Topic 2. Visual data analysis

Practice. Analyzing "Titanic" passengers

Fill in the missing code ("You code here").

Competition Kaggle "Titanic: Machine Learning from Disaster".

```
import numpy as np
import pandas as pd
import seaborn as sns

sns.set()
import matplotlib.pyplot as plt
```

Read data

```
train_df = pd.read_csv("titanic_train.csv", index_col="PassengerId")
```

```
train_df.head(2)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC	71.2833	C85	C

```
train_df.describe(include="all")
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	Dooley, Mr. Patrick	male	NaN	NaN	NaN	347082	NaN	G6	S
freq	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN
max	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Name      891 non-null    object
 3   Sex       891 non-null    object
 4   Age       714 non-null    float64
 5   SibSp     891 non-null    int64
 6   Parch     891 non-null    int64
 7   Ticket    891 non-null    object
 8   Fare      891 non-null    float64
 9   Cabin     204 non-null    object
 10  Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

Let's drop `Cabin` , and then – all rows with missing values.

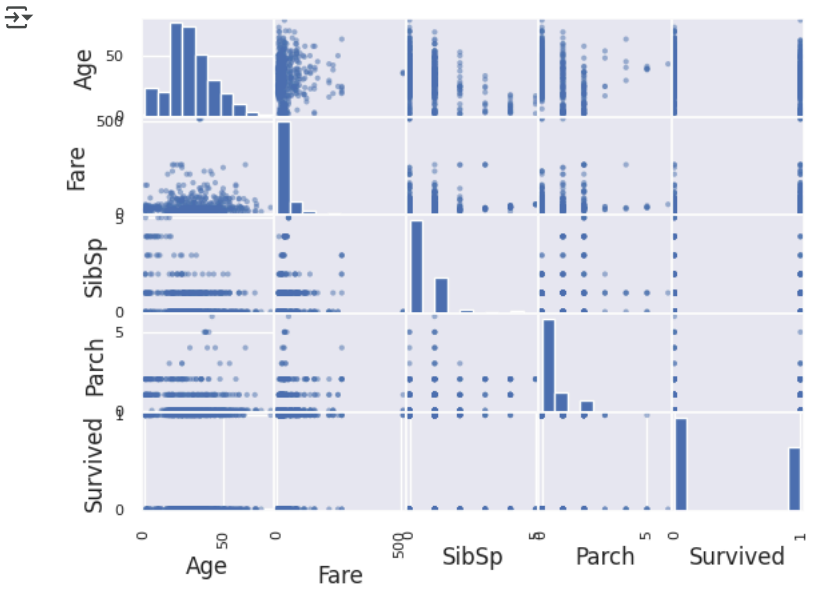
```
train_df = train_df.drop("Cabin", axis=1).dropna()
```

```
train_df.shape
```

```
(712, 10)
```

1. Build a picture to visualize all scatter plots for each pair of features `Age`, `Fare`, `SibSp`, `Parch` and `Survived`. (`scatter_matrix` from Pandas or `pairplot` from Seaborn)

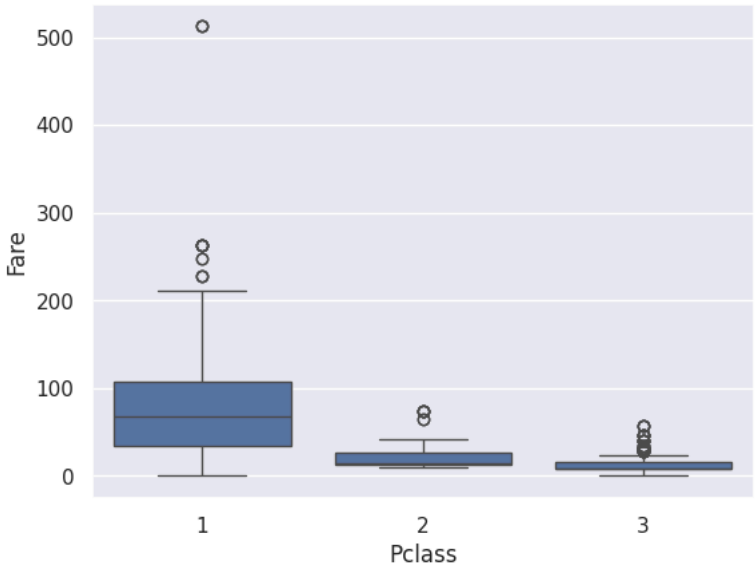
```
from pandas.plotting import scatter_matrix

features = ['Age', 'Fare', 'SibSp', 'Parch', 'Survived']
scatter_matrix(train_df[features])
plt.show()
```



2. How does ticket price (`Fare`) depend on `Pclass` ? Build a boxplot.

```
sns.boxplot(x="Pclass", y="Fare", data=train_df)
```

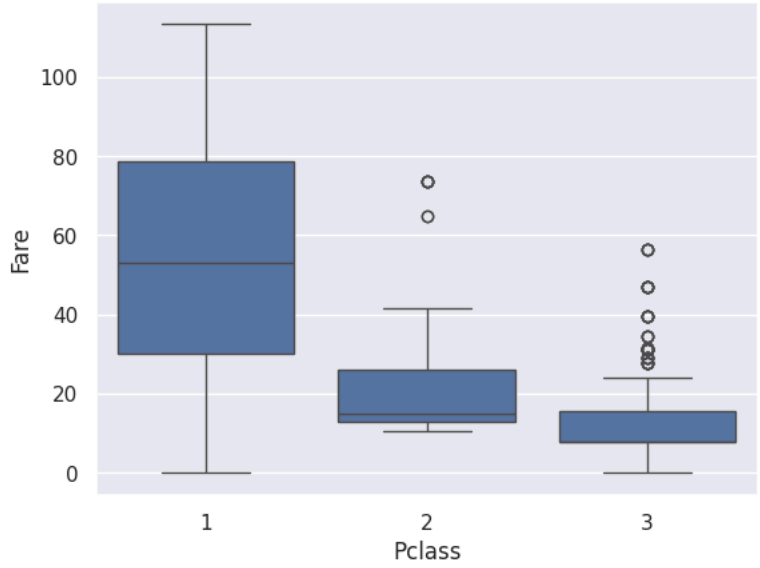
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



3. Let's build the same plot but restricting values of `Fare` to be less than 95% quantile of the initial vector (to drop outliers that make the plot less clear).

```
train_df_95_quantile = train_df[train_df["Fare"] < train_df["Fare"].quantile(0.95)]
sns.boxplot(x="Pclass", y="Fare", data=train_df_95_quantile)
```

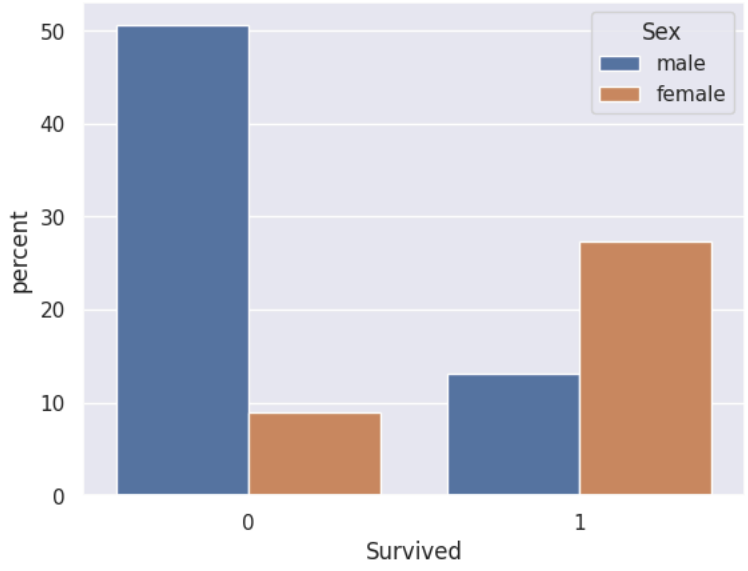
```
<Axes: xlabel='Pclass', ylabel='Fare'>
```



4. How is the percentage of surviving passengers dependent on passengers' gender? Depict it with `Seaborn.countplot` using the `hue` argument.

```
sns.countplot(train_df, x="Survived", hue="Sex", stat="percent")
```

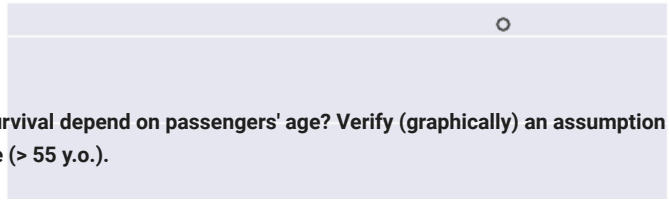
```
<Axes: xlabel='Survived', ylabel='percent'>
```



5. How does the distribution of ticket prices differ for those who survived and those who didn't. Depict it with `Seaborn.boxplot`

```
sns.boxplot(x="Survived", y="Fare", data=train_df)
```

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



6. How does survival depend on passengers' age? Verify (graphically) an assumption that youngsters (< 30 y.o.) survived more frequently than old people (> 55 y.o.).

```
def get_age_category(age: int) -> str:
    if age < 30:
        return "<30"
    elif age > 55:
        return ">55"
    else:
        return "30-55"
```

```
age_category = [get_age_category(age) for age in train_df.Age]
train_df["Age category"] = age_category
```

```
sns.countplot(train_df, x="Survived", hue="Age category")
```

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