Preprocessing_Daily_Task

December 5, 2024

[288]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Datasets already available inside seaborn
# sns.get_dataset_names()

data = sns.load_dataset('titanic')
data
```

[288]:

```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0           0       3    male  22.0      1      0   7.2500        S   Third
1           1       1  female  38.0      1      0  71.2833        C   First
2           1       3  female  26.0      0      0   7.9250        S   Third
3           1       1  female  35.0      1      0  53.1000        S   First
4           0       3    male  35.0      0      0   8.0500        S   Third
..        …       …       …     …      …      …        …        …       …
886         0       2    male  27.0      0      0  13.0000        S  Second
887         1       1  female  19.0      0      0  30.0000        S   First
888         0       3  female   NaN      1      2  23.4500        S   Third
889         1       1    male  26.0      0      0  30.0000        C   First
890         0       3    male  32.0      0      0   7.7500        Q   Third

       who  adult_male deck  embark_town alive  alone
0      man        True  NaN  Southampton    no  False
1    woman       False    C    Cherbourg   yes  False
2    woman       False  NaN  Southampton   yes   True
3    woman       False    C  Southampton   yes  False
4      man        True  NaN  Southampton    no   True
..     …          …    …          …     …      …
886    man        True  NaN  Southampton    no   True
887  woman       False    B  Southampton   yes   True
888  woman       False  NaN  Southampton    no  False
889    man        True    C    Cherbourg   yes   True
890    man        True  NaN   Queenstown    no   True
```

```
[891 rows x 15 columns]
```

[289]:
```
data.dtypes
```

[289]:
```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

[290]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

[291]:
```
data.describe()
```

[291]:

```
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

[292]:

```
# Converting yes/no in 'alive' column to binary
data['alive'].replace({'yes':1,'no':0},inplace=True)
data['alive'] = data['alive'].astype('int64')
```

```
/var/folders/jw/jny11qx97778yjjc7534g4bm0000gn/T/ipykernel_58882/3730205506.py:2
: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  data['alive'].replace({'yes':1,'no':0},inplace=True)
/var/folders/jw/jny11qx97778yjjc7534g4bm0000gn/T/ipykernel_58882/3730205506.py:2
: FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  data['alive'].replace({'yes':1,'no':0},inplace=True)
```

[293]:

```
data['class'].unique()
```

[293]:

```
['Third', 'First', 'Second']
Categories (3, object): ['First', 'Second', 'Third']
```

[294]:

```
# Converting textual info in 'class' column to numerical values
data['class'].replace({'First':1,'Second':2, 'Third':3},inplace=True)
data['class'] = data['class'].astype('int64')
```

```
/var/folders/jw/jny11qx97778yjjc7534g4bm0000gn/T/ipykernel_58882/4177372890.py:2
: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.
```

```
For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  data['class'].replace({'First':1,'Second':2, 'Third':3},inplace=True)
/var/folders/jw/jny11qx97778yjjc7534g4bm0000gn/T/ipykernel_58882/4177372890.py:2
: FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  data['class'].replace({'First':1,'Second':2, 'Third':3},inplace=True)
/var/folders/jw/jny11qx97778yjjc7534g4bm0000gn/T/ipykernel_58882/4177372890.py:2
: FutureWarning: The behavior of Series.replace (and DataFrame.replace) with
CategoricalDtype is deprecated. In a future version, replace will only be used
for cases that preserve the categories. To change the categories, use
ser.cat.rename_categories instead.
  data['class'].replace({'First':1,'Second':2, 'Third':3},inplace=True)
```

[295]:

```
data.dtypes
```

[295]:

```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class             int64
who              object
adult_male         bool
deck           category
embark_town      object
alive             int64
alone              bool
dtype: object
```

[296]:

```
# Dropping duplicate columns
data = data.iloc[:,2:]
```

[297]:

```
# Dropping duplicate rows
data.drop_duplicates(inplace=True)
# dup_rows = data.duplicated()
# data = data[~dup_rows]
```

[298]:

```
data
```

[298]:

```
        sex   age  sibsp  parch     fare embarked  class    who  adult_male  \
0      male  22.0      1      0   7.2500        S      3    man        True
1    female  38.0      1      0  71.2833        C      1  woman       False
2    female  26.0      0      0   7.9250        S      3  woman       False
3    female  35.0      1      0  53.1000        S      1  woman       False
4      male  35.0      0      0   8.0500        S      3    man        True
..      …   …    …    …      …      …    …    …         …
885  female  39.0      0      5  29.1250        Q      3  woman       False
887  female  19.0      0      0  30.0000        S      1  woman       False
888  female   NaN      1      2  23.4500        S      3  woman       False
889    male  26.0      0      0  30.0000        C      1    man        True
890    male  32.0      0      0   7.7500        Q      3    man        True

    deck  embark_town  alive  alone
0    NaN  Southampton      0  False
1      C    Cherbourg      1  False
2    NaN  Southampton      1   True
3      C  Southampton      1  False
4    NaN  Southampton      0   True
..   …          …    …    …
885  NaN   Queenstown      0  False
887    B  Southampton      1   True
888  NaN  Southampton      0  False
889    C    Cherbourg      1   True
890  NaN   Queenstown      0   True

[784 rows x 13 columns]
```

[299]:

```
# count of missing values
data.isna().sum()
```

[299]:

```
sex              0
age            106
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           582
embark_town      2
alive            0
alone            0
dtype: int64
```

[300]:

```
data.dtypes
```

[300]:

```
sex               object
age              float64
sibsp              int64
parch              int64
fare             float64
embarked          object
class              int64
who               object
adult_male          bool
deck            category
embark_town       object
alive              int64
alone               bool
dtype: object
```

[301]:

```python
num_data = data.select_dtypes(include="number")
cat_data = data.select_dtypes(exclude="number")
```

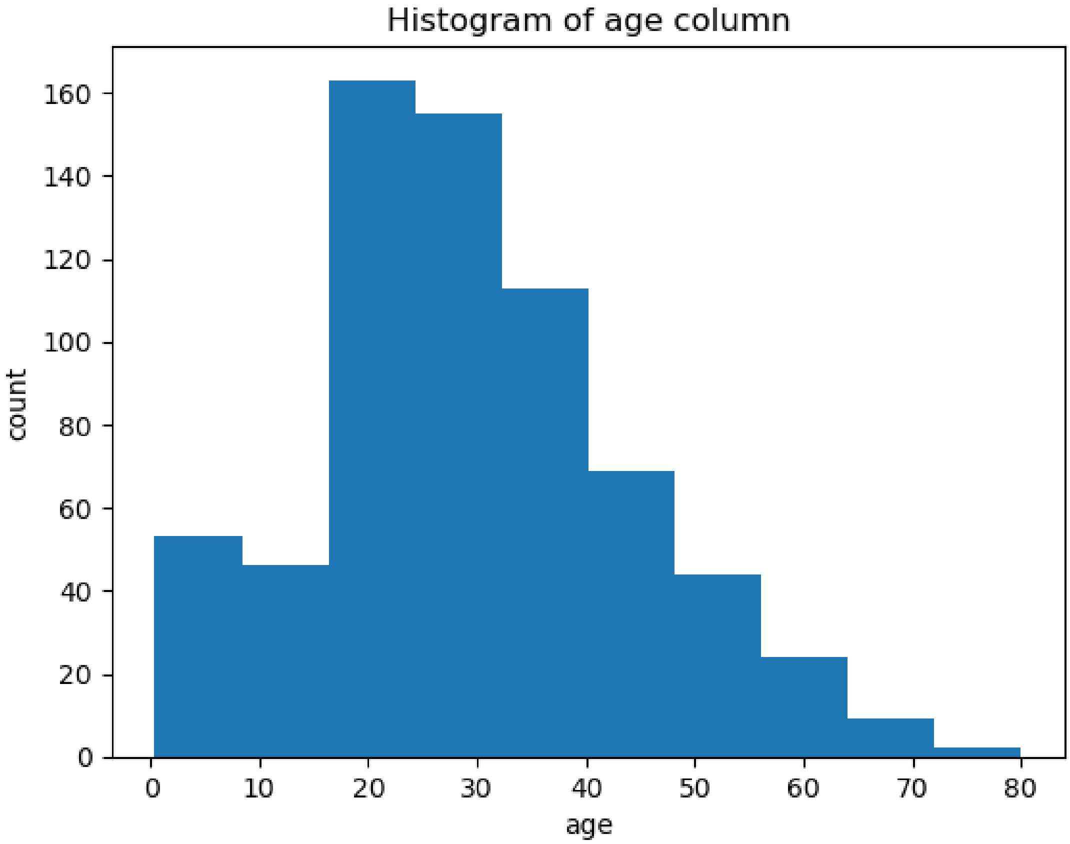
[302]:

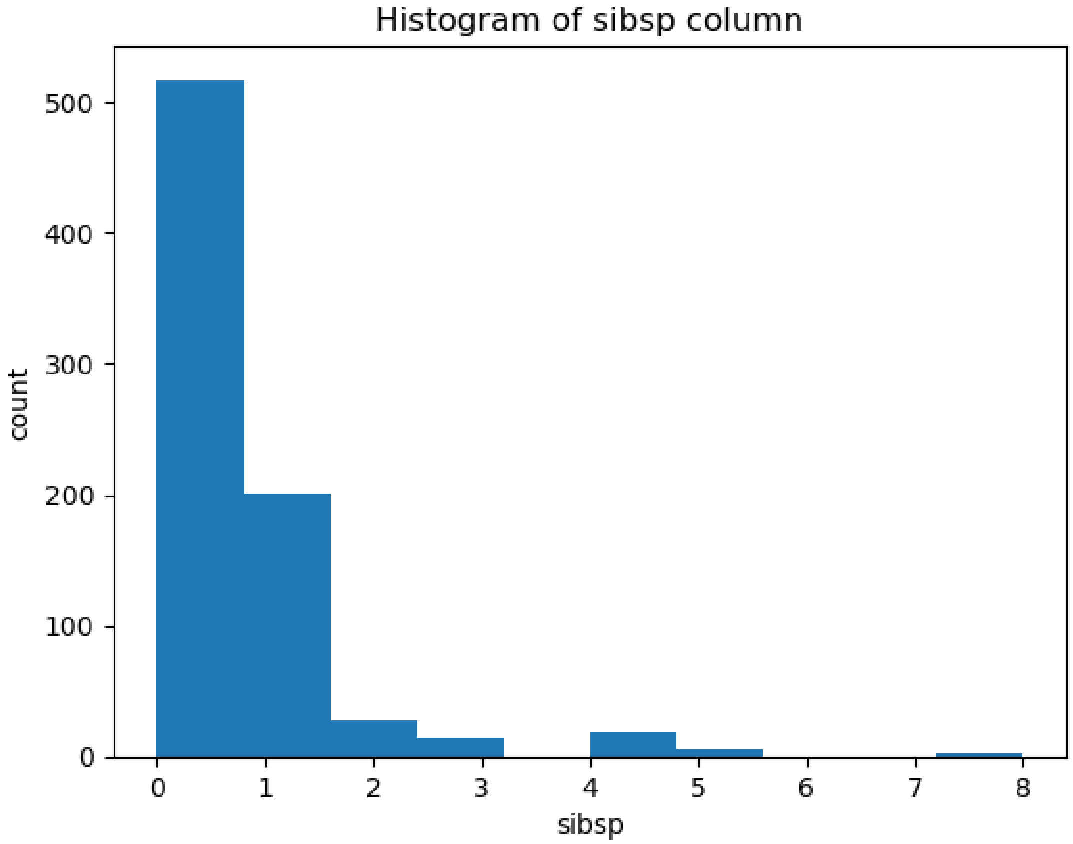
```python
print("Numerical columns are: ")
num_cols = num_data.columns.tolist()
print(num_cols)

print("\nCategorical columns are: ")
cat_cols = cat_data.columns.tolist()
print(cat_cols)
```

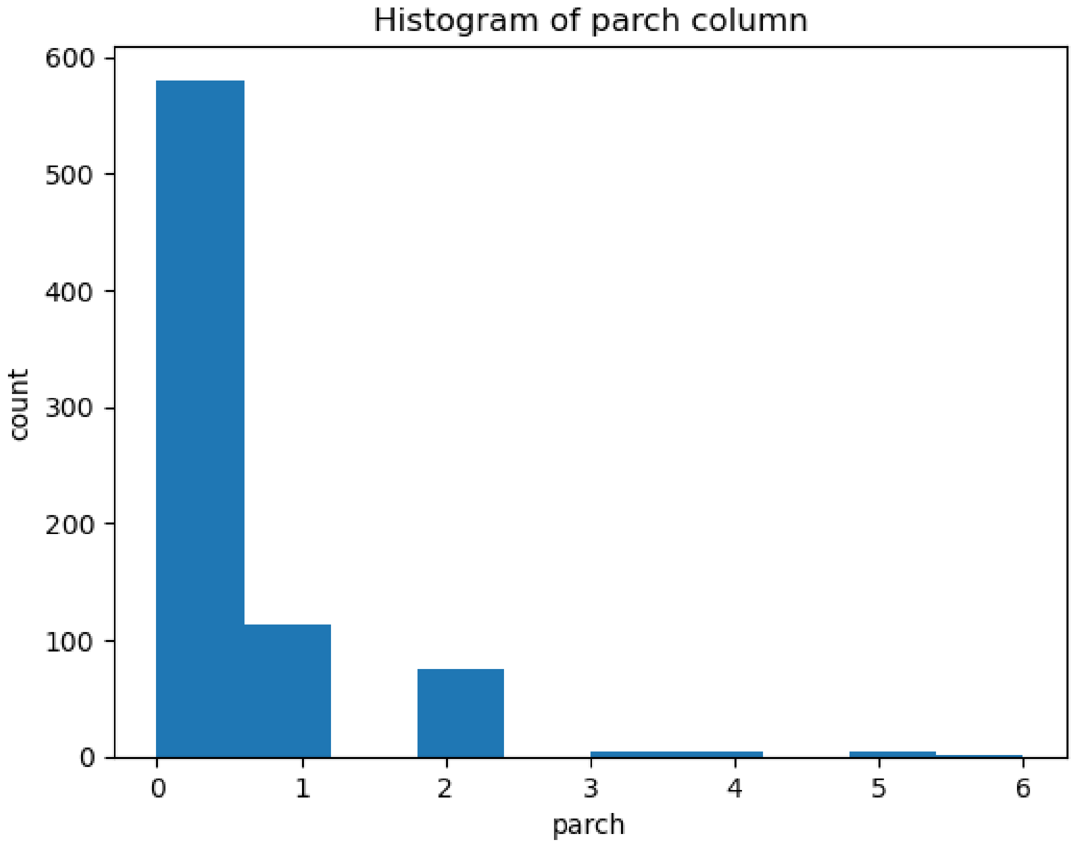
```
Numerical columns are:
['age', 'sibsp', 'parch', 'fare', 'class', 'alive']

Categorical columns are:
['sex', 'embarked', 'who', 'adult_male', 'deck', 'embark_town', 'alone']
```

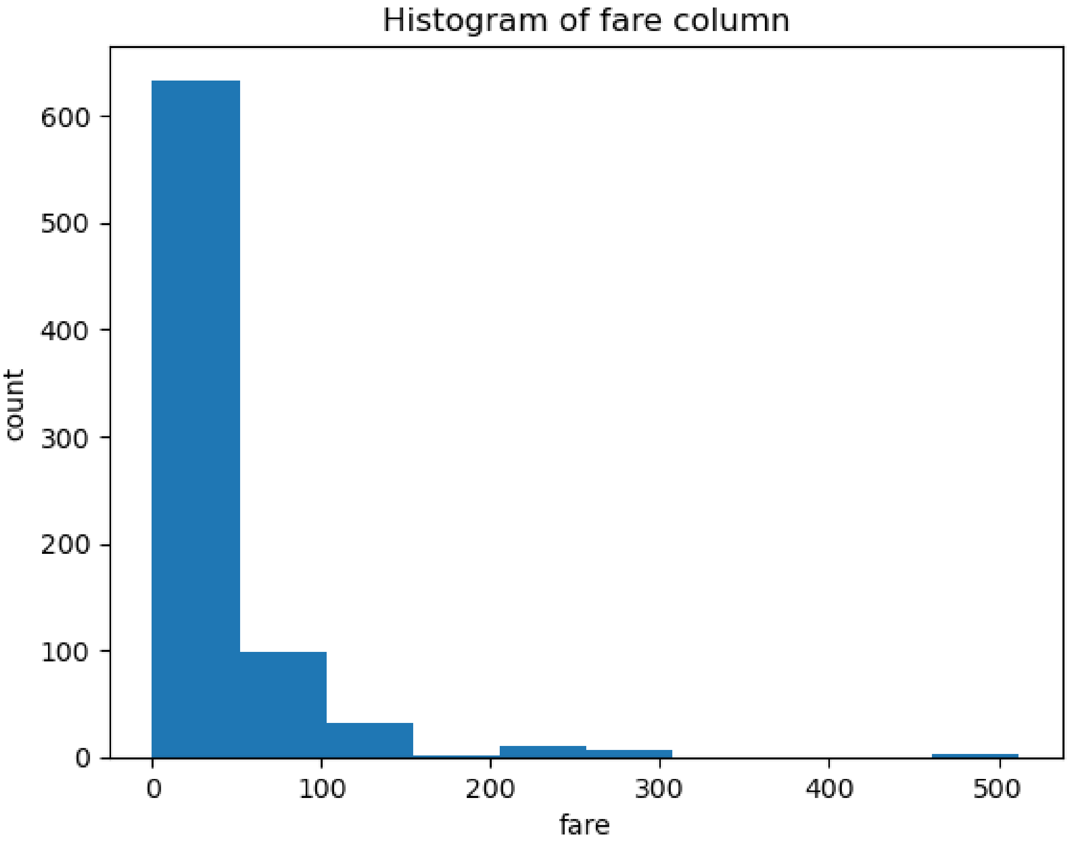
[303]:

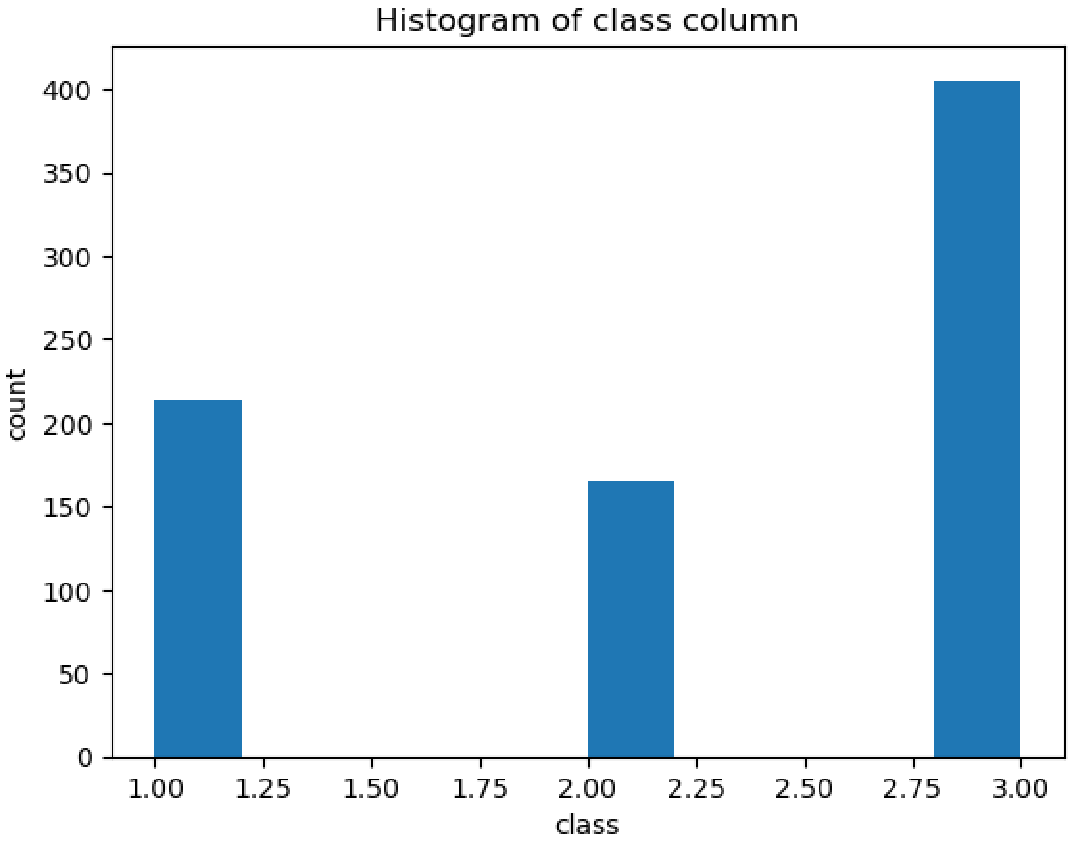
```python
for col in num_cols:
    plt.hist(data[col])
    plt.title("Histogram of {} column".format(col))
    plt.xlabel(col)
    plt.ylabel("count")
    plt.show()
```

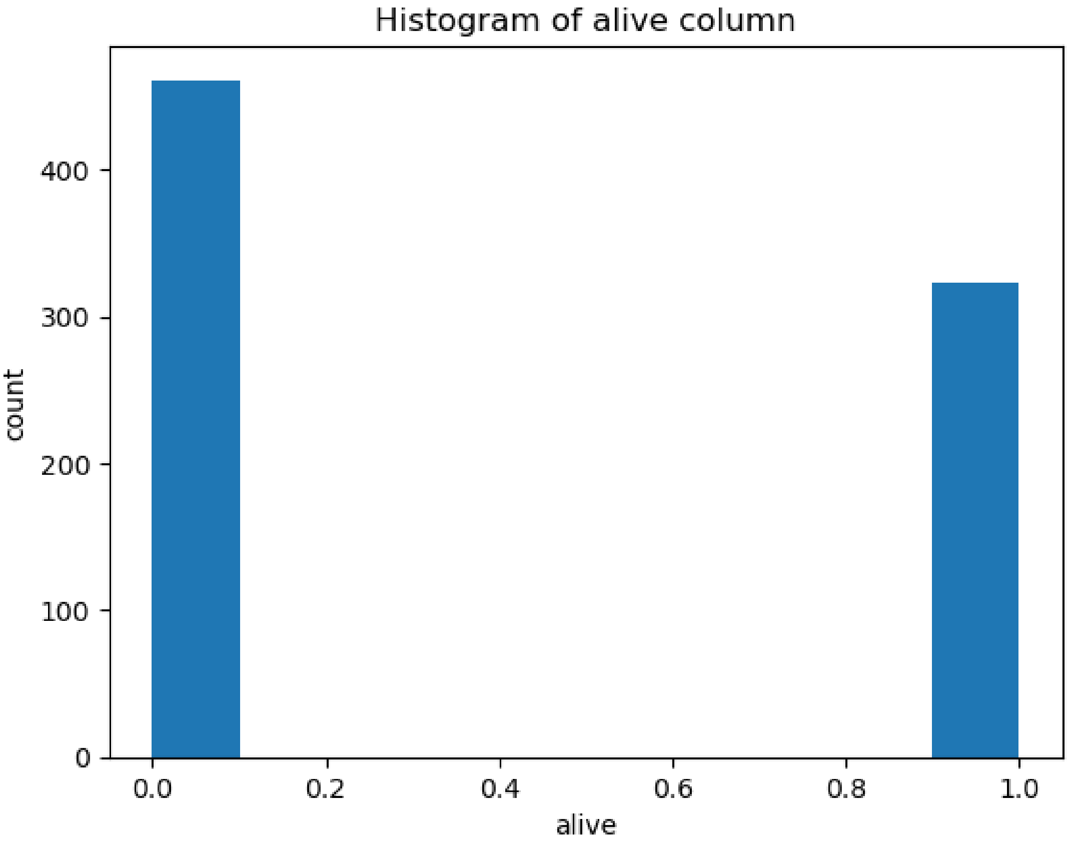

Histogram of age column
count
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
60
70
80
age


Histogram of sibsp column
count
0
100
200
300
400
500
sibsp
0
1
2
3
4
5
6
7
8


Histogram of parch column
count
600
500
400
300
200
100
0
0
1
2
3
4
5
6
parch


Histogram of fare column
count
600
500
400
300
200
100
0
0
100
200
300
400
500
fare


Histogram of class column
count
400
350
300
250
200
150
100
50
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
class



```
[304]: for col in num_cols:
           if (col=='alive' or col=='class'):
               data[col] = data[col].fillna(data[col].mode())
           elif (col=='age'):
               data[col] = data[col].fillna(data[col].mean())
           else:
               data[col] = data[col].fillna(data[col].median())
```

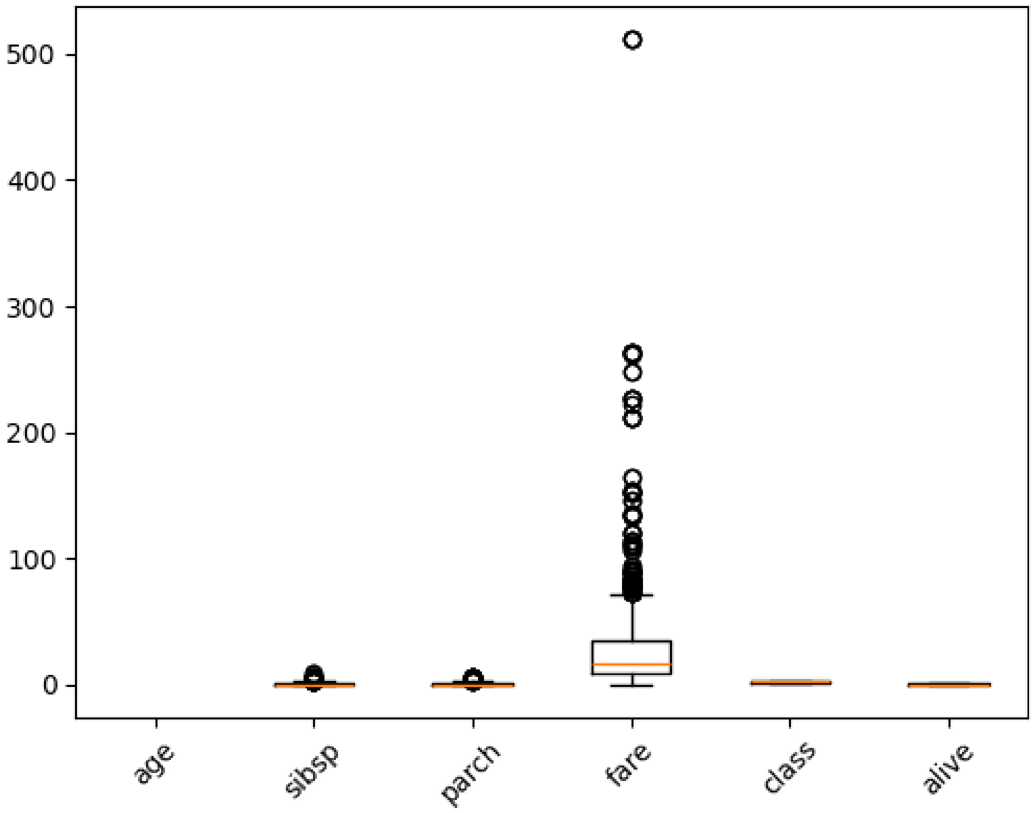
```
[305]: for col in cat_cols:
           data[col] = data[col].fillna(data[col].mode()[0])
```

```
[306]: data.isna().sum()
```

```
[306]: sex           0
       age           0
       sibsp         0
       parch         0
       fare          0
       embarked      0
       class         0
```

```
who            0
adult_male     0
deck           0
embark_town    0
alive          0
alone          0
dtype: int64
```

[307]:
```
plt.boxplot(num_data)
plt.xticks([1, 2, 3, 4,5,6], num_cols, rotation=45)
plt.show()
```

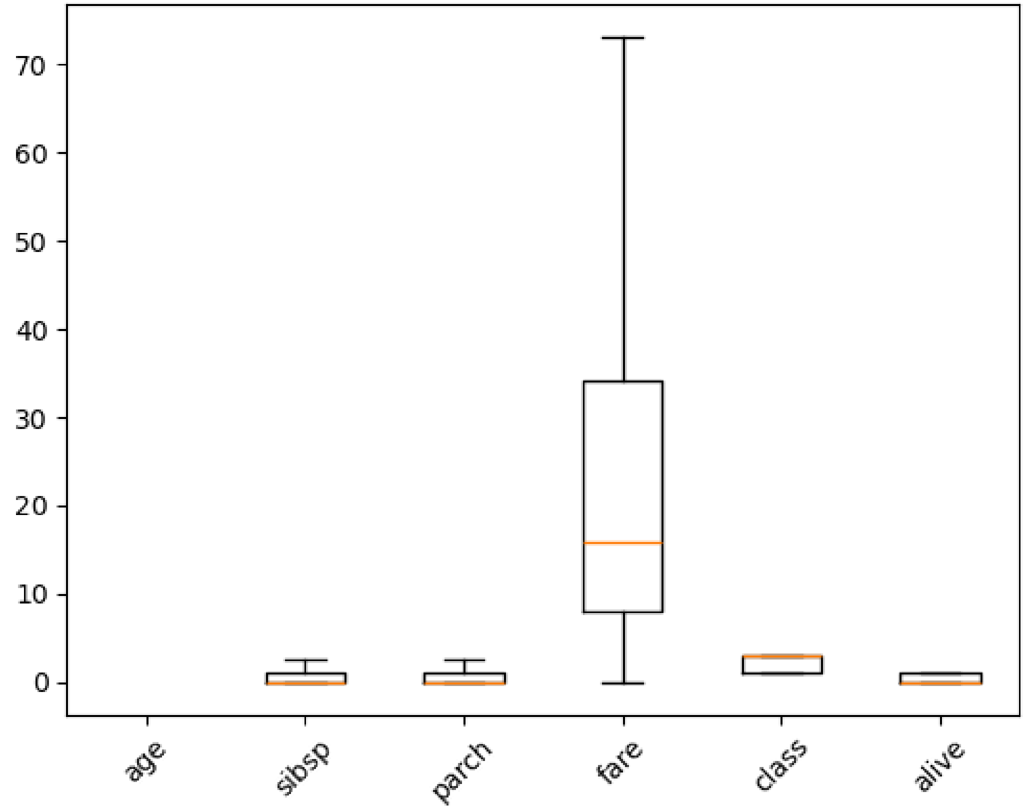


[308]:
```
def remove_outliers(df, column_name):
    q1 = df[column_name].quantile(0.25)
    q3 = df[column_name].quantile(0.75)
    iqr = q3-q1
    lower_bound = q1 - 1.5*iqr
    upper_bound = q3 + 1.5*iqr
    df[column_name] = df[column_name].clip(upper=upper_bound)
    df[column_name] = df[column_name].clip(lower=lower_bound)
    return df[column_name]
```

[311]:
```
for col in num_cols:
    data[col] = remove_outliers(data, col)
```

[313]:
```
for col in num_cols:
    num_data[col] = remove_outliers(num_data, col)
```

[314]:
```
plt.boxplot(num_data)
plt.xticks([1, 2, 3, 4,5,6], num_cols, rotation=45)
plt.show()
```



[315]:
```
data
```

[315]:
```
         sex        age  sibsp  parch     fare embarked  class    who  \
0       male  22.000000    1.0    0.0   7.2500        S      3    man
1     female  38.000000    1.0    0.0  71.2833        C      1  woman
2     female  26.000000    0.0    0.0   7.9250        S      3  woman
3     female  35.000000    1.0    0.0  53.1000        S      1  woman
4       male  35.000000    0.0    0.0   8.0500        S      3    man
..      …        …    …    …      …      …    …    …
```

```
885  female  39.000000   0.0    2.5  29.1250         Q       3  woman
887  female  19.000000   0.0    0.0  30.0000         S       1  woman
888  female  29.869351   1.0    2.0  23.4500         S       3  woman
889    male  26.000000   0.0    0.0  30.0000         C       1    man
890    male  32.000000   0.0    0.0   7.7500         Q       3    man

     adult_male deck  embark_town  alive  alone
0          True    C  Southampton      0  False
1         False    C    Cherbourg      1  False
2         False    C  Southampton      1   True
3         False    C  Southampton      1  False
4          True    C  Southampton      0   True
..          …   …          …    …     …
885       False    C   Queenstown      0  False
887       False    B  Southampton      1   True
888       False    C  Southampton      0  False
889        True    C    Cherbourg      1   True
890        True    C   Queenstown      0   True

[784 rows x 13 columns]
```

[318]:

```
data['embark_town'].unique()
```

[318]:

```
array(['Southampton', 'Cherbourg', 'Queenstown'], dtype=object)
```

[325]:

```
label_encoding = LabelEncoder()
data['sex'] = label_encoding.fit_transform(data['sex'])
data['embarked'] = label_encoding.fit_transform(data['embarked'])
data['who'] = label_encoding.fit_transform(data['who'])
data['deck'] = label_encoding.fit_transform(data['deck'])
data
```

[325]:

```
     sex        age  sibsp  parch     fare  embarked  class  who  adult_male  \
0      1  22.000000    1.0    0.0   7.2500         2      3    1        True
1      0  38.000000    1.0    0.0  71.2833         0      1    2       False
2      0  26.000000    0.0    0.0   7.9250         2      3    2       False
3      0  35.000000    1.0    0.0  53.1000         2      1    2       False
4      1  35.000000    0.0    0.0   8.0500         2      3    1        True
..   …         …    …     …      …       …   …        …
885    0  39.000000    0.0    2.5  29.1250         1      3    2       False
887    0  19.000000    0.0    0.0  30.0000         2      1    2       False
888    0  29.869351    1.0    2.0  23.4500         2      3    2       False
889    1  26.000000    0.0    0.0  30.0000         0      1    1        True
890    1  32.000000    0.0    0.0   7.7500         1      3    1        True

     deck  embark_town  alive  alone
0       2  Southampton      0  False
```

```
1        2    Cherbourg      1  False
2        2  Southampton      1   True
3        2  Southampton      1  False
4        2  Southampton      0   True
..    …          …    …    …
885      2   Queenstown      0  False
887      1  Southampton      1   True
888      2  Southampton      0  False
889      2    Cherbourg      1   True
890      2   Queenstown      0   True

[784 rows x 13 columns]
```

[328]:

```
data['adult_male'] = data['adult_male'].astype('int')
data['alone'] = data['alone'].astype('int')
data
```

[328]:

```
     sex        age  sibsp  parch     fare  embarked  class  who  adult_male  \
0      1  22.000000    1.0    0.0   7.2500         2      3    1           1
1      0  38.000000    1.0    0.0  71.2833         0      1    2           0
2      0  26.000000    0.0    0.0   7.9250         2      3    2           0
3      0  35.000000    1.0    0.0  53.1000         2      1    2           0
4      1  35.000000    0.0    0.0   8.0500         2      3    1           1
..   …        …    …    …      …       …    …  …         …
885    0  39.000000    0.0    2.5  29.1250         1      3    2           0
887    0  19.000000    0.0    0.0  30.0000         2      1    2           0
888    0  29.869351    1.0    2.0  23.4500         2      3    2           0
889    1  26.000000    0.0    0.0  30.0000         0      1    1           1
890    1  32.000000    0.0    0.0   7.7500         1      3    1           1

     deck  embark_town  alive  alone
0       2  Southampton      0      0
1       2    Cherbourg      1      0
2       2  Southampton      1      1
3       2  Southampton      1      0
4       2  Southampton      0      1
..    …          …    …    …
885     2   Queenstown      0      0
887     1  Southampton      1      1
888     2  Southampton      0      0
889     2    Cherbourg      1      1
890     2   Queenstown      0      1

[784 rows x 13 columns]
```

[331]:

```
town_onehot_encoding = pd.
 ↪get_dummies(data,columns=['embark_town'],prefix='et',dtype='int')
```

```
town_onehot_encoding
```

[331]:
```
     sex        age  sibsp  parch     fare  embarked  class  who  adult_male  \
0      1  22.000000    1.0    0.0   7.2500         2      3    1           1
1      0  38.000000    1.0    0.0  71.2833         0      1    2           0
2      0  26.000000    0.0    0.0   7.9250         2      3    2           0
3      0  35.000000    1.0    0.0  53.1000         2      1    2           0
4      1  35.000000    0.0    0.0   8.0500         2      3    1           1
..   …        …    …    …      …       …    …  …         …
885    0  39.000000    0.0    2.5  29.1250         1      3    2           0
887    0  19.000000    0.0    0.0  30.0000         2      1    2           0
888    0  29.869351    1.0    2.0  23.4500         2      3    2           0
889    1  26.000000    0.0    0.0  30.0000         0      1    1           1
890    1  32.000000    0.0    0.0   7.7500         1      3    1           1

     deck  alive  alone  et_Cherbourg  et_Queenstown  et_Southampton
0       2      0      0             0              0               1
1       2      1      0             1              0               0
2       2      1      1             0              0               1
3       2      1      0             0              0               1
4       2      0      1             0              0               1
..    …    …    …           …            …             …
885     2      0      0             0              1               0
887     1      1      1             0              0               1
888     2      0      0             0              0               1
889     2      1      1             1              0               0
890     2      0      1             0              1               0

[784 rows x 15 columns]
```

[ ]:
```

```