

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, cm
import warnings
warnings.filterwarnings("ignore")
```

## Importing Data set

```
In [2]: df=pd.read_csv("citicanic_train.csv")
df.head()
```

```
Out[2]:
```

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

## EDA

```
In [3]: df.describe()
```

```
Out[3]:
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
In [4]: df["Survived"].value_counts() # Data is unbalanced
```

```
Out[4]:
```

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

```
In [5]: df.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(3), object(3)
memory usage: 66.2+ KB
```

From above we observed that, there are some missing values in Age,Cabin, Embarked columns, If there are more numbers of missing values (<50%) we should drop that feature otherwise replace/impute mean values.

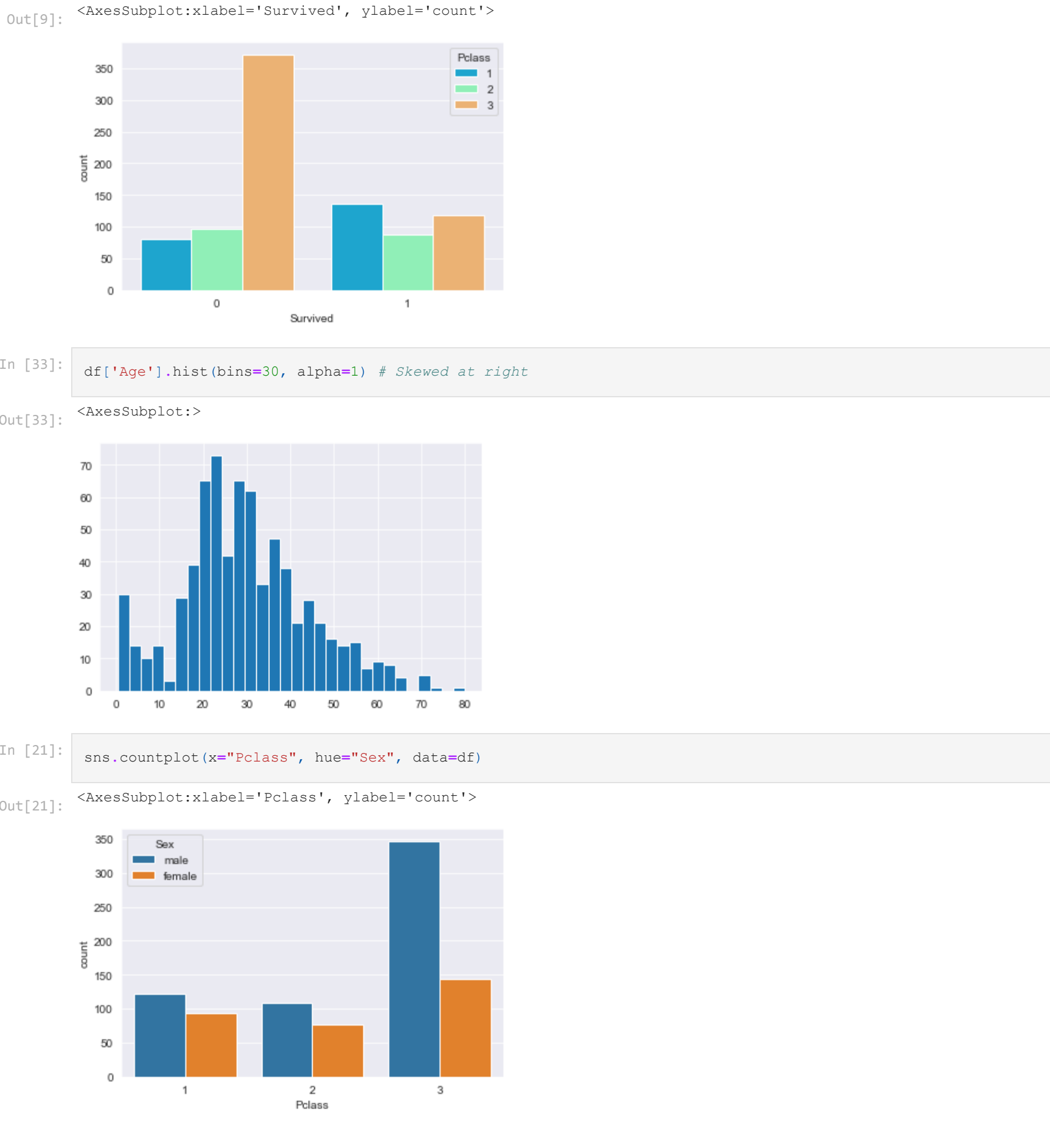
```
In [6]: sns.heatmap(df.isnull(),xticklabels=True,cbar=False,cmap='viridis')

<AxesSubplot:>
```

```
In [7]: df=df.drop(["Cabin"],axis=1)
```

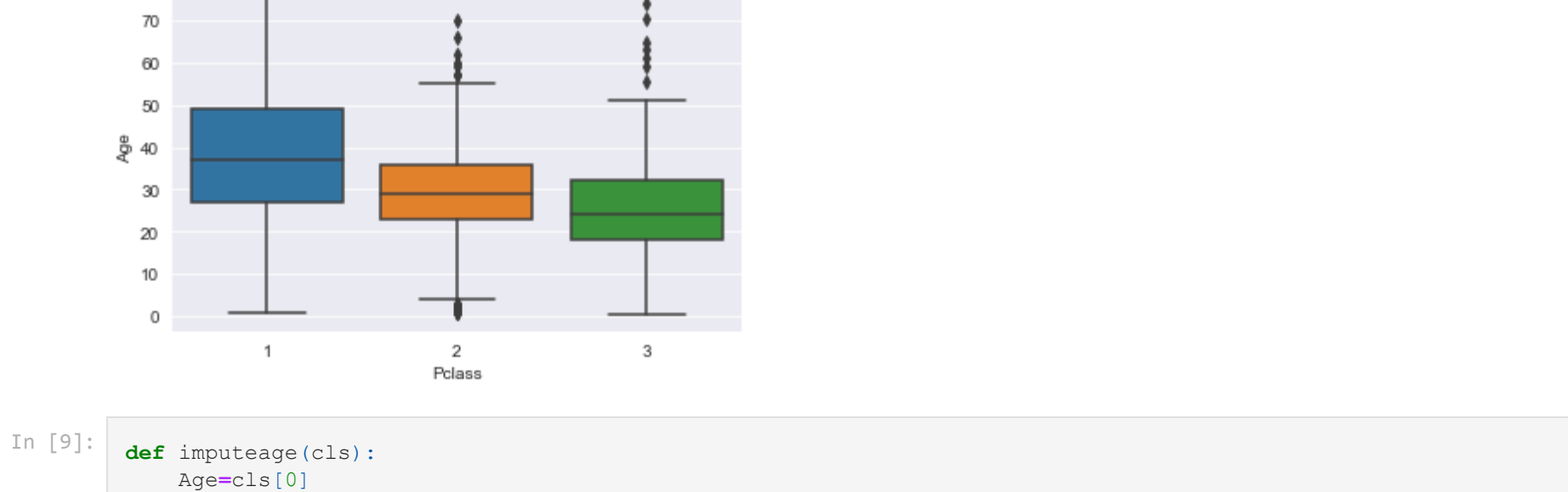
```
In [43]: sns.pairplot(df,hue="Survived")

<seaborn.axisgrid.PairGrid at 0x981340>
```



```
In [7]: sns.countplot(data=df,x="Survived")

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



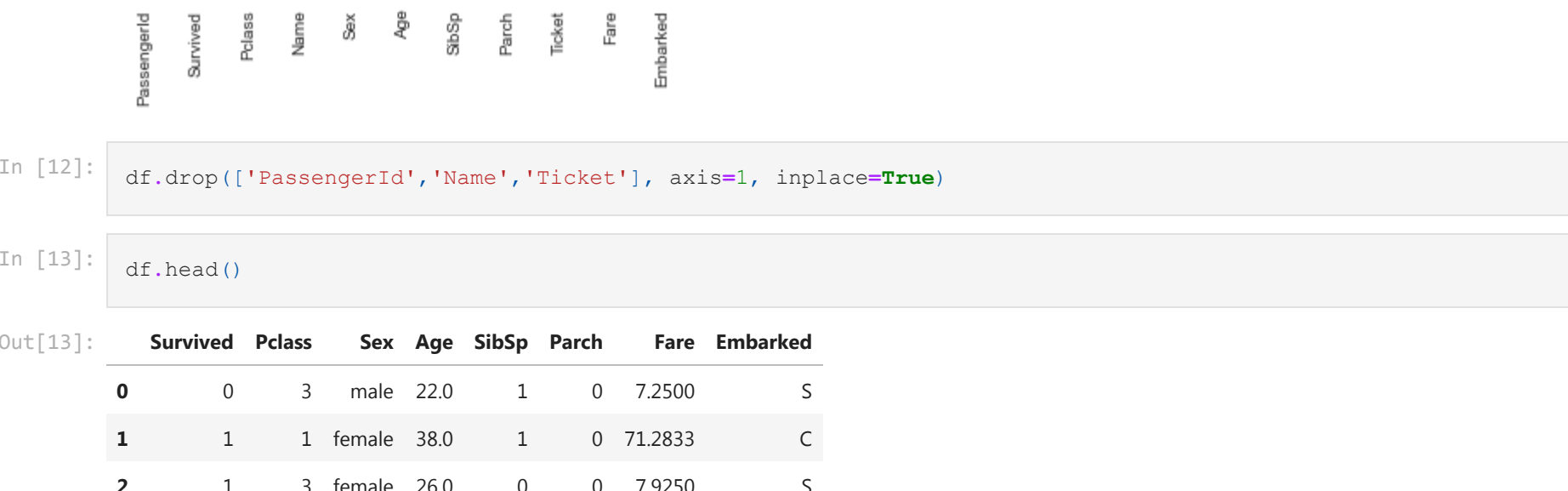
```
In [8]: sns.countplot(x="Survived", hue="Sex", data=df)

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



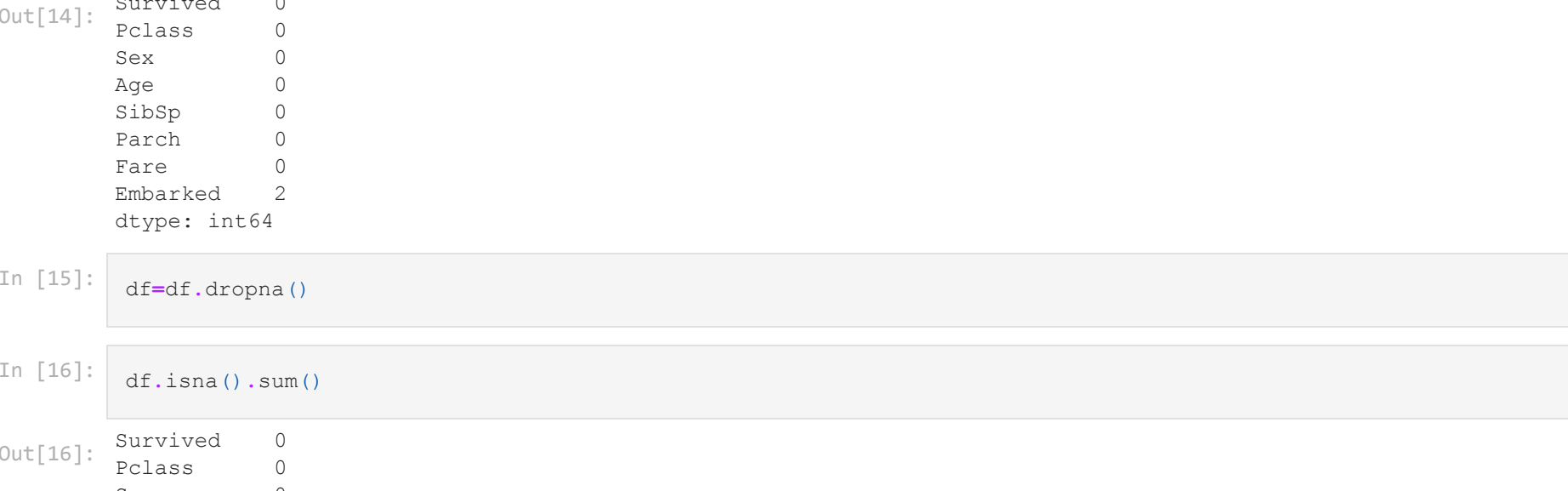
```
In [9]: sns.countplot(x="Survived", hue="Pclass", data=df, palette="rainbow")

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



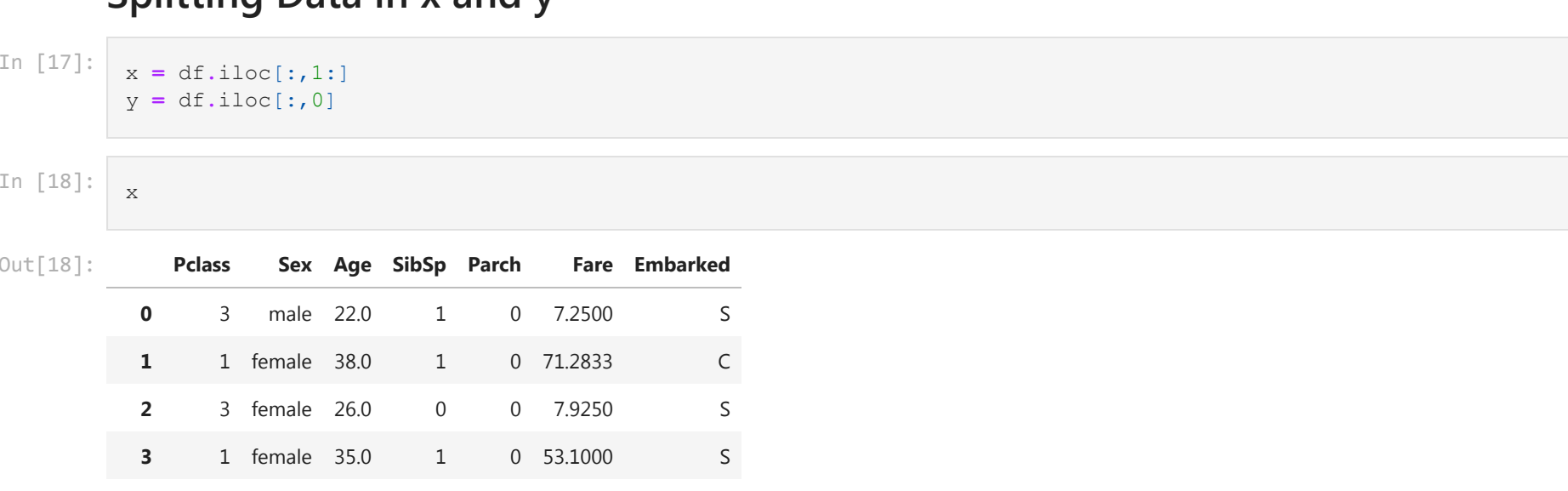
```
In [33]: df["Age"].hist(bins=30, alpha=1) # Skewed at right

<AxesSubplot:>
```



```
In [21]: sns.countplot(x="Pclass", hue="Sex", data=df)

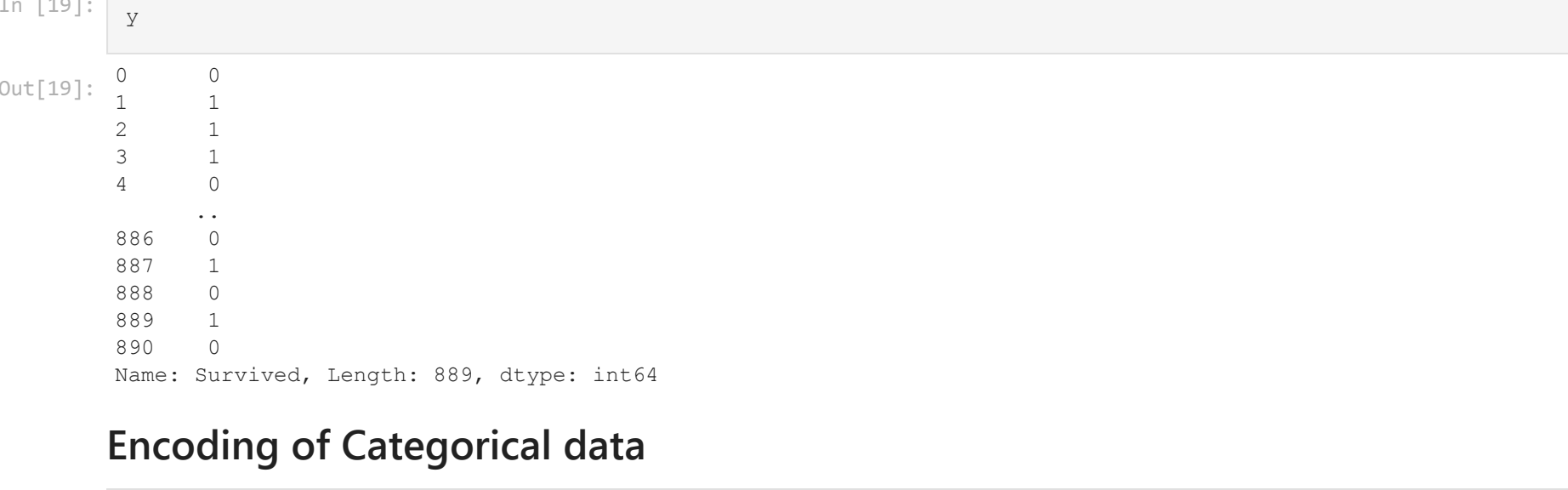
<AxesSubplot:xlabel='Pclass', ylabel='count'>
```



## Data Cleaning

```
In [8]: sns.boxplot(x="Pclass",y="Age",data=df)

<AxesSubplot:xlabel='Pclass', ylabel='Age'>
```

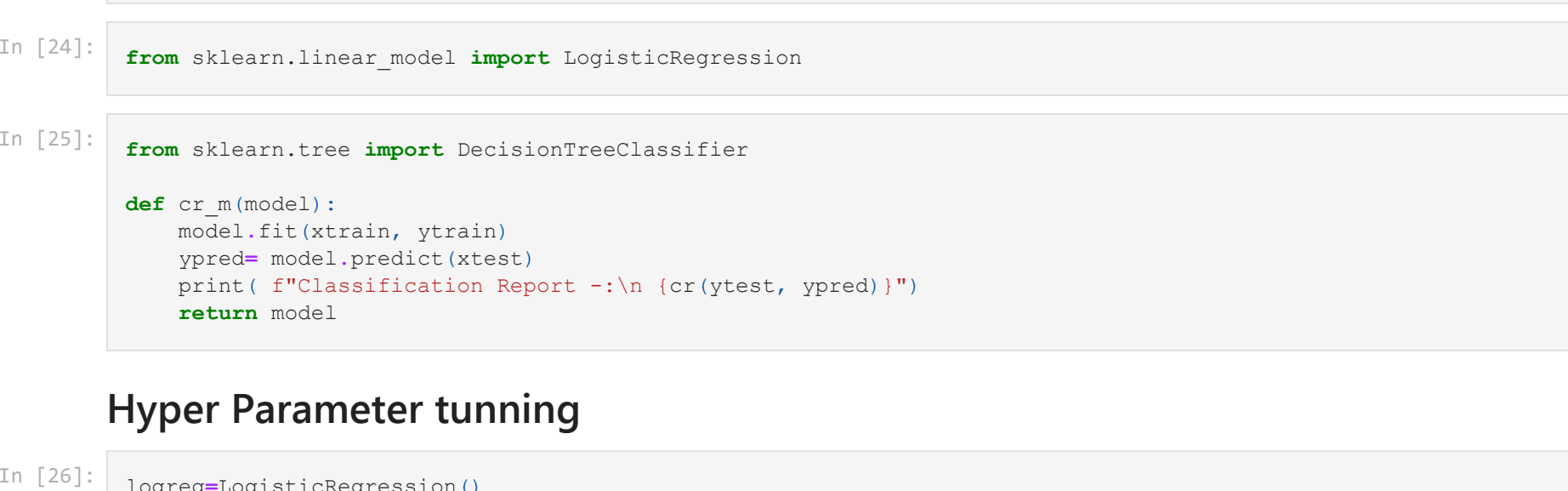


```
In [9]: def imputeage(cls):
    Age=cls[0]
    Pclass=cls[1]
    if pd.isnull(Age):
        if Pclass==1:
            return 37
        elif Pclass==2:
            return 29
        else:
            return 25
    else:
        return Age
```

```
In [10]: df["Age"]=df[["Age","Pclass"]].apply(imputeage,axis=1)
```

```
In [11]: sns.heatmap(df.isnull(), yticklabels=False, cbar = False, cmap='viridis')

<AxesSubplot:>
```



```
In [12]: df.drop(['PassengerId','Name','Ticket'], axis=1, inplace=True)
```

```
In [13]: df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S

```
In [14]: df.isna().sum()
```

```
Survived    0
Pclass      0
Sex          0
Age          0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

```
In [15]: df=df.dropna()
```

```
In [16]: df.isna().sum()
```

```
Survived    0
Pclass      0
Sex          0
Age          0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

## Splitting Data in x and y

```
In [17]: x = df.iloc[:,1:]
y = df.iloc[:,0]
```

```
In [18]: x
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S

```
In [14]: df.isna().sum()
```

```
Survived    0
Pclass      0
Sex          0
Age          0
SibSp       0
Parch       0
Fare        0
Embarked    2
dtype: int64
```

## Encoding of Categorical data

```
In [20]: from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

```
In [21]: ct=ColumnTransformer(transformers=[("encoder",OneHotEncoder(),["Sex","Embarked"])],remainder="passthrough")
x = np.array(ct.fit_transform(x))
```

```
In [22]: x
```

```
array([[ 0. ,  1. ,  0. , ...,  1. ,  0. ,  7.25 ],
       [ 1. ,  0. ,  1. , ...,  1. ,  0. , 71.2833],
       [ 1. ,  0. ,  0. , ...,  0. ,  0. ,  7.925 ],
       ...,
       [ 1. ,  0. ,  0. , ...,  1. ,  2. , 23.45 ],
       [ 0. ,  1. ,  1. , ...,  0. ,  0. , 30. ],
       [ 0. ,  1. ,  0. , ...,  0. ,  0. ,  7.75 ]])
```

## ML Model training & Creating

```
In [23]: from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state=1)
```

```
In [24]: from sklearn.linear_model import LogisticRegression
```

```
In [25]: from sklearn.tree import DecisionTreeClassifier
def cr_m(model):
    model.fit(xtrain, ytrain)
    ypred=model.predict(xtest)
    print("Classification Report ->\n",cr(ytest, ypred))
    return model
```

## Hyper Parameter tuning

```
In [26]: logreg=LogisticRegression()
cr_m(logreg)
```

```
Classification Report ->
              precision    recall  f1-score   support

    0       0.88      0.88      0.88       138
    1       0.80      0.80      0.80        85

 accuracy          0.84
 macro avg         0.84
weighted avg         0.85
```

```
Out[26]: LogisticRegression()
```

```
In [27]: for i in range(2,100):
    dt2 = DecisionTreeClassifier(min_samples_leaf=i)
    print(i)
    cr_m(dt2)
```





