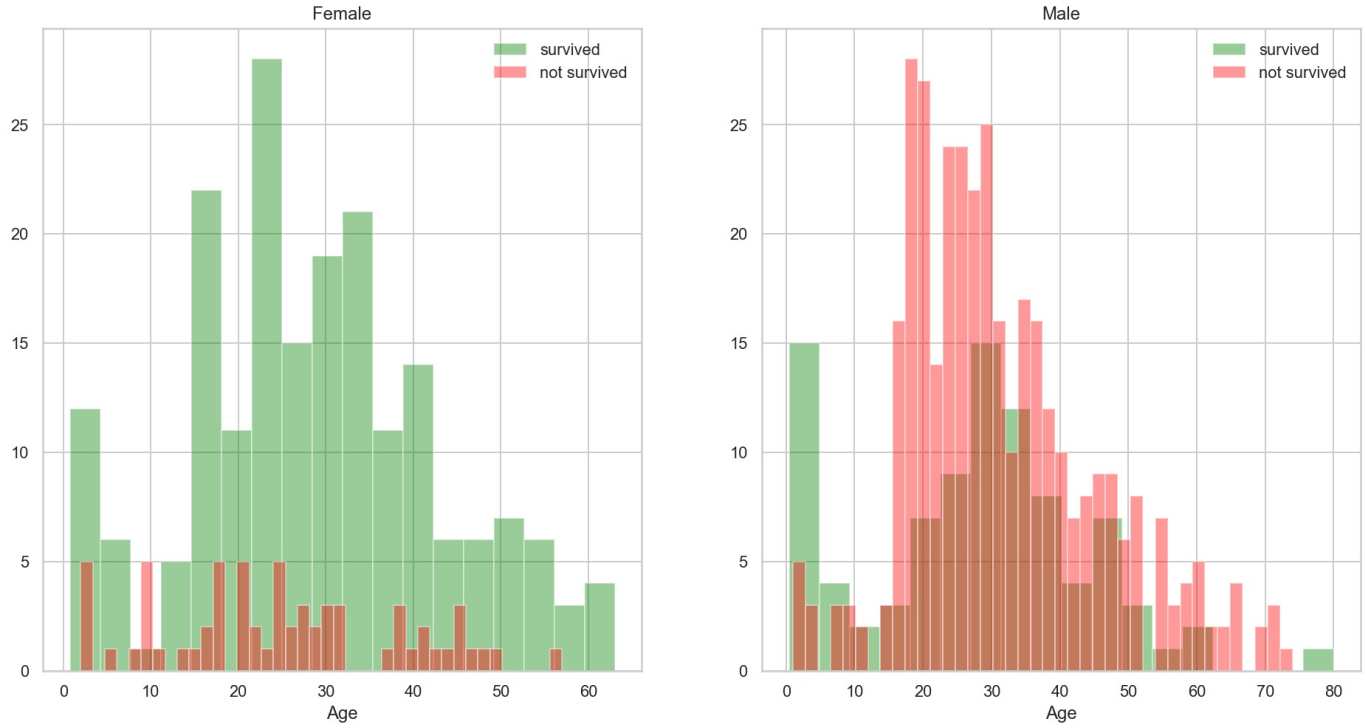




As we can see in this case and because the target variable is a binary variable, this type of plots is not useful, so lets try another kind of representation.

In [371…

```
# Lets analyse the male or female influence depending on the Age
survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(16, 8))
women = df_train[df_train['Sex']=='female']
men = df_train[df_train['Sex']=='male']
ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[0], kde =False,
ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[0], kde =Fal
ax.legend()
ax.set_title('Female')
ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False, col
ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[1], kde = False,
ax.legend()
_ = ax.set_title('Male');
```



We can see that appears to influence the genre and more females survived to the disaster, so we could model it taking into account this influence.

In [372…

```
# We will do feature engineering and create a new sex that is children, so that we could differenciate between ma
print(f"Female before Feature engineering : {len(df_train[df_train['Sex']=='female'])}")
print(f"Male before Feature engineering : {len(df_train[df_train['Sex']=='male'])}")

df_train.loc[df_train['Age'] < 15, 'Sex'] = "Children"

print(f"Female after Feature engineering : {len(df_train[df_train['Sex']=='female'])}")
print(f"Male after Feature engineering : {len(df_train[df_train['Sex']=='male'])}")
print(f"Children after Feature engineering : {len(df_train[df_train['Sex']=='Children'])}")
```

```
Female before Feature engineering : 314
Male before Feature engineering : 577
Female after Feature engineering : 275
Male after Feature engineering : 538
Children after Feature engineering : 78
```

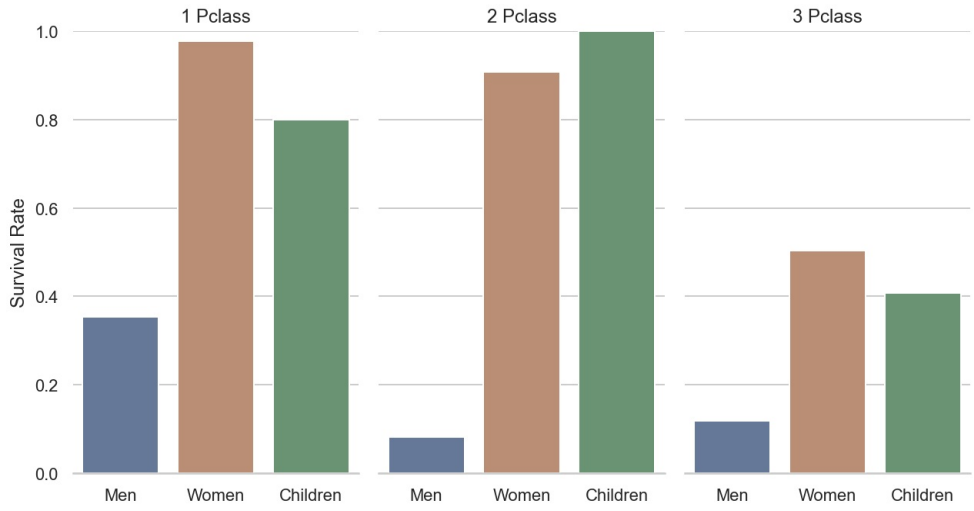
In [373…

```
g = sns.catplot(x="Sex", y="Survived", col="Pclass",
```

```
                data=df_train, saturation=.5,
                kind="bar", ci=None, aspect=.6)

(g.set_axis_labels("", "Survival Rate")
  .set_xticklabels(["Men", "Women", "Children"])
  .set_titles("{col_name} {col_var}")
  .set(ylim=(0, 1))
  .despine(left=True))
```

Out[373…

```
<seaborn.axisgrid.FacetGrid at 0x29682e07c40>
```



Here we could appreciate also that the influence of the class and genre is pretty important in this target value.

In [374…

```
# Lets fill the embarked column missin values with the mode
embarked_mode = df_train_categ['Embarked'].mode()
data = [df_train_categ]
for dataset in data:
    dataset['Embarked'] = dataset['Embarked'].fillna(embarked_mode)
```

Feature Engineering

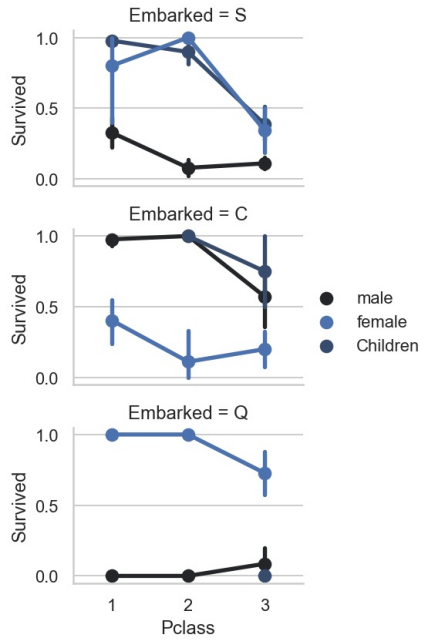
Let's do some feature engineering in order to extract some extra information of the available dataset

In [375…

```
FacetGrid = sns.FacetGrid(df_train, row='Embarked', size=2, aspect=1.6)
FacetGrid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', order=None, hue_order=None )
FacetGrid.add_legend()
```

Out[375…

```
<seaborn.axisgrid.FacetGrid at 0x2969b42a500>
```



Here we could see the influence of the class in the survival variable. Also here in the embarked variable we can appreciate that in Q C has

Here we could see the influence of the class in the survival variable. Also here in the embarked variable we can appreciate that in Q,C has more possibilities to survived than S.

- Female passengers have much better survival rate than males.
- males have better survşval rate in pclass 3 in C.
- embarked and sex will be used in training.

In [376…

```
# Visualize data characteristics
df_train.describe()
```

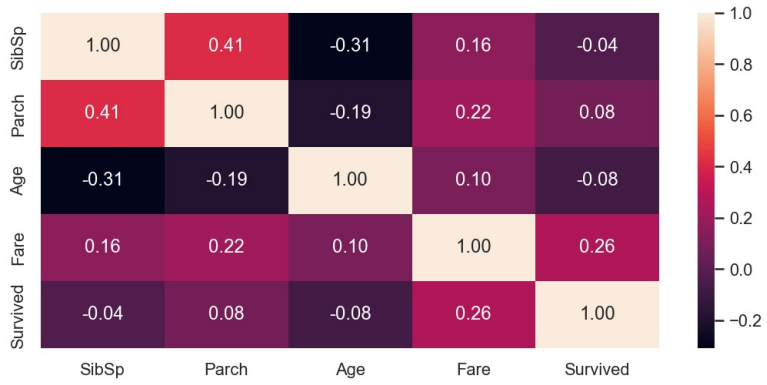
Out[376…

	Survived	Pclass	Age	SibSp	Parch	Fare	relatives
count	891.00	891.00	714.00	891.00	891.00	891.00	891.00
mean	0.38	2.31	29.70	0.52	0.38	32.20	0.90
std	0.49	0.84	14.53	1.10	0.81	49.69	1.61
min	0.00	1.00	0.42	0.00	0.00	0.00	0.00
25%	0.00	2.00	20.12	0.00	0.00	7.91	0.00
50%	0.00	3.00	28.00	0.00	0.00	14.45	0.00
75%	1.00	3.00	38.00	1.00	0.00	31.00	1.00
max	1.00	3.00	80.00	8.00	6.00	512.33	10.00

Data visualization

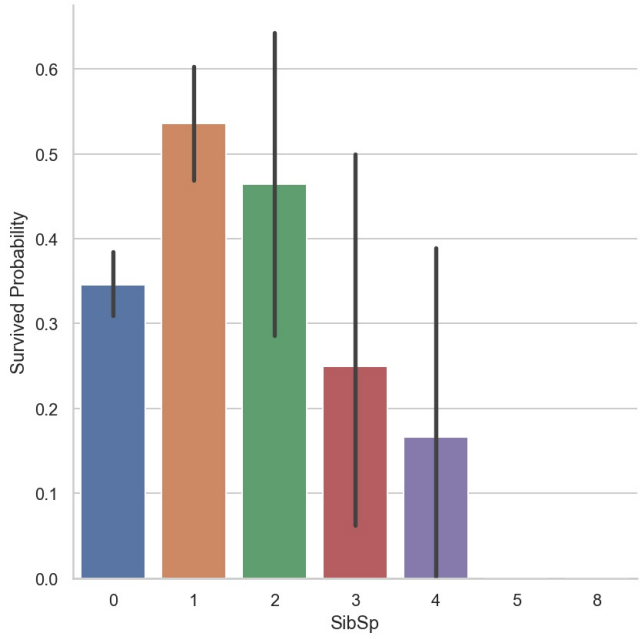
In [377…

```
list1 = ["SibSp", "Parch", "Age", "Fare", "Survived"]
sns.heatmap(df_train[list1].corr(), annot = True, fmt = ".2f")
plt.show()
```



In [378…

```
g = sns.factorplot(x = "SibSp", y = "Survived", data = df_train, kind = "bar", size = 6)
g.set_ylabels("Survived Probability")
plt.show()
```

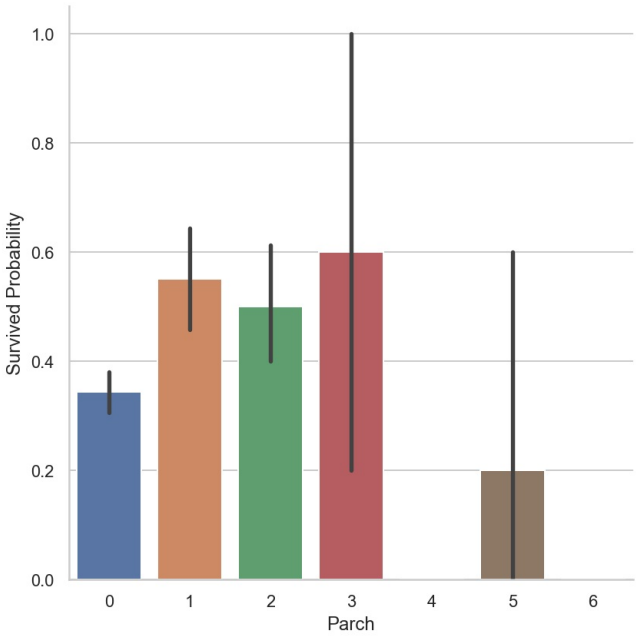


This graph confirm that apparently the number of relatives is not important.

- Having a lot of SibSp have less chance to survive.
- if sibsp == 0 or 1 or 2, passenger has more chance to survive
- we can consider a new feature describing these categories.

In [379…

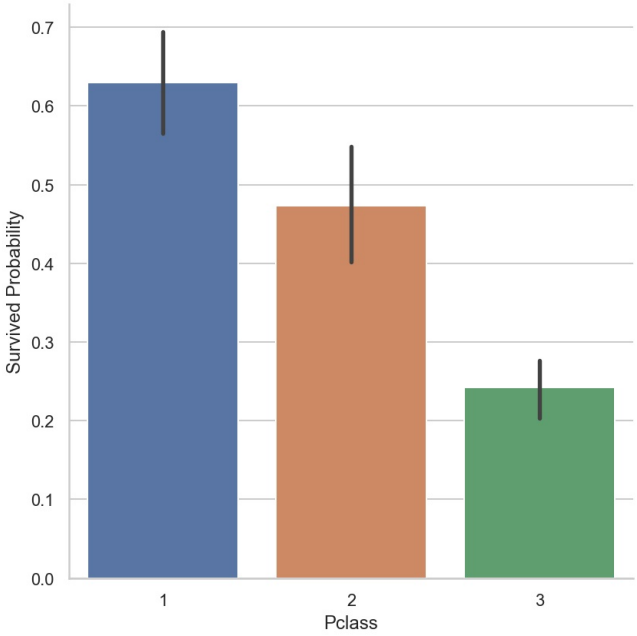
```
g = sns.factorplot(x = "Parch", y = "Survived", kind = "bar", data = df_train, size = 6)
g.set_ylabels("Survived Probability")
plt.show()
```



- Sibsp and parch can be used for new feature extraction with th = 3
- small familes have more chance to survive.
- there is a std in survival of passenger with parch = 3

In [380…

```
g = sns.factorplot(x = "Pclass", y = "Survived", data = df_train, kind = "bar", size = 6)
g.set_ylabels("Survived Probability")
plt.show()
```

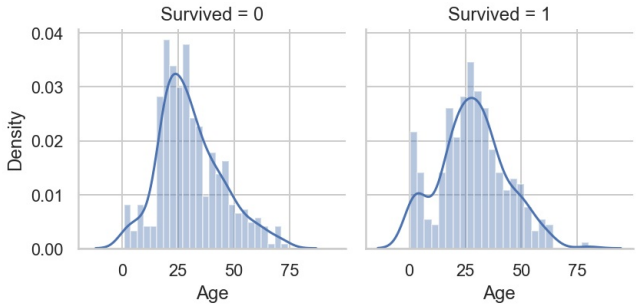


As I said previously, the class is pretty influent in the outcome situation for each person

In [381…

```
g = sns.FacetGrid(df_train, col = "Survived")
```

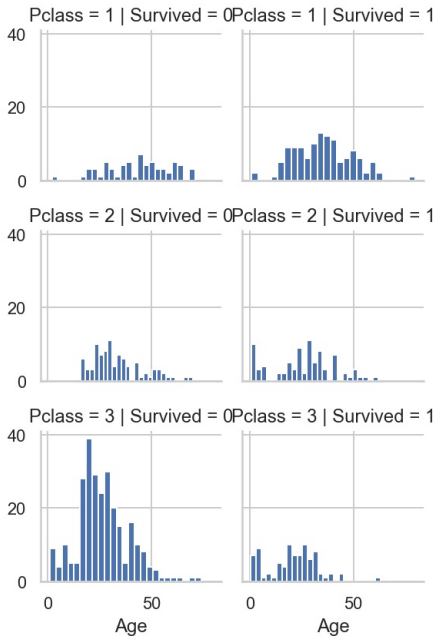
```
g = sns.FacetGrid(df_train, col = "Survived")
g.map(sns.distplot, "Age", bins = 25)
plt.show()
```



- age <= 10 has a high survival rate,
- oldest passengers (80) survived,
- large number of 20 years old did not survive,
- most passengers are in 15-35 age range,
- use age feature in training
- use age distribution for missing value of age

In [382…

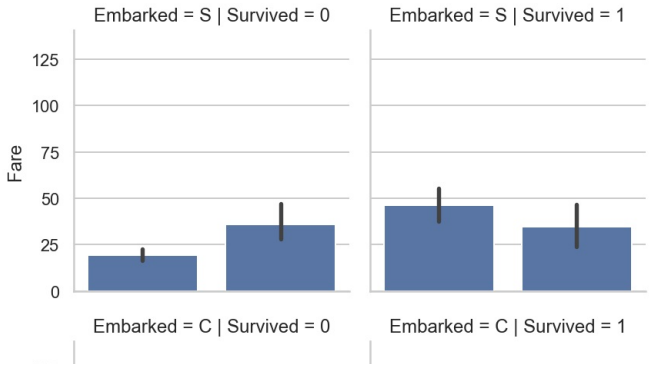
```
g = sns.FacetGrid(df_train, col = "Survived", row = "Pclass", size = 2)
g.map(plt.hist, "Age", bins = 25)
g.add_legend()
plt.show()
```

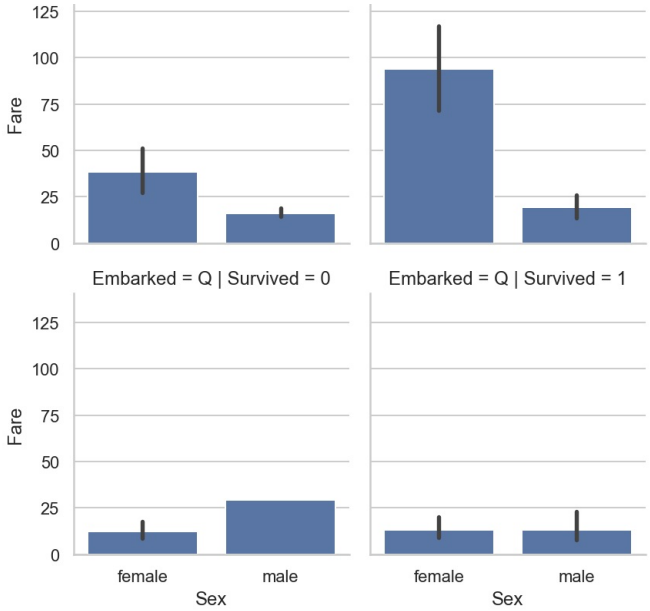


- pclass is important feature for model training.

In [383…

```
g = sns.FacetGrid(df_train, row = "Embarked", col = "Survived", size = 3)
g.map(sns.barplot, "Sex", "Fare")
g.add_legend()
plt.show()
```

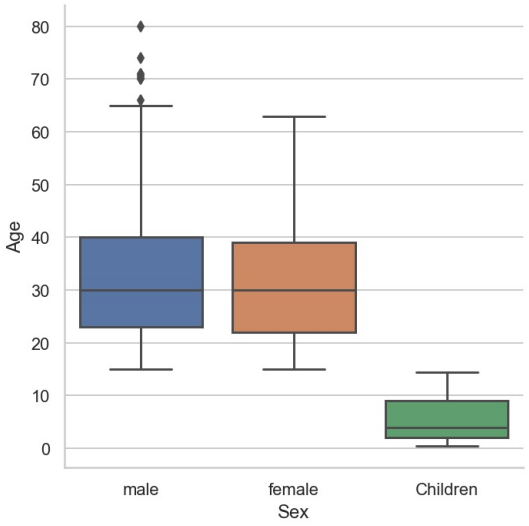




- Passsengers who pay higher fare have better survival. Fare can be used as categorical for training.

In [384…

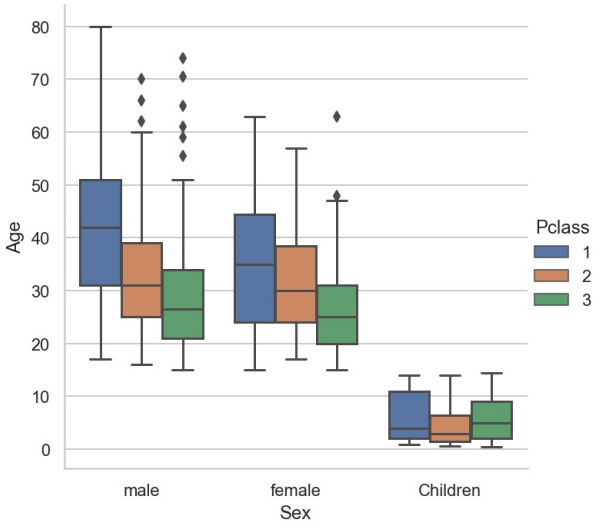
```
sns.factorplot(x = "Sex", y = "Age", data = df_train, kind = "box")
plt.show()
```



- Sex is not informative for age prediction, age distribution seems to be same.

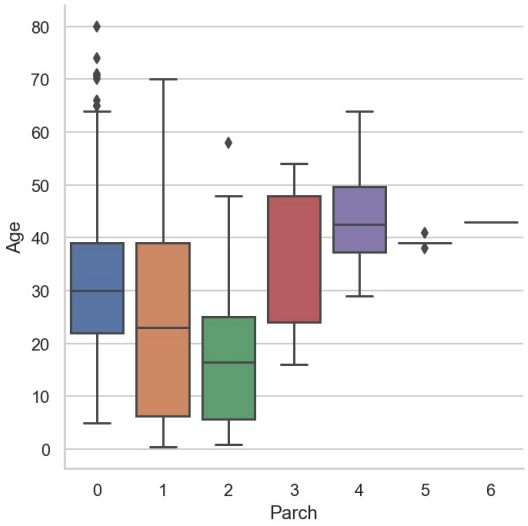
In [385…

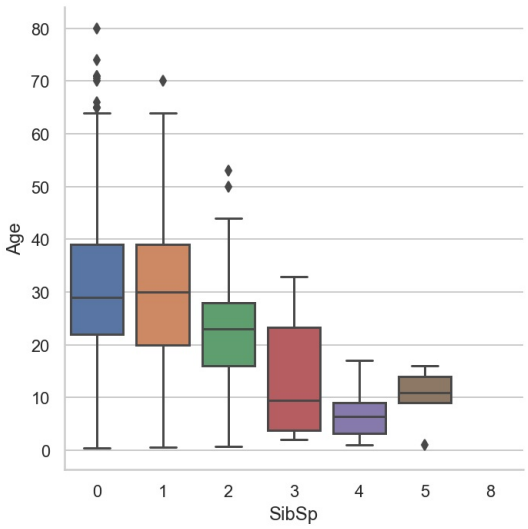
```
sns.factorplot(x = "Sex", y = "Age", hue = "Pclass",data = df_train, kind = "box")
plt.show()
```



- 1st class passengers are older than 2nd, and 2nd is older than 3rd class.

In [386…
```
sns.factorplot(x = "Parch", y = "Age", data = df_train, kind = "box")
sns.factorplot(x = "SibSp", y = "Age", data = df_train, kind = "box")
plt.show()
```





- Any relevant conclusion extracted from this plots.

In [387…
```
df_train["Name"].head(10)
```

Out[387…
```
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
```

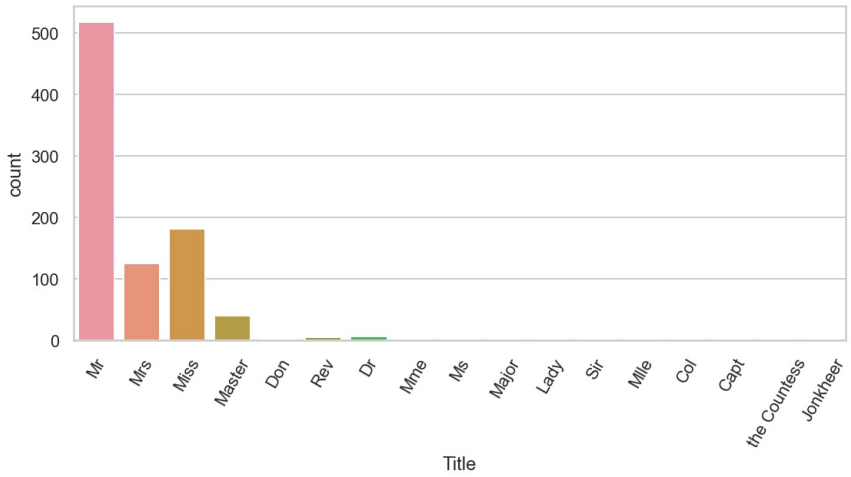
In [388…
```
name = df_train["Name"]
df_train["Title"] = [i.split(".")[0].split(",")[-1].strip() for i in name]
df_train["Title"].head(10)
```

Out[388…
```
0        Mr
1       Mrs
2      Miss
3       Mrs
```

```
4        Mr
5        Mr
6        Mr
7    Master
8       Mrs
9       Mrs
Name: Title, dtype: object
```

In [389…

```
sns.countplot(x="Title", data = df_train)
plt.xticks(rotation = 60)
plt.show()
```



In [390…

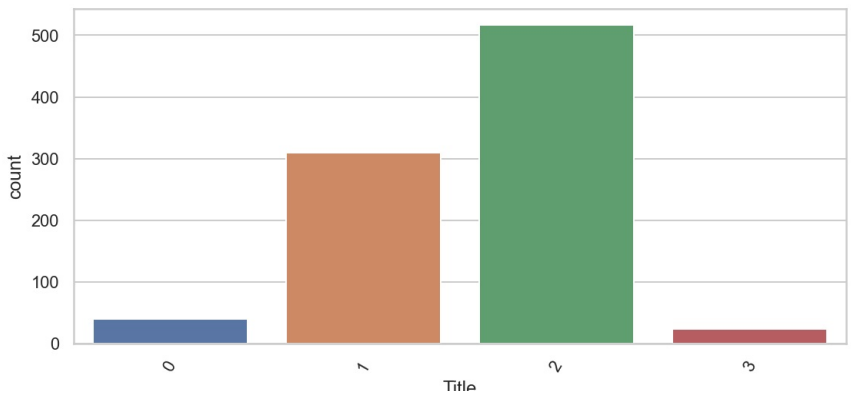
```
# convert to categorical
df_train["Title"] = df_train["Title"].replace(["Lady","the Countess","Capt","Col","Don","Dr","Major","Rev","Sir",
df_train["Title"] = [0 if i == "Master" else 1 if i == "Miss" or i == "Ms" or i == "Mlle" or i == "Mrs" else 2 if
df_train["Title"].head(20)
```

Out[390…

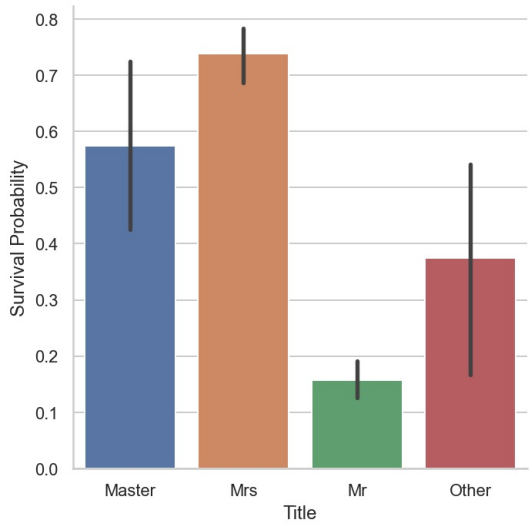
```
0     2
1     1
2     1
3     1
4     2
5     2
6     2
7     0
8     1
9     1
10    1
11    1
12    2
13    2
14    1
15    1
16    0
17    2
18    1
19    1
Name: Title, dtype: int64
```

In [391…

```
sns.countplot(x="Title", data = df_train)
plt.xticks(rotation = 60)
plt.show()
```



In [392…]
```
g = sns.factorplot(x = "Title", y = "Survived", data = df_train, kind = "bar")
g.set_xticklabels(["Master","Mrs","Mr","Other"])
g.set_ylabels("Survival Probability")
plt.show()
```



It's important this analysis to notice that women and people with high society titles has more preferences in the survived outcome.

In [393…]
```
# Drop names column now that we have extracted useful information
df_train.drop(labels = ["Name"], axis = 1, inplace = True)
```

In [394…]
```
train_df = pd.get_dummies(df_train,columns=["Title"])
train_df.head()
```

Out[394…]

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	relatives	travelled_alone	Title_0	Title_1	Title_2	Title_3
0	0	3	male	22.00	1	0	A/5 21171	7.25	NaN	S	1	No	0	0	1	0
1	1	1	female	38.00	1	0	PC 17599	71.28	C85	C	1	No	0	1	0	0
2	1	3	female	26.00	0	0	STON/O2. 3101282	7.92	NaN	S	0	Yes	0	1	0	0
3	1	1	female	35.00	1	0	113803	53.10	C123	S	1	No	0	1	0	0
4	0	3	male	35.00	0	0	373450	8.05	NaN	S	0	Yes	0	0	1	0

In [395…]
```
train_df = pd.get_dummies(train_df, columns=["Embarked"])
train_df.head()
```

Out[395…]

	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	relatives	travelled_alone	Title_0	Title_1	Title_2	Title_3	Embarked
0	0	3	male	22.00	1	0	A/5 21171	7.25	NaN	1	No	0	0	1	0	
1	1	1	female	38.00	1	0	PC 17599	71.28	C85	1	No	0	1	0	0	
2	1	3	female	26.00	0	0	STON/O2. 3101282	7.92	NaN	0	Yes	0	1	0	0	
3	1	1	female	35.00	1	0	113803	53.10	C123	1	No	0	1	0	0	
4	0	3	male	35.00	0	0	373450	8.05	NaN	0	Yes	0	0	1	0	

Let's handle the type of ticket that has more possibilities.

In [396…]
```
train_df["Ticket"].head(20)
```

Out[396…]
```
0           A/5 21171
1            PC 17599
2    STON/O2. 3101282
3              113803
4              373450
5              330877
```

```
6               17463
7              349909
8              347742
9              237736
10             PP 9549
11              113783
12           A/5. 2151
13              347082
14              350406
15              248706
16              382652
17              244373
18              345763
19                2649
Name: Ticket, dtype: object
```

In [397…

```
a = "A/5. 2151"
a.replace(".","").replace("/","").strip().split(" ")[0]
```

Out[397…

```
'A5'
```

In [398…

```
tickets = []
for i in list(train_df.Ticket):
    if not i.isdigit():
        tickets.append(i.replace(".","").replace("/","").strip().split(" ")[0])
    else:
        tickets.append("x")
train_df["Ticket"] = tickets
```

In [399…

```
train_df["Ticket"].head(20)
```

Out[399…

```
0         A5
1         PC
2     STONO2
3          x
4          x
5          x
6          x
7          x
8          x
9          x
10        PP
11         x
12        A5
13         x
14         x
15         x
16         x
17         x
18         x
19         x
Name: Ticket, dtype: object
```

In [400…

```
train_df = pd.get_dummies(train_df, columns= ["Ticket"], prefix = "T")
train_df.head(10)
```

Out[400…

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	relatives	travelled_alone	...	T_SOPP	T_SOTONO2	T_SOTONOQ	T_SP	T_S
0	0	3	male	22.00	1	0	7.25	NaN	1	No	...	0	0	0	0	
1	1	1	female	38.00	1	0	71.28	C85	1	No	...	0	0	0	0	
2	1	3	female	26.00	0	0	7.92	NaN	0	Yes	...	0	0	0	0	
3	1	1	female	35.00	1	0	53.10	C123	1	No	...	0	0	0	0	
4	0	3	male	35.00	0	0	8.05	NaN	0	Yes	...	0	0	0	0	
5	0	3	male	NaN	0	0	8.46	NaN	0	Yes	...	0	0	0	0	
6	0	1	male	54.00	0	0	51.86	E46	0	Yes	...	0	0	0	0	
7	0	3	Children	2.00	3	1	21.07	NaN	4	No	...	0	0	0	0	
8	1	3	female	27.00	0	2	11.13	NaN	2	No	...	0	0	0	0	
9	1	2	Children	14.00	1	0	30.07	NaN	1	No	...	0	0	0	0	

10 rows × 48 columns

In [401…

```
train_df["Pclass"] = train_df["Pclass"].astype("category")
train_df = pd.get_dummies(train_df, columns= ["Pclass"])
train_df.head()
```

Out[401…

	Survived	Sex	Age	SibSp	Parch	Fare	Cabin	relatives	travelled_alone	Title_0	...	T_SP	T_STONO	T_STONO2	T_SWPP	T_WC	T_
0	0	male	22.00	1	0	7.25	NaN	1	No	0	...	0	0	0	0	0	
1	1	female	38.00	1	0	71.28	C85	1	No	0	...	0	0	0	0	0	
2	1	female	26.00	0	0	7.92	NaN	0	Yes	0	...	0	0	1	0	0	
3	1	female	35.00	1	0	53.10	C123	1	No	0	...	0	0	0	0	0	
4	0	male	35.00	0	0	8.05	NaN	0	Yes	0	...	0	0	0	0	0	

5 rows × 50 columns

In [402…

```
train_df["Sex"] = train_df["Sex"].astype("category")
train_df = pd.get_dummies(train_df, columns=["Sex"])
train_df.head()
```

Out[402…

	Survived	Age	SibSp	Parch	Fare	Cabin	relatives	travelled_alone	Title_0	Title_1	...	T_SWPP	T_WC	T_WEP	T_x	Pclass_1	Pclass_
0	0	22.00	1	0	7.25	NaN	1	No	0	0	...	0	0	0	0	0	
1	1	38.00	1	0	71.28	C85	1	No	0	1	...	0	0	0	0	1	
2	1	26.00	0	0	7.92	NaN	0	Yes	0	1	...	0	0	0	0	0	
3	1	35.00	1	0	53.10	C123	1	No	0	1	...	0	0	0	1	1	
4	0	35.00	0	0	8.05	NaN	0	Yes	0	0	...	0	0	0	1	0	

5 rows × 52 columns

In [403…

```
# Drop cabin column because it not affect to the target variable
train_df.drop(labels = ["Cabin"], axis = 1, inplace = True)
```

Hyphothesis Testing

- Null Hypothesis(H0): There is no difference in the survival rate between the male and female passengers. or the mean difference between male and female passenger in the survival rate is zero.
- Alternative Hypothesis(HA): There is a difference in the survival rate between the male and female passengers. or the mean difference in the survival rate between male and female is not zero.

  One thing we can do is try to set up the Null and Alternative Hypothesis in such way that, when we do our t-test, we can choose to do one tailed test. According to this article, one-tailed tests are more powerful than two-tailed test. In addition to that, this video is also quite helpful understanding these topics. with this in mind we can update/modify our null and alternative hypothesis. Let's see how we can rewrite this..

Null Hypothesis(H0): male mean is greater or equal to female mean.

Alternative Hypothesis(H1): male mean is less than female mean.

Determine the test statistics: This will be a two-tailed test since the difference between male and female passenger in the survival rate could be higher or lower than 0. Since we do not know the standard deviation(σ) and n is small, we will use the t-distribution.

Specify the significance level: Specifying a significance level is an important step of the hypothesis test. It is an ultimate balance between type 1 error and type 2 error. We will discuss more in-depth about those in another lesson. For now, we have decided to make our significance level(α) = 0.05. So, our confidence interval or non-rejection region would be (1 - α)=(1-0.05) = 95%.

Computing T-statistics and P-value: Let's take a random sample and see the difference.

In [406…

```
male_mean = df_train[df_train['Sex'] == 'male'].Survived.mean()
female_mean = df_train[df_train['Sex'] == 'female'].Survived.mean()
print ("Male survival mean: " + str(male_mean))
print ("female survival mean: " + str(female_mean))

print ("The mean difference between male and female survival rate: " + str(female_mean - male_mean))
```

```
Male survival mean: 0.16356877323420074
female survival mean: 0.76
```

```
The mean difference between male and female survival rate: 0.5964312267657993
```

In [409…

```
# separating male and female dataframe.
import random
from statsmodels.stats.weightstats import ztest as ztest
male = df_train[df_train['Sex'] == 'male']
female = df_train[df_train['Sex'] == 'female']

## empty list for storing mean sample
m_mean_samples = []
f_mean_samples = []

for i in range(50):
    m_mean_samples.append(np.mean(random.sample(list(male['Survived']),50,)))
    f_mean_samples.append(np.mean(random.sample(list(female['Survived']),50,)))


# Print them out
print (f"Male mean sample mean: {round(np.mean(m_mean_samples),2)}")
print (f"Female mean sample mean: {round(np.mean(f_mean_samples),2)}")
print (f"Difference between male and female mean sample mean: {round(np.mean(f_mean_samples) - np.mean(m_mean_sam
```

```
Male mean sample mean: 0.16
Female mean sample mean: 0.76
Difference between male and female mean sample mean: 0.6
```

According to the samples our male samples ($\bar{x}m$) and female samples($\bar{x}f$) mean measured difference is ~ 0.6(statistically this is called the point estimate of the male population mean and female population mean). keeping in mind that...

- We randomly select 50 people to be in the male group and 50 people to be in the female group.
- We know our sample is selected from a broader population(trainning set).
- We know we could have totally ended up with a different random sample of males and females.

In [415…

```
#perform two sample z-test
ztest(male['Survived'], female['Survived'], value=0.1)
```

Out[415…

```
(-24.04918087261463, 8.514134674211484e-128)
```

As we can see, the null hypothesis is not true, so that our test determine to get the alternative hypothesis, which tell us that the mean on survived woman is bigger than mean of survived man

Conclusion

To sum up, we have detected the most important variables and its main level of influence ibn the target variable.

- Class: Its infuence is high
- Sex: Its infuence is high
- Fare: Its infuence is high
- Relatives: Its infuence is sigh
- Ticket: Its infuence is a little bit undetermined
- Age: Its infuence is medium because as we have seen children have preferences and also age is in relation with class
- Title: Its infuence is medium because as we habe seen the master title have more preference in survived status
- Embarked: Appear to have a sigh influence in outcome.
- Cabin, ID: No influence