Step1 = Import all Necessary Libraries

In [61]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
```

Step2 = Load Dataset

In [32]:
```
df=pd.read_csv("/content/titanic_train.csv")
df.head()
```

Out[32]:

	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	Cumings, 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

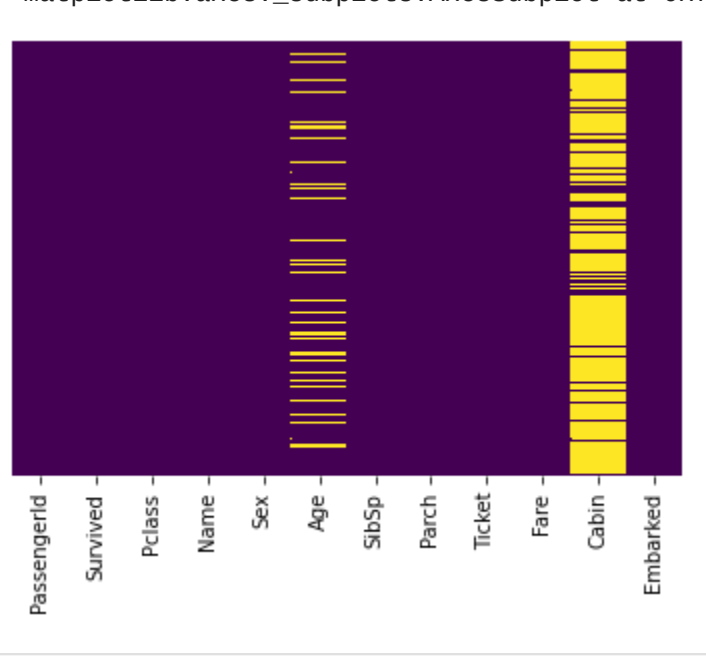
Step3= EDA

In [33]:
```
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(5), object(5)
memory usage: 83.7+ KB
```

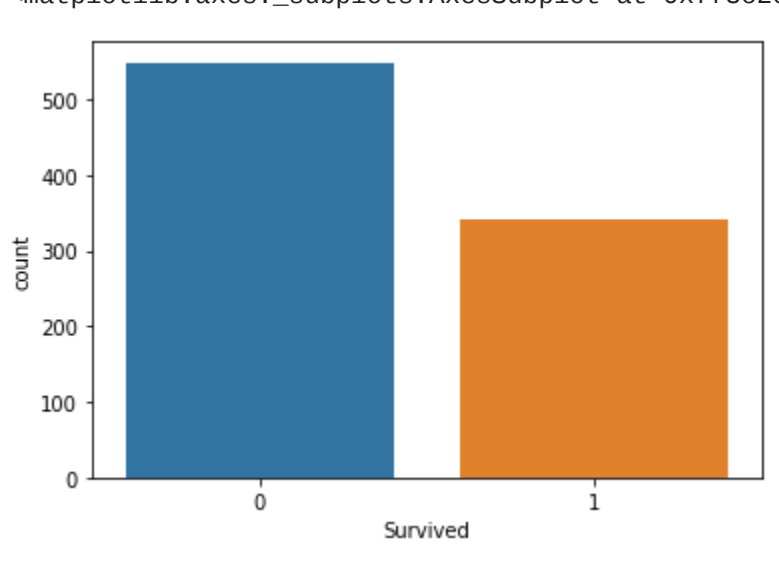
In [5]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar = False, cmap='viridis')
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c4db76790>



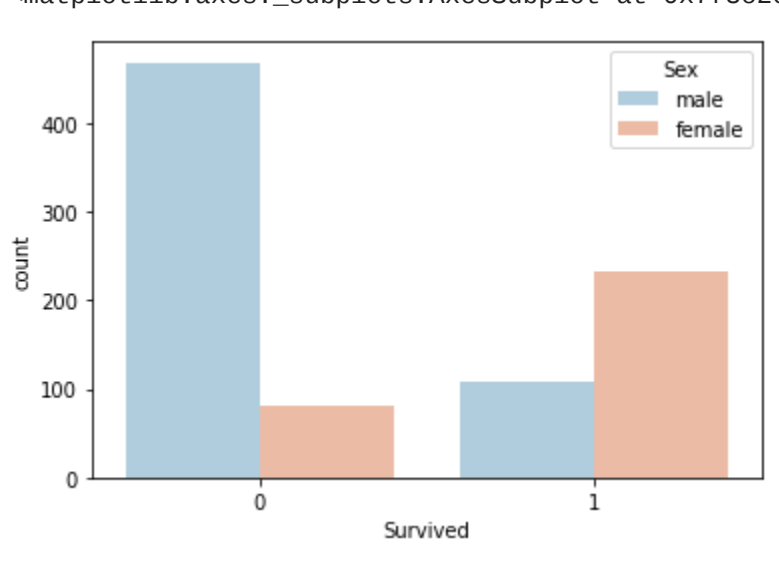
In [34]:
```
sns.countplot(x='Survived', data=df)
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ece5210>



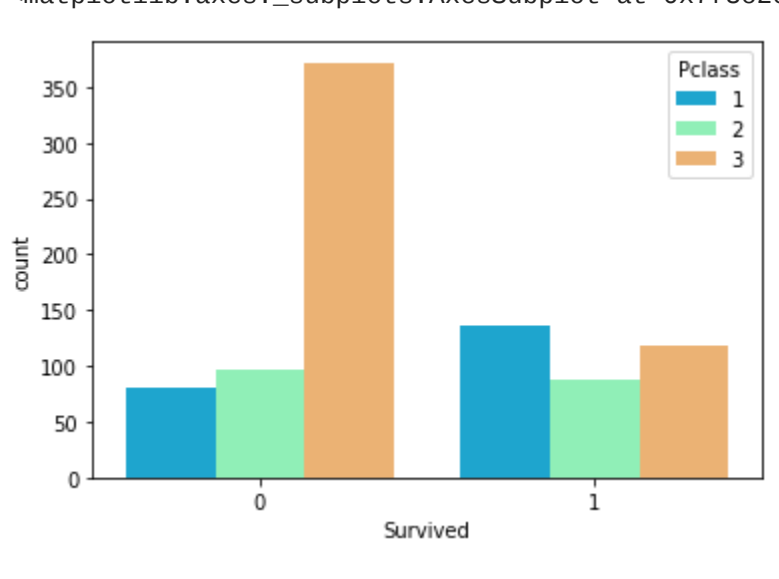
In [35]:
```
sns.countplot(x='Survived', hue='Sex', data=df, palette='RdBu_r')
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ed668d0>



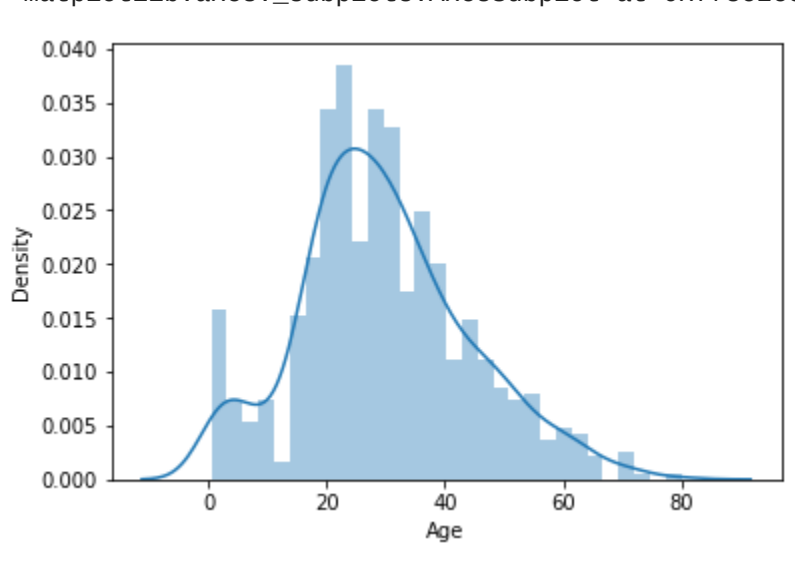
In [36]:
```
sns.countplot(x='Survived', hue='Pclass', data=df, palette='rainbow')
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ecb55d0>



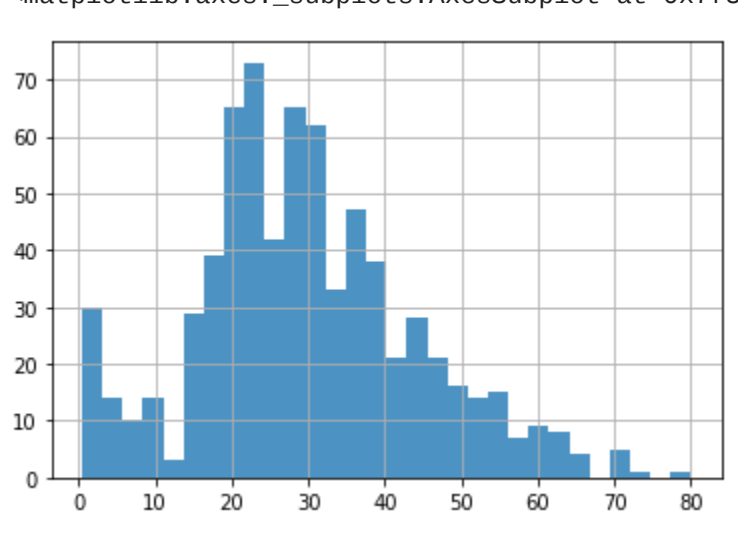
In [37]:
```
sns.distplot(df['Age'].dropna(), bins=30)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ec0b790>



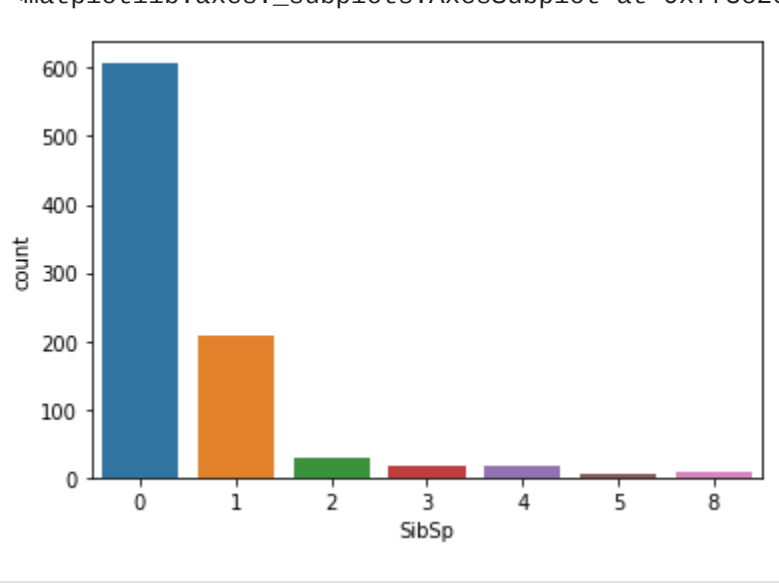
In [38]:
```
df['Age'].hist(bins=30, alpha=0.8)
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ebbead0>



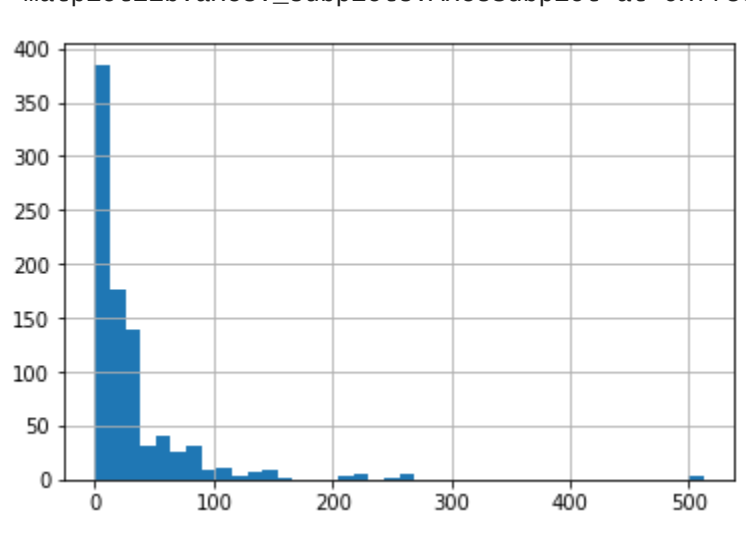
In [39]:
```
sns.countplot(x='SibSp',data=df)
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ead8550>



In [40]:
```
df['Fare'].hist(bins=40)
```

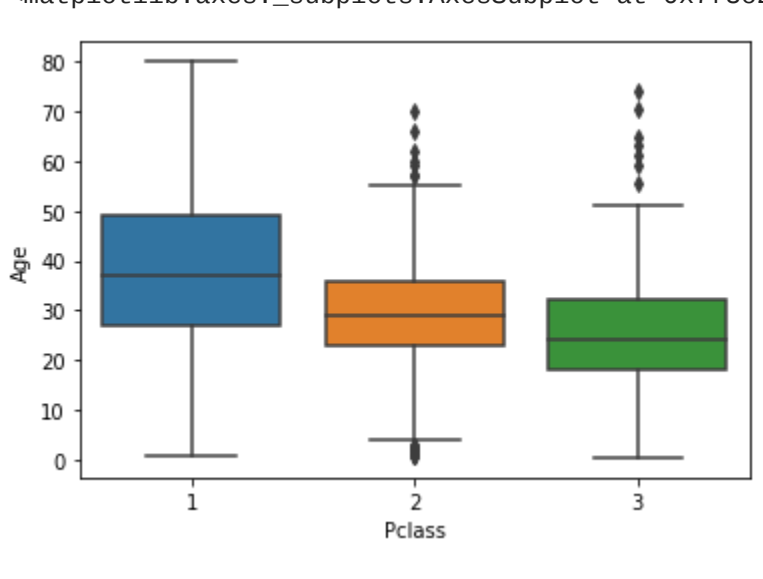
Out[40]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2ea38450>



Step4 = Data Cleaning

In [41]:
```
sns.boxplot(x="Pclass",y='Age',data=df)
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2e8e5290>

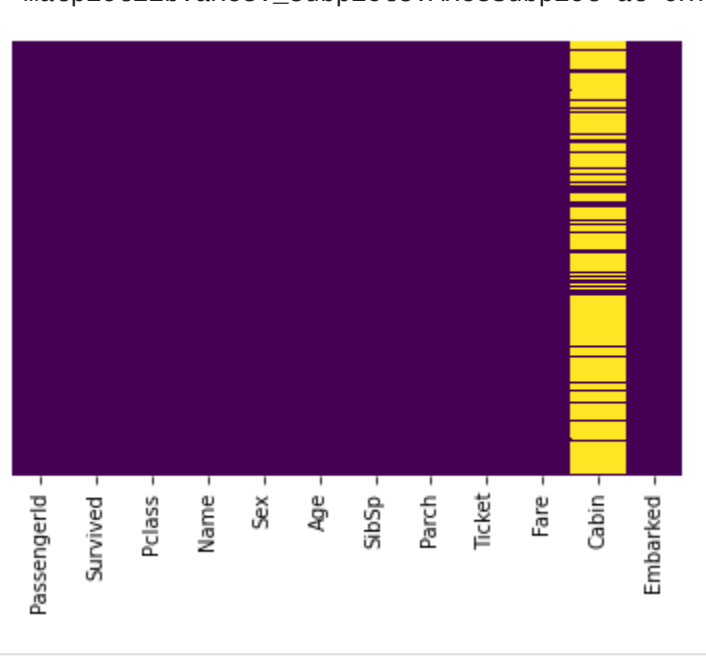


In [43]:
```
#IMPUTE

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

    if(pd.isnull(Age)):
        if(Pclass==1):
            return 37
        elif(Pclass==2):
            return 29
        else:
            return 25
    else:
        return Age
```

In [44]:
```
df['Age'] = df[['Age','Pclass']].apply(imputeage, axis=1)
```

In [45]:
```
sns.heatmap(df.isnull(), yticklabels=False, cbar = False, cmap='viridis')
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3c2e881210>



In [46]:
```
#dropping Columns
df.drop(['PassengerId','Name','Cabin','Ticket'], axis=1, inplace=True)
```

In [47]:
```
df.head()
```

Out[47]:

	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 [48]:
```
df = df.dropna()
```

In [49]:
```
df.isna().sum()
```

Out[49]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

Step5 = Split X and Y

In [50]:
```
x = df.iloc[:,1:]
y = df.iloc[:,0]
```

In [51]:
```
x
```

Out[51]:

	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
...	...	...	...	...	...	...	...
886	2	male	27.0	0	0	13.0000	S
887	1	female	19.0	0	0	30.0000	S
888	3	female	25.0	1	2	23.4500	S
889	1	male	26.0	0	0	30.0000	C
890	3	male	32.0	0	0	7.7500	Q

889 rows × 7 columns

In [52]:
```
y
```

Out[52]:
```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 889, dtype: int64
```

Step6 = Encoding

In [53]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

In [54]:
```
ct = ColumnTransformer(transformers = [('encoder',OneHotEncoder(), ['Sex','Embarked'])],
                       remainder='passthrough')

x = np.array( ct.fit_transform(x))
```

In [55]:
```
x[0]
```

Out[55]:
```
array([ 0.  ,  1.  ,  0.  ,  0.  ,  1.  ,  3.  , 22.  ,  1.  ,  0.  ,
        7.25])
```

Step7 = Train and Test

In [56]:
```
from sklearn.model_selection import train_test_split, cross_val_score

xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size=0.30, random_state=101)
```

Step8 = Model Creation

In [60]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(xtrain, ytrain)

ypred = rf.predict(xtest)
rf.score(xtrain, ytrain)
acc_rf = round(rf.score(xtrain, ytrain) * 100, 2)
```

Step9 = Model Evaluation

In [58]:
```
#Confusion matrics, accuracy and classificatio report

from sklearn.metrics import confusion_matrix as cm
r = cm(ytest, ypred)
print(r)
from sklearn.metrics import accuracy_score
print(f"Accuracy -: {accuracy_score(ytest, ypred)}")
from sklearn.metrics import classification_report as cr
r = cr(ytest, ypred)
print(r)
```

```
[[140  23]
 [ 27  77]]
Accuracy -: 0.8127340823970037
              precision    recall  f1-score   support

           0       0.84      0.86      0.85       163
           1       0.77      0.74      0.75       104

    accuracy                           0.81       267
   macro avg       0.80      0.80      0.80       267
weighted avg       0.81      0.81      0.81       267
```

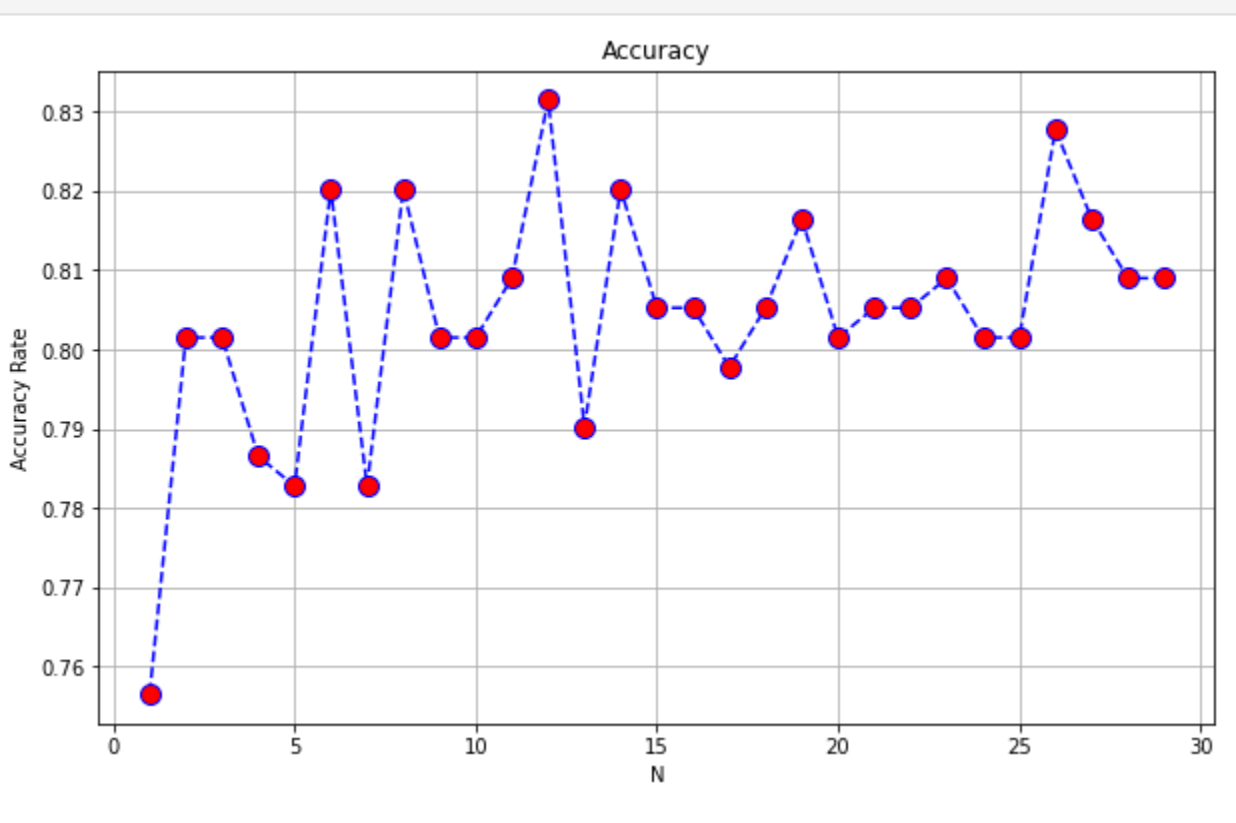
In [62]:
```
#Mean Accuracy with cross validation score

from sklearn.model_selection import cross_val_score
cvs = cross_val_score(KNeighborsClassifier(n_neighbors=8), x,y, cv = 16, scoring='accuracy')
print(f" Accuracy -: {cvs.mean()*100}")
print(f'Standard Deviation:{cvs.std()}')
```

```
 Accuracy -: 70.31655844155844
Standard Deviation:0.059720350207942974
```

Step10 = Hyperparmeter Tunning with GridSearch CV

In [63]:
```
#create model again

n = list(range(1,30))
accuracy = []

for i in n:
 rf = RandomForestClassifier(n_estimators=i )
 rf.fit(xtrain, ytrain)
 ypred = rf.predict(xtest)
 ac = accuracy_score(ytest, ypred)
 accuracy.append(ac)
```

In [64]:
```
#plot from loop

plt.figure(figsize=(10,6))
plt.plot(range(1,30), accuracy, color='blue', linestyle='dashed', marker='o',
 markerfacecolor='red', markersize=10)
plt.title('Accuracy')
plt.xlabel('N')
plt.ylabel('Accuracy Rate')
plt.grid(True)
plt.show()
```



Step11 = Retrain Model

In [65]:
```
#Model Creation

rf = RandomForestClassifier(n_estimators=100, oob_score = True)
rf.fit(xtrain, ytrain)
ypred = rf.predict(xtest)

rf.score(xtrain, ytrain)

acc_rf = round(rf.score(xtrain, ytrain) * 100, 2)
```

In [66]:
```
#GridSearchCv

from sklearn.model_selection import GridSearchCV
```

In [67]:
```
params = {"n_estimators":[10,100,1000],"criterion" : ["gini", "entropy"],"n_estimators": [5, 10, 15, 20, 25],
    "max_depth": [3, 5, 7, 9, 11, 13]}
model_rf = GridSearchCV(rf, params, cv=5,scoring='accuracy')
```

In [68]:
```
model_rf.fit(xtrain,ytrain)
```

Out[68]:
```
GridSearchCV(cv=5, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              n_estimators=100, n_jobs=None,
                                              oob_score=True, random_state=None,
                                              verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 5, 7, 9, 11, 13],
                         'n_estimators': [5, 10, 15, 20, 25]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

In [69]:
```
model_rf.best_estimator_
```

Out[69]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=9, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=15,
                       n_jobs=None, oob_score=True, random_state=None,
                       verbose=0, warm_start=False)
```

In [70]:
```
model_rf.best_params_
```

Out[70]:
```
{'criterion': 'entropy', 'max_depth': 9, 'n_estimators': 15}
```

In [71]:
```
gpred=model_rf.predict(xtest)
```

In [72]:
```
print(f"Accuracy -: {accuracy_score(ytest, gpred)}")
```

```
Accuracy -: 0.8389513108614233
```

In [73]:
```
c=cm(ytest,gpred)
print(f"Confusion Matrics-: \n {c}")
```

```
Confusion Matrics-:
 [[151  12]
 [ 31  73]]
```

In [74]:
```
r=cr(ytest,gpred)
print(f"Classification Report -: \n {r}")
```

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

           0       0.83      0.93      0.88       163
           1       0.86      0.70      0.77       104

    accuracy                           0.84       267
   macro avg       0.84      0.81      0.82       267
weighted avg       0.84      0.84      0.84       267
```