# Classification on titanic dataset using Logistic Regression

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from warnings import filterwarnings
filterwarnings('ignore')
```

```
# We are working With titanic dataset
# lets convert our csv file to dataframe
```

```
train = pd.read_csv('titanic_train.csv')
```

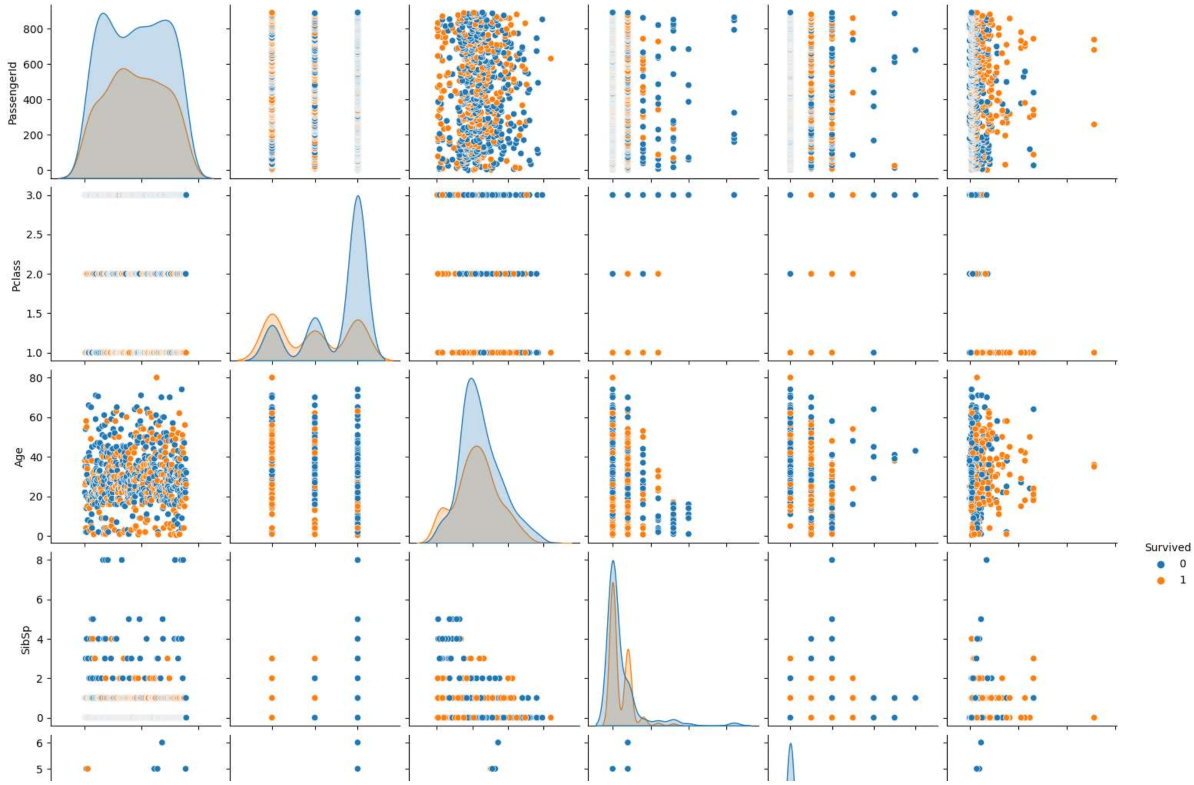
```
train.head()
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embark |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | NaN | |
| **1** | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.2833 | C85 | |
| **2** | 3 | 1 | 3 | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.9250 | NaN | |

```
# This code generates a pair plot of the 'train' data with the 'Survived' column used for color coding.

sns.pairplot(train, hue='Survived')
```

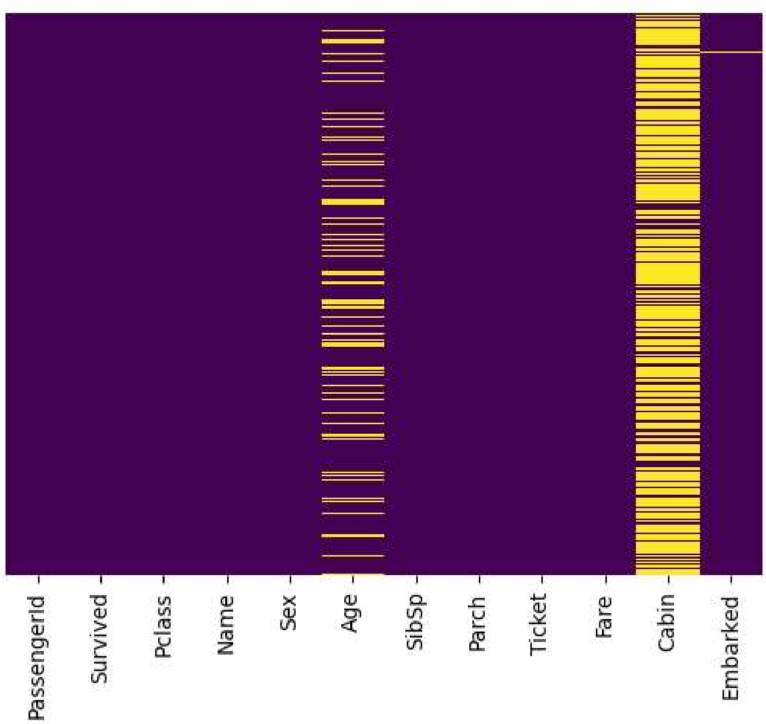
```
<seaborn.axisgrid.PairGrid at 0x7fa2eaecdf90>
```




```
# This code creates a heatmap visualization of missing values in a pandas dataframe.

# Create the heatmap.
sns.heatmap(train.isnull(), yticklabels=False, cbar=False, cmap='viridis')
```

```
<Axes: >
```




```
# As we can see that most of the cabin values are none means missing
# around 20% age values are missing we can fill this values.


# Now lets visualise some data
```

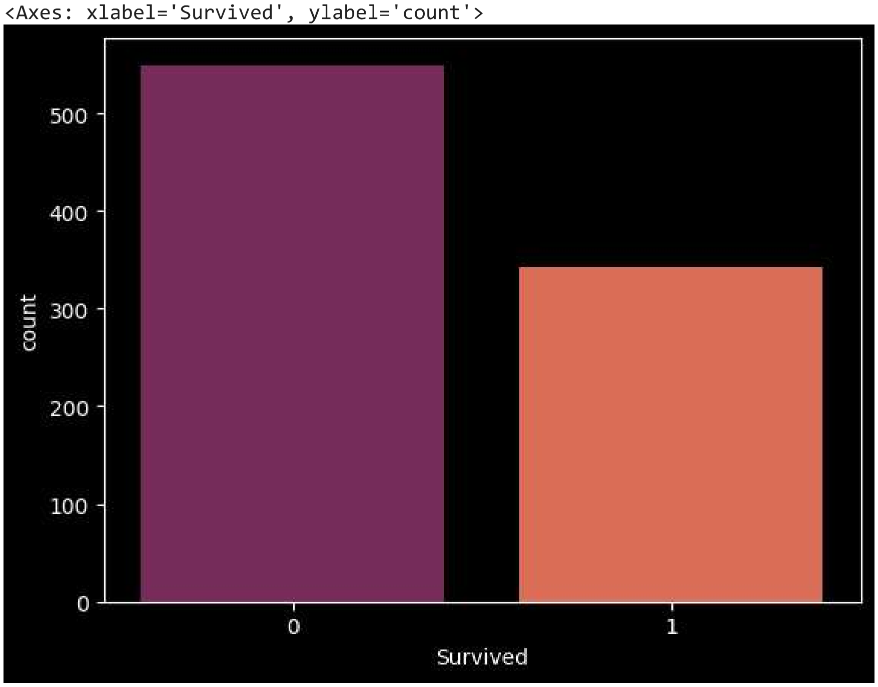
```
'''see the data column of survived and analyse who survived and who not
survived.
the best way to handle classification problems first understand the ratio
of the respective columns
For this use count plots'''
```

```
'see the data column of survived and analyse who survived and who not \nsurvived.\nthe best way to han
dle classification problems first understand the ratio \nof the respective columns\nFor this use count
plots'
```

```
plt.style.use('dark_background')
```

```
sns.countplot(x='Survived',data=train,palette='rocket')
```

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




```
sns.set_style('darkgrid')
```

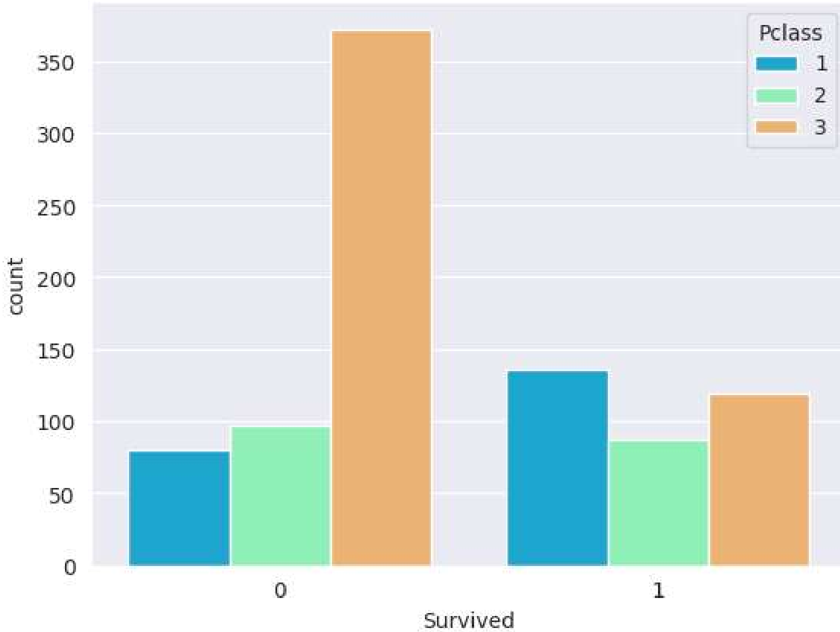
```
# Now differentiate the data acccording to sex
sns.countplot(x='Survived',data=train,hue='Sex',palette='icefire')
```

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

```
# analyse by class
```

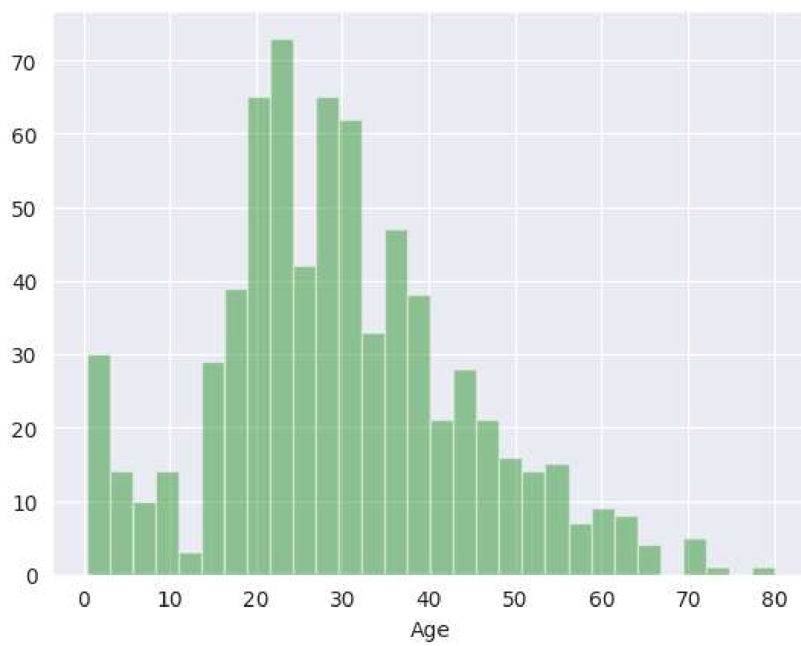
```
sns.countplot(x='Survived',data=train,hue='Pclass',palette='rainbow')
```

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




```
# Now, see how age column is distributed
```

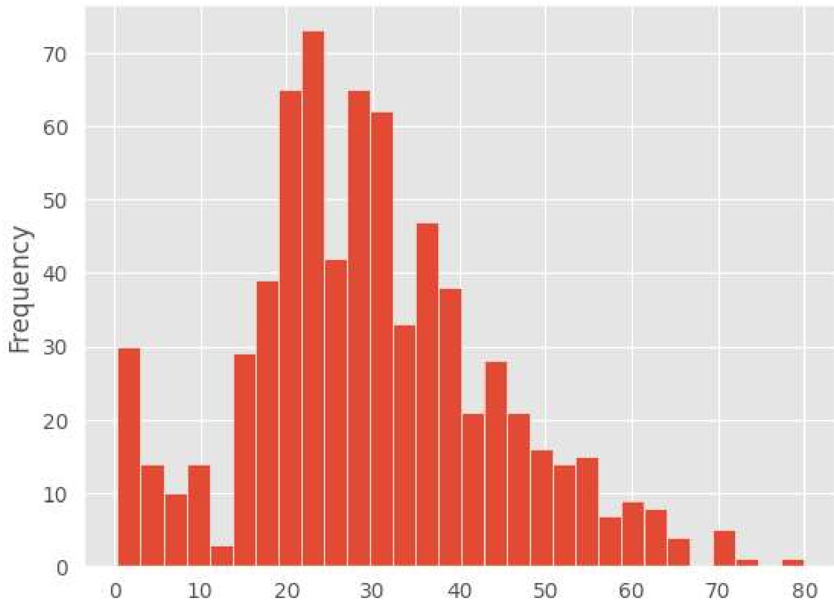
```
sns.distplot(train['Age'],kde=False,bins=30,color='green');
```




```
plt.style.use('ggplot')
```

```
train['Age'].plot.hist(bins=30)
```

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




```
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
```

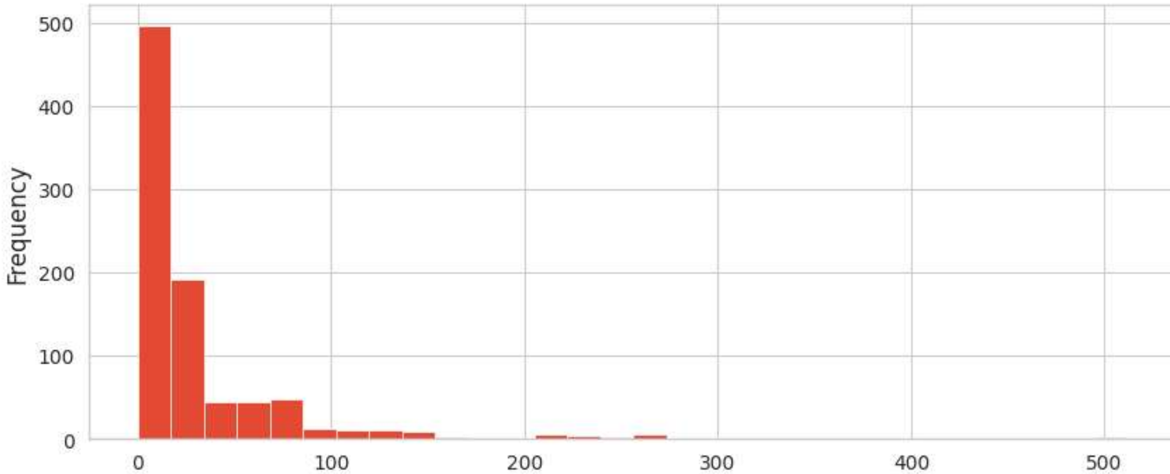
```
sns.set_style(style='whitegrid')
```

```
sns.countplot(x='SibSp',data=train)
```

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


```
train['Fare'].plot.hist(bins=30,figsize=(10,4))
```

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




```
# Lets do this with cufflinks
```

```
import cufflinks as cf
```

```
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
```

```
init_notebook_mode(connected=True)
```

```
cf.go_offline()
```

```
train['Age'].iplot(kind='hist',bins=30,colors='red')
```

## Cleaning Of The Data

```
# You see that there are missing data points in age columns as well as in cabin
```

```
'''Filling the data with values is konwn as emputation'''
```

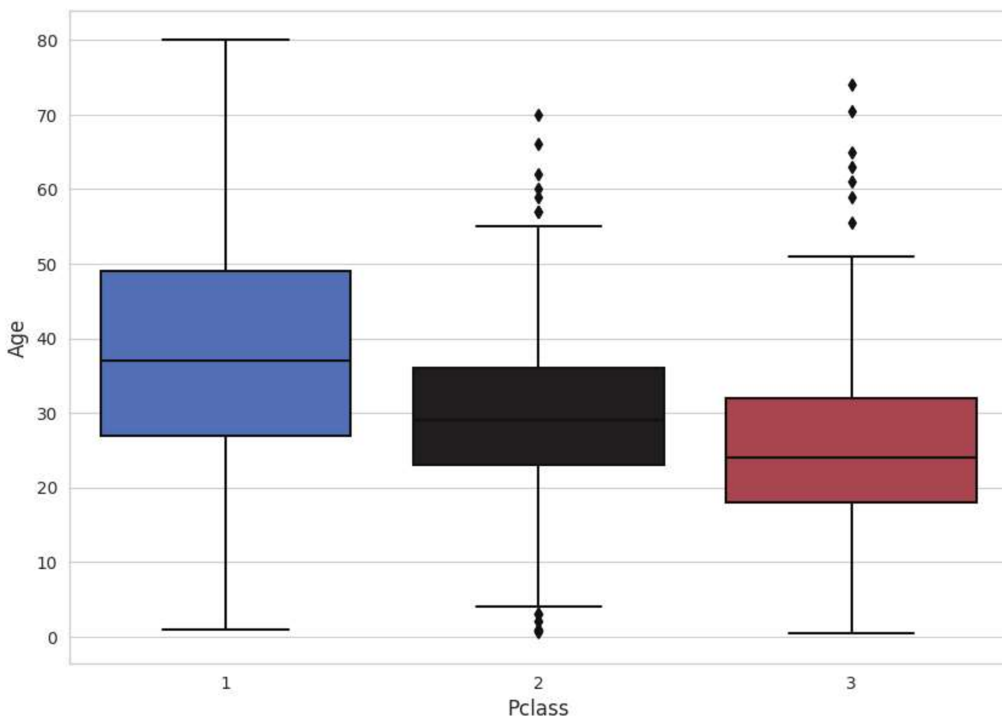
```
'Filling the data with values is konwn as emputation'
```

```
"""Now we have to fill the data with the avg of age according to class of
the persons in the titanic"""
```

```
'Now we have to fill the data with the avg of age according to class of \nthe persons i
n the titanic'
```

```
plt.figure(figsize=(10,7))
sns.boxplot(x='Pclass',y='Age',data=train,palette='icefire')
```

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



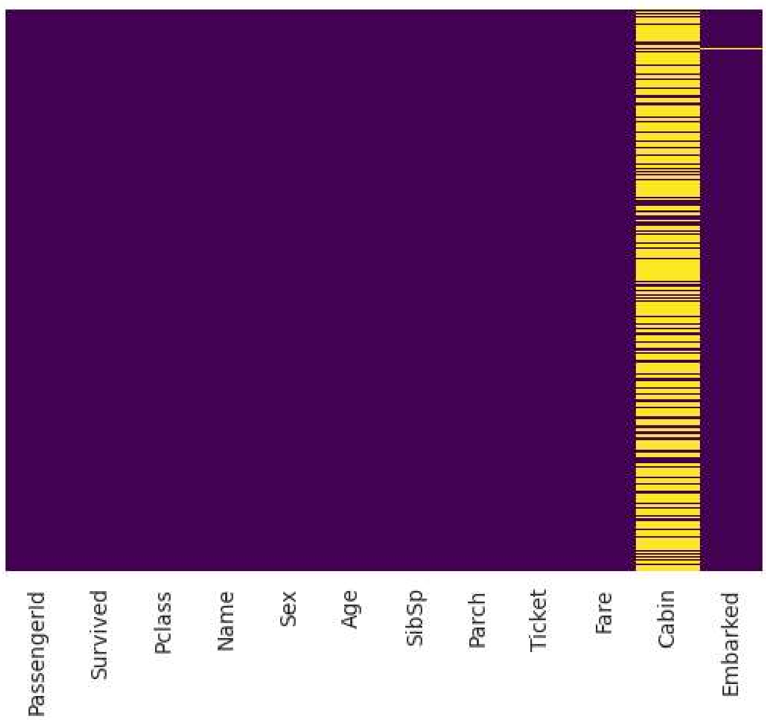

```
def impute_age(cols):
    Age = cols[0]
    Pclass = cols[1]
```

```
    if pd.isnull(Age):
        if Pclass == 1:
            return 37
        elif Pclass== 2:
            return 29
        else:
            return 24
    else:
        return Age
```

```
train['Age'] = train[['Age','Pclass']].apply(impute_age,axis=1)
```

```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<Axes: >
```




```
''' see cabin datapoints,there are so many missing values in this dataset
we can simply drop the column aslo we can classify the column as is cabbin or
not by filling the values as 1 or 0'''
```

```
' see cabin datapoints,there are so many missing values in this dataset\nwe can simply
drop the column aslo we can classify the column as is cabbin or \nnot by filling the va
lues as 1 or 0'
```

```
train.drop('Cabin',axis=1,inplace=True)
```

```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<Axes: >
```

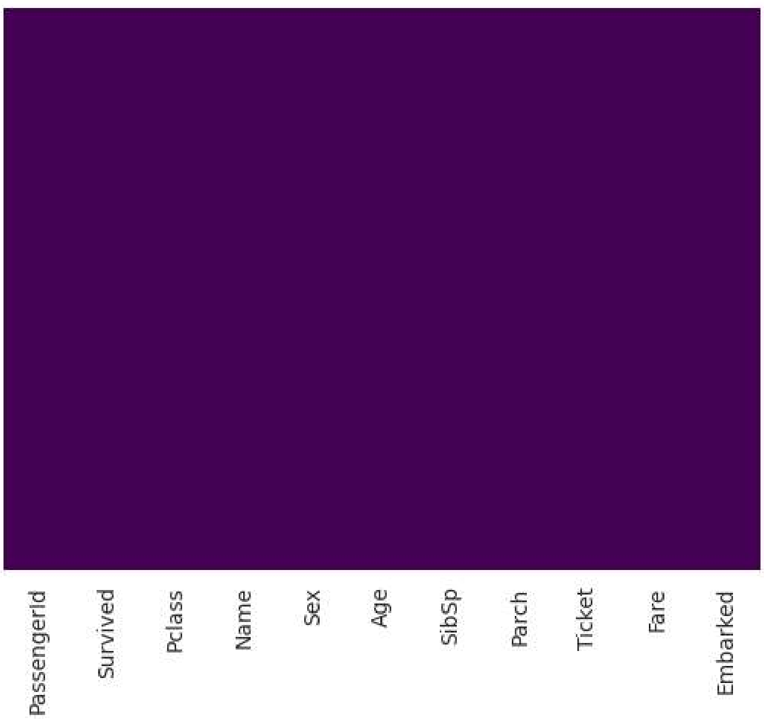
```
'''Now there is 1 row affect of missing data points'''
```

```
'Now there is 1 row affect of missing data points'
```

```
train.dropna(inplace=True)
```

```
sns.heatmap(train.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

```
<Axes: >
```




```
'''Now the next step is to create dummy variables for categorical values
such as for sex column dummy variables may be male=1 and female=0 like that
using pandas otherwise our machine learning algorithm won't be able to
directly take the values as input '''
```

```
'Now the next step is to create dummy variables for categorical values \nsuch as for se
x column dummy variables may be male=1 and female=0 like that\nusing pandas otherwise o
ur machine learning algorithm won't be able to \ndirectly take the values as input '
```

```
pd.get_dummies(train['Sex'],drop_first=True)
```

| | male |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 1 |
| **...** | ... |
| **886** | 1 |
| **887** | 0 |
| **888** | 0 |
| **889** | 1 |
| **890** | 1 |

```
'''
--->
there's one slight issue with this - one column is the perfect predictor
of the another column means if we give these values to our machine learnig
algorithm  if one says 0 for female machine learning model definately
going to say that the value is male this issue is known as 'multicolinearity'
Thats why we don't need one column we drop it
--->
Similarly a bunch of columns would be the perfect predictor of another
columns'''
```

```
'\n---> \nthere's one slight issue with this - one column is the perfect predictor \nof
the another column means if we give these values to our machine learnig \nalgorithm  if
one says 0 for female machine learning model definately\ngoing to say that the value is
male this issue is known as 'multicolinearity'\nThats why we don't need one column we d
rop it \n--->\nSimilarly a bunch of columns would be the perfect predictor of another\n
```

```
# droping one column
```

```
sex = pd.get_dummies(train['Sex'],drop_first=True)
```

```
sex.head()
```

| | male |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 0 |
| **3** | 0 |
| **4** | 1 |

```
embark =pd.get_dummies(train['Embarked'],drop_first=True)
```

```
embark.head()
```

| | Q | S |
|---|---|---|
| **0** | 0 | 1 |

```
# Note : Dummy variables are also known as indicators of the categorical values
```

| | | |
|---|---|---|
| **2** | 0 | 1 |

```
train = pd.concat([train,sex,embark],axis=1)
```

```
train.head(2)
```

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | E |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | 1 | 0 | A/5 21171 | 7.2500 | |
| | | | | Cumings, Mrs. John | | | | | | | |

```
''' See there we don't need sex,embark columns as we already have encoded them
also there is no need of text columns like name and ticket we simply drop them
'''
```

' See there we don't need sex,embark columns as we already have encoded them\nalso there is no need of text columns like name and ticket we simply drop them\n'

```
train.drop(['Sex','Embarked','Name','Ticket'],axis=1,inplace = True)
```

```
train.head()
```

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare | male | Q | S |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | 1 | 0 | 1 |
| **1** | 2 | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | 0 | 0 | 0 |
| **2** | 3 | 1 | 3 | 26.0 | 0 | 0 | 7.9250 | 0 | 0 | 1 |
| **3** | 4 | 1 | 1 | 35.0 | 1 | 0 | 53.1000 | 0 | 0 | 1 |
| **4** | 5 | 0 | 3 | 35.0 | 0 | 0 | 8.0500 | 1 | 0 | 1 |

```
# see ID column is just a serial wise arrangement as we have index simply
# will drop Id column also
```

```
train.drop('PassengerId',axis=1,inplace=True)
```

```
train.head(2)
```

| | Survived | Pclass | Age | SibSp | Parch | Fare | male | Q | S |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 3 | 22.0 | 1 | 0 | 7.2500 | 1 | 0 | 1 |
| **1** | 1 | 1 | 38.0 | 1 | 0 | 71.2833 | 0 | 0 | 0 |

```
'''Now everything is converted into numerical values which is good for our
algorithm
Survived is our label column which we are going to predict using features
given '''
```

```
'Now everything is converted into numerical values which is good for our \nalgorithm \n
```

```
# Now make model to predict whether passenger alives or died
```

```
X = train.drop('Survived',axis = 1)
y = train['Survived']
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
from sklearn.linear_model import LogisticRegression
```

```
logmodel = LogisticRegression()
```

```
logmodel.fit(X_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
predictions = logmodel.predict(X_test)
```

```
predictions
```

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 1])
```

```
y_test
```

```
511    0
613    0
615    1
337    1
718    0
      ..
792    0
828    1
732    0
669    1
634    0
Name: Survived, Length: 267, dtype: int64
```

```
# Now we have to test acurecy of our model
# evaluate our model
```

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86       163
           1       0.82      0.71      0.76       104

    accuracy                           0.83       267
   macro avg       0.83      0.81      0.81       267
weighted avg       0.83      0.83      0.83       267
```

```
from sklearn.metrics import confusion_matrix
```

```
print(confusion_matrix(y_test,predictions))
```

```
[[147  16]
 [ 30  74]]
```

✓ 0s completed at 11:21 PM