Titanic data set cleaning, visulaization and simple logestic regression predection

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

```
df=sns.load_dataset("titanic")
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Next steps: Generate code with df New interactive sheet

EDA

```
df.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
```

```
df.describe()
```

	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

```
df.isnull().sum()
```

	0
survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0

dtype: int64

```
df["age"].fillna(df["age"].mean(),inplace=True)
df=df.drop("deck",axis=1)
df=df.dropna()
df.isnull().sum()
```

```
/tmp/ipython-input-674/1813444921.py:1: 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


  df["age"].fillna(df["age"].mean(),inplace=True)
```

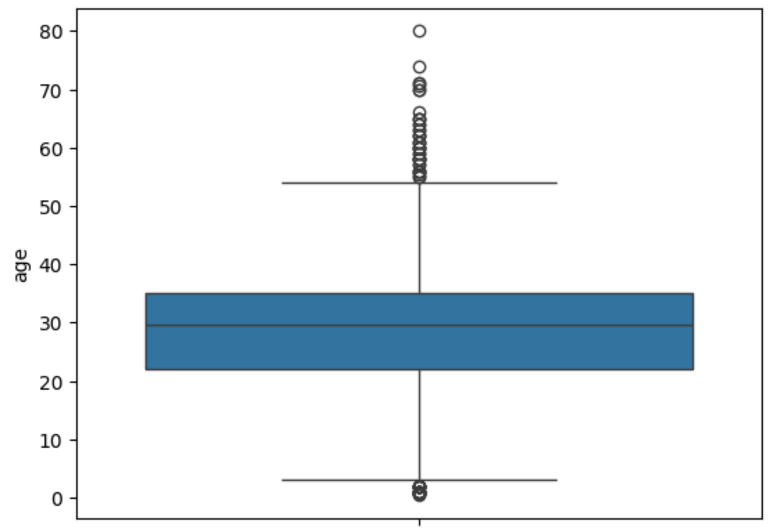
```
              0
survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare          0
embarked      0
class         0
who           0
adult_male    0
embark_town   0
alive         0
alone         0

dtype: int64
```

```
def lab(x):
  if x>0:
    return True
  else :
    return False
df["survived"]=df["survived"].apply(lab)
```

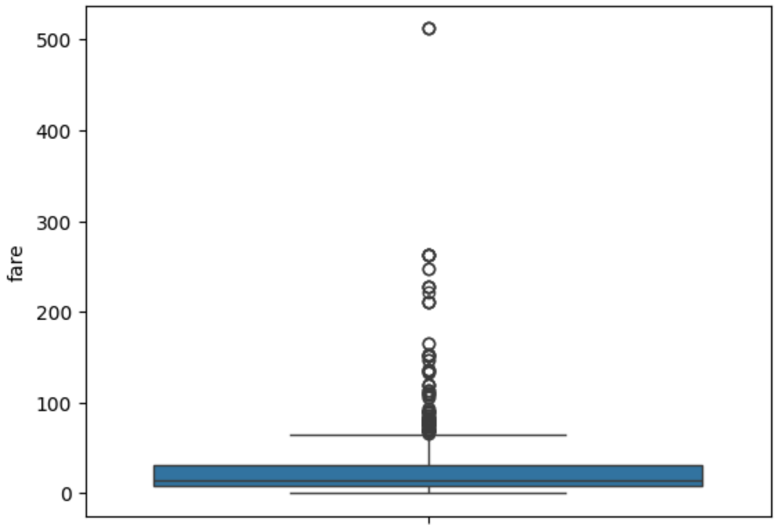
```
sns.boxplot(df["age"])
```

```
<Axes: ylabel='age'>
```



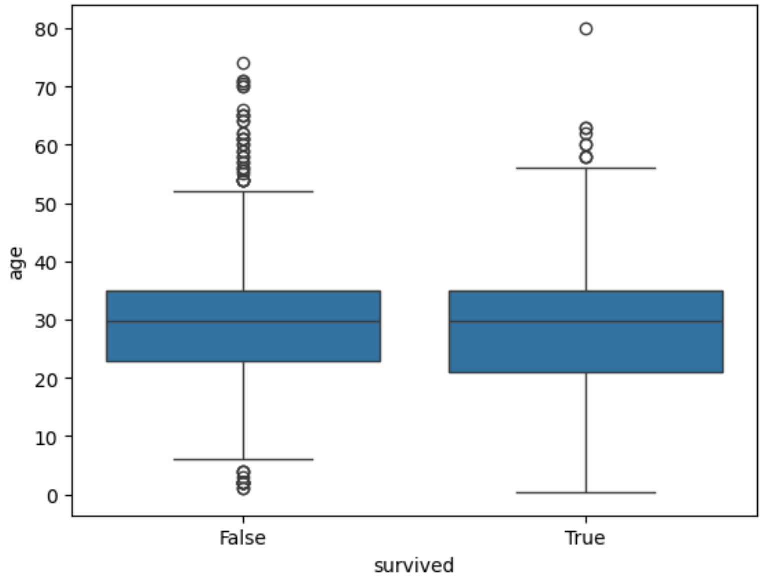
```
sns.boxplot(df["fare"])
```

```
<Axes: ylabel='fare'>
```



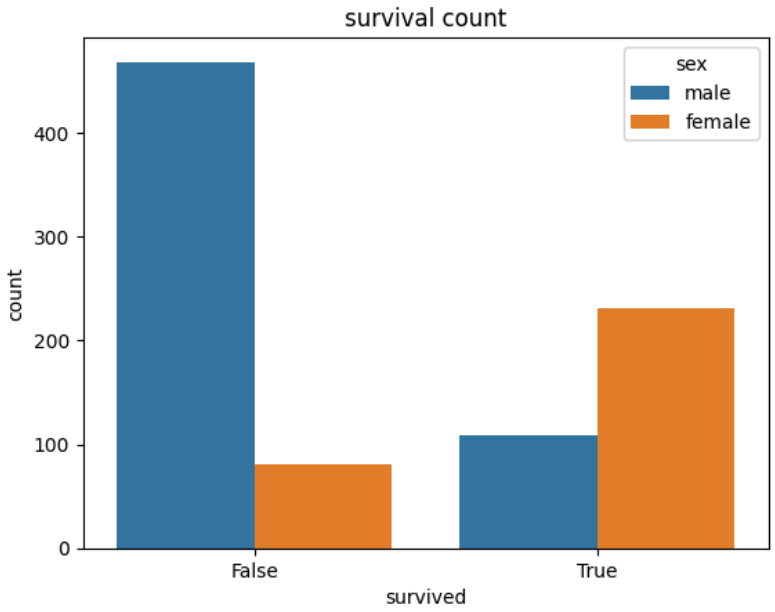
```
sns.boxplot(x="survived",y="age",data=df)
```

```
<Axes: xlabel='survived', ylabel='age'>
```



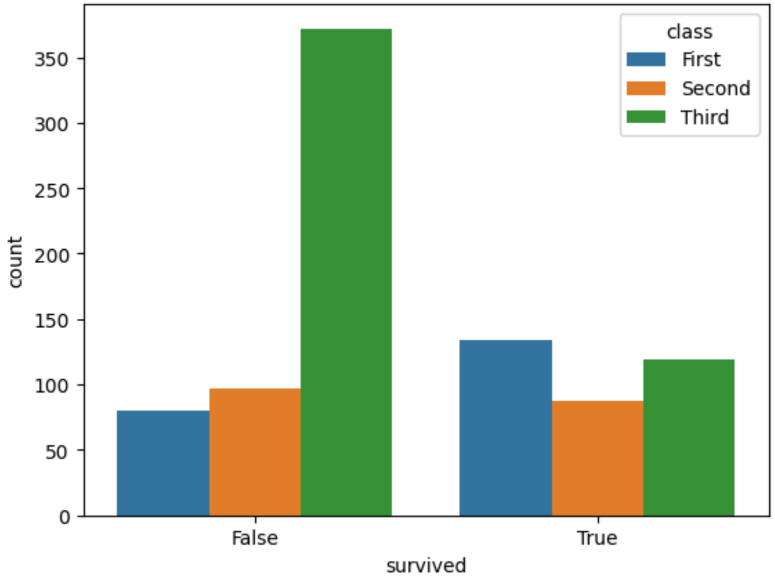
```
sns.countplot(x="survived",hue="sex",data=df)
plt.title("survival count")
```

```
Text(0.5, 1.0, 'survival count')
```



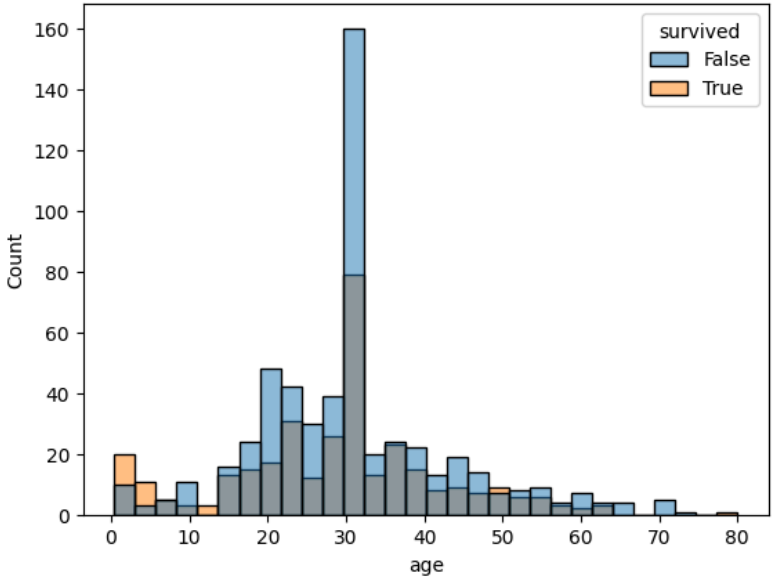
```
sns.countplot(x="survived",hue="class",data=df)
```

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



```
sns.histplot(x="age",hue="survived",data=df)
```

```
<Axes: xlabel='age', ylabel='Count'>
```



Logestic regression model

```
label_encoder=LabelEncoder()
```

```
# Target
y = df["survived"]

# Features
X = df[["pclass", "sex", "age", "fare", "embarked"]]
```

```
X["sex"]=label_encoder.fit_transform(X["sex"])
X = pd.get_dummies(X, columns=["embarked"], drop_first=True)
```

```
/tmp/ipython-input-674/941690834.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X["sex"]=label_encoder.fit_transform(X["sex"])
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
model=LogisticRegression()
```

```
model.fit(X_test,y_test)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8033707865168539
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

       False       0.82      0.86      0.84       109
        True       0.77      0.71      0.74        69

    accuracy                           0.80       178
   macro avg       0.80      0.79      0.79       178
weighted avg       0.80      0.80      0.80       178
```