In [48]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [49]:

```
data=pd.read_csv('https://raw.githubusercontent.com/dsrscientist/dataset1/master/titanic_train.csv')
```

In [50]:

```
data.head()
```

Out[50]:

	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

In [51]:

```
data.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 [52]:

```
#How many survived
num=data[data['Survived']==1]['Survived'].count()
num
```

Out[52]:

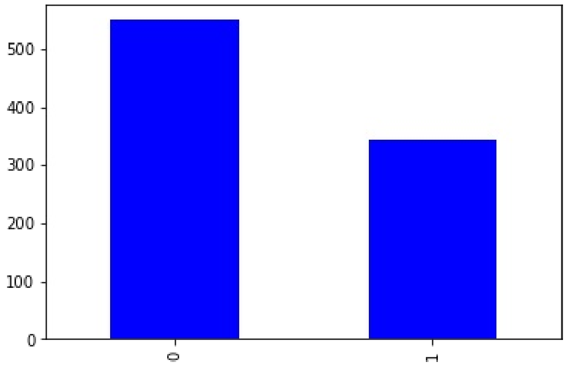
```
342
```

In [53]:

```
survival=data['Survived'].value_counts()
survival.plot(kind='bar',color='blue')
```

Out[53]:

```
<AxesSubplot:>
```



In [54]:

```
#Children died but their relatives survived
child=data[data['Age']<18]
cond1=child[(child['SibSp']!=0) | (child['Parch']!=0)]['Survived'].value_counts()
cond1[0]
```

Out[54]:

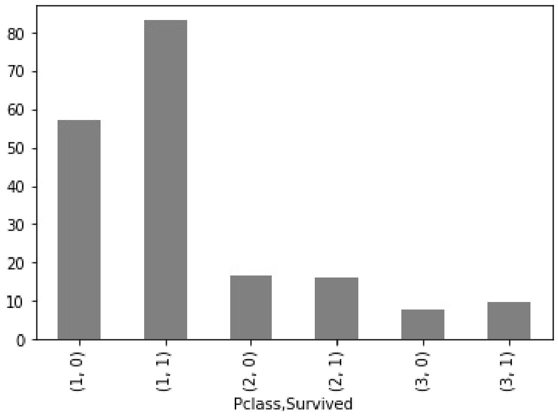
```
39
```

In [55]:

```
#no.of people above 50 travelling and have survived along with their class
senior=data[data['Age']>=50]
senior.groupby(['Pclass','Survived'])['Fare'].mean().plot(kind='bar',color='grey')
```

Out[55]:

```
<AxesSubplot:xlabel='Pclass,Survived'>
```

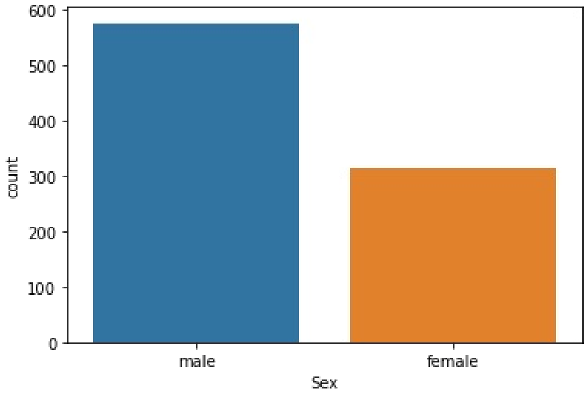


In [56]:

```
#Countplot
sns.countplot(x='Sex',data=data)
```

Out[56]:

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

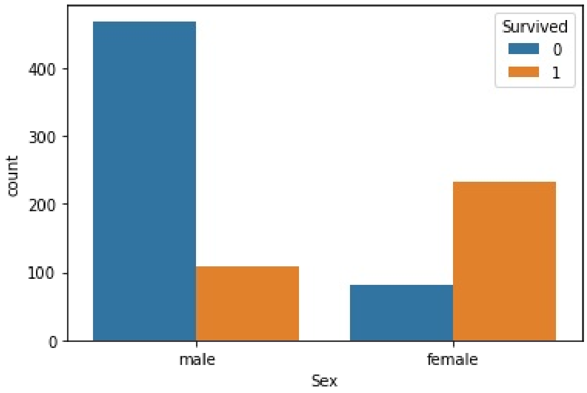


In [57]:

```
sns.countplot(x='Sex',hue='Survived',data=data)
```

Out[57]:

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

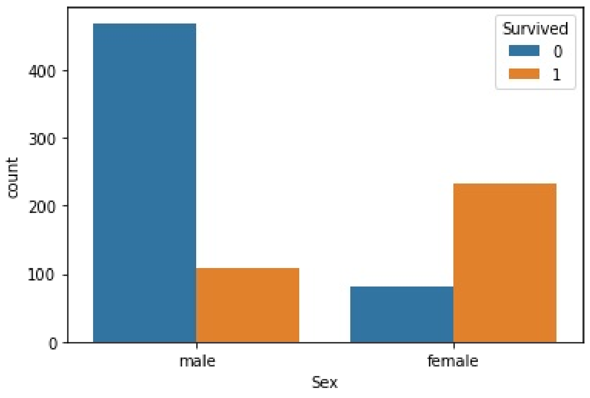


In [58]:

```
sns.countplot(x='Sex',hue='Survived',data=data)
```

Out[58]:

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

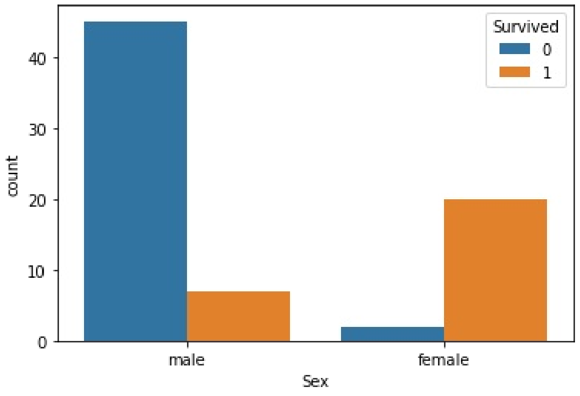


In [59]:

```
sns.countplot(x='Sex',hue='Survived',data=senior)
```

Out[59]:

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



In [60]:

```
senior.groupby('Pclass')['Sex'].value_counts()
```

Out[60]:

```
Pclass  Sex   
1       male      29
        female    15
2       male      13
        female     6
3       male      10
        female     1
Name: Sex, dtype: int64
```

In [61]:

```
#Sex ratio initially
total=data['Sex'].value_counts()
total['female']/total['male']
```

Out[61]:

```
0.5441941074523396
```

In [62]:

```
data.shape
```

Out[62]:

```
(891, 12)
```

In [63]:

```
data.head()
```

Out[63]:

	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

In [64]:

```
data.isnull().sum()
```

Out[64]:

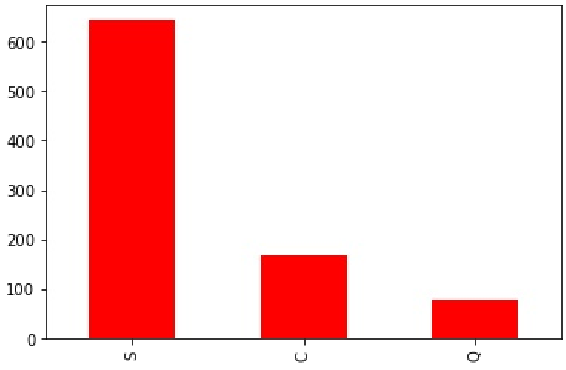
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [65]:

```
data['Embarked'].value_counts().plot(kind='bar',color='red')
```

Out[65]:

```
<AxesSubplot:>
```

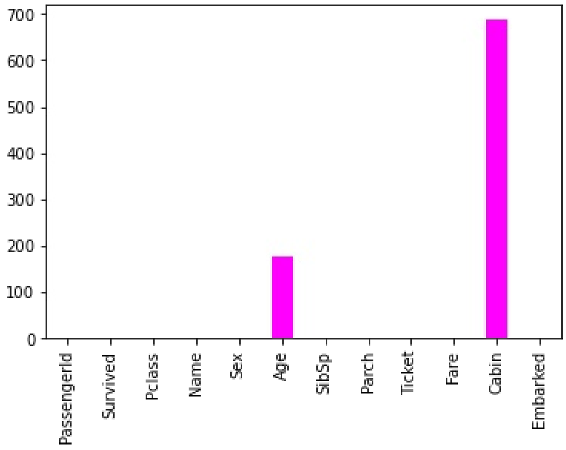


In [66]:

```
data['Embarked'].fillna('S',inplace=True)
data.isnull().sum().plot(kind='bar',color='magenta')
```

Out[66]:

```
<AxesSubplot:>
```



In [67]:

In [67]:

```
#Class wise mean of age
data.groupby('Pclass')["Age"].mean()
```

Out[67]:

```
Pclass
1    38.233441
2    29.877630
3    25.140620
Name: Age, dtype: float64
```

In [68]:

```
def input_age(cols):
    Age=cols[0]
    Pclass=cols[1]
    if pd.isnull(Age):
        if Pclass==1:
            return 40.918367
        if Pclass==2:
            return 28.777500
        if Pclass==3:
            return 24.027945
    else:
        return Age
```

In [69]:

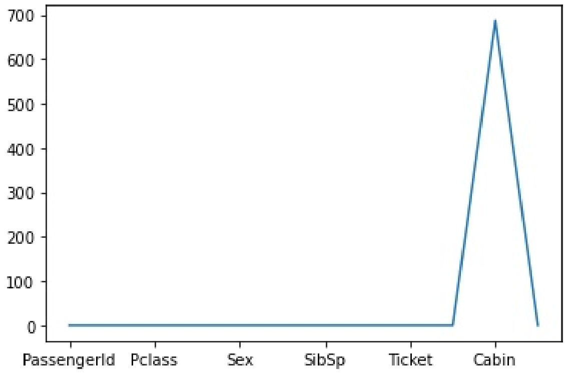
```
data["Age"]=data[["Age","Pclass"]].apply(input_age,axis=1)
```

In [70]:

```
data.isnull().sum().plot()
```

Out[70]:

```
<AxesSubplot:>
```



In [71]:

```
#Class wise mean of fare
data.groupby('Pclass')["Fare"].mean()
```

Out[71]:

```
Pclass
1    84.154687
2    20.662183
3    13.675550
Name: Fare, dtype: float64
```

In [72]:

```
#Converting Categorical values into Numerical Ones
from sklearn.preprocessing import LabelEncoder
lb=LabelEncoder()
data['Sex']=lb.fit_transform(data["Sex"])
data['Embarked']=lb.fit_transform(data["Embarked"])
```

In [73]:

```
data['Sex']
```

Out[73]:

```
0      1
1      0
2      0
3      0
4      1
      ..
886    1
887    0
888    0
889    1
```

```
890    1
Name: Sex, Length: 891, dtype: int32
```

In [74]:

```
data['Embarked']
```

Out[74]:

```
0      2
1      0
2      2
3      2
4      2
      ..
886    2
887    2
888    2
889    0
890    1
Name: Embarked, Length: 891, dtype: int32
```

In [75]:

```
data.head()
```

Out[75]:

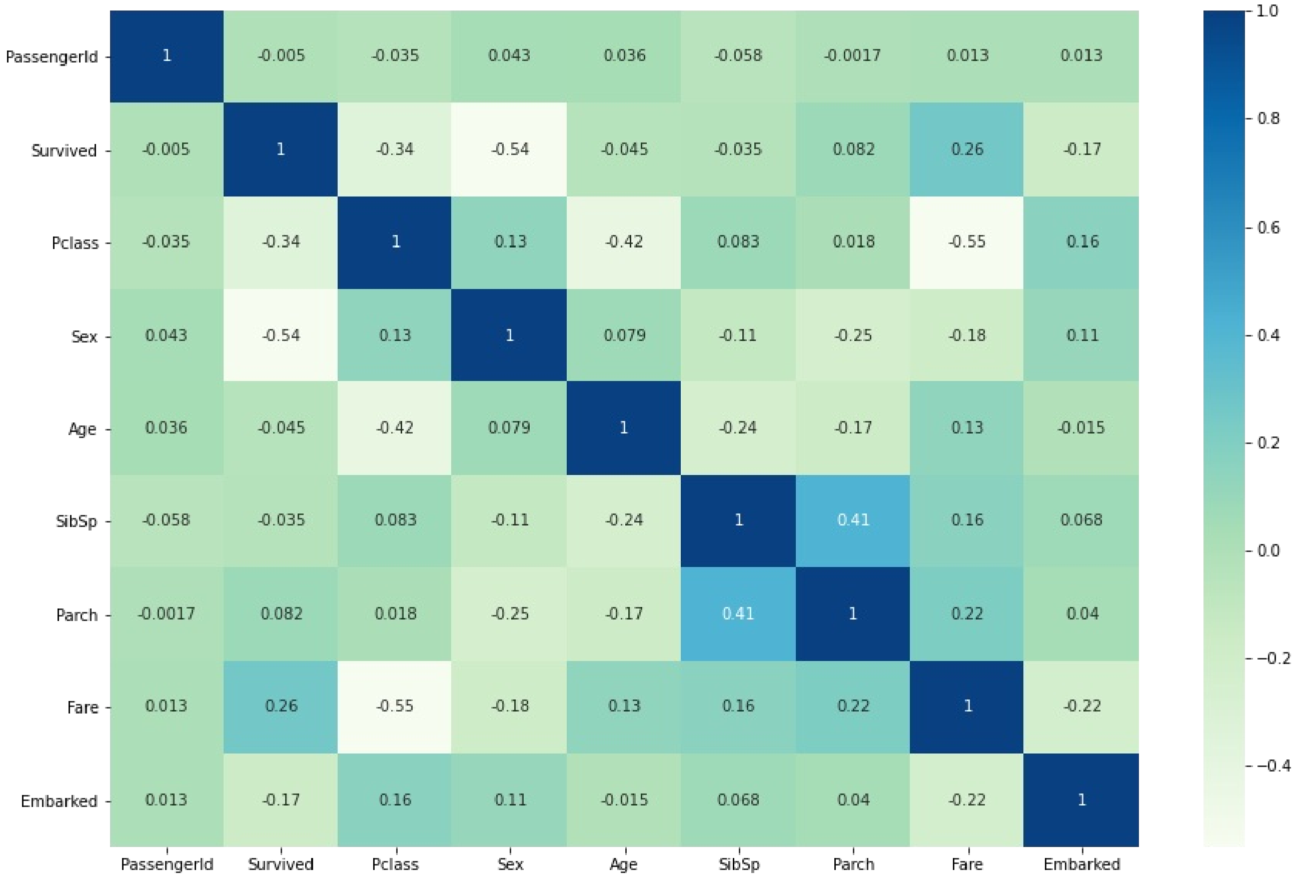
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NaN	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	0
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250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

In [76]:

```
plt.figure(figsize=(15,10))
sns.heatmap(data.corr(),annot=True,cmap='GnBu')
```

Out[76]:

```
<AxesSubplot:>
```



In [82]:

```
data_new=data.drop(columns=['PassengerId','Cabin','Ticket','Name'],axis=1)
data_new
```

Out[82]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.000000	1	0	7.2500	2
1	1	1	0	38.000000	1	0	71.2833	0
2	1	3	0	26.000000	0	0	7.9250	2
3	1	1	0	35.000000	1	0	53.1000	2
4	0	3	1	35.000000	0	0	8.0500	2
...	...	...	...	...	...	...	...	...
886	0	2	1	27.000000	0	0	13.0000	2
887	1	1	0	19.000000	0	0	30.0000	2
888	0	3	0	24.027945	1	2	23.4500	2
889	1	1	1	26.000000	0	0	30.0000	0
890	0	3	1	32.000000	0	0	7.7500	1

891 rows × 8 columns

In [80]:

```
##
```

In [83]:

```
data_new.head()
```

Out[83]:

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

In [84]:

```
x=data_new.drop('Survived',axis=1)
y=data_new['Survived']
```

In [85]:

```
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=30)
```

In [86]:

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

In [87]:

```
model=LogisticRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
y_test
```

Out[87]:

```
417    1
307    1
87     0
577    1
684    0
      ..
150    0
800    0
645    1
824    0
17     1
Name: Survived, Length: 268, dtype: int64
```

In [88]:

```
y_pred
```

Out[88]:

```
array([1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 0], dtype=int64)
```

In [89]:

```
from sklearn.metrics import accuracy_score
lr=accuracy_score(y_test,y_pred)*100
lr
```

Out[89]:

```
79.47761194029852
```

In [90]:

```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test)
```

In [91]:

```
from sklearn.metrics import accuracy_score
nb=accuracy_score(y_test,y_pred)*100
nb
```

Out[91]:

```
75.3731343283582
```

In [92]:

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier()
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
#Accuracy using Decision Tree
from sklearn import metrics
dt=metrics.accuracy_score(y_test,y_pred)*100
dt
```

Out[92]:

```
80.59701492537313
```

In [93]:

```
from sklearn.svm import SVC
svc_classifier=SVC(kernel='linear')
svc_classifier.fit(X_train,y_train)
y_pred=svc_classifier.predict(X_test)
y_pred=svc_classifier.predict(X_test)
from sklearn import metrics
sv=metrics.accuracy_score(y_test,y_pred)*100
sv
```

Out[93]:

```
76.49253731343283
```

In [94]:

```
from sklearn.neighbors import KNeighborsClassifier
for i in range(3,10):
    knn_classifier=KNeighborsClassifier(n_neighbors=i,metric='euclidean')
    knn_classifier.fit(X_train,y_train)
    y_pred=knn_classifier.predict(X_test)
    from sklearn import metrics
    print(i, metrics.accuracy_score(y_test,y_pred)*100)
```

```
3 71.64179104477611
4 69.02985074626866
5 70.8955223880597
6 70.8955223880597
7 72.01492537313433
8 71.64179104477611
9 72.76119402985076
```

In [95]:

```
knn_classifier=KNeighborsClassifier(n_neighbors=9,metric='euclidean')
knn_classifier.fit(X_train,y_train)
y_pred=knn_classifier.predict(X_test)
from sklearn import metrics
kn=metrics.accuracy_score(y_test,y_pred)*100
kn
```

Out[95]:

```
72.76119402985076
```

In [96]:

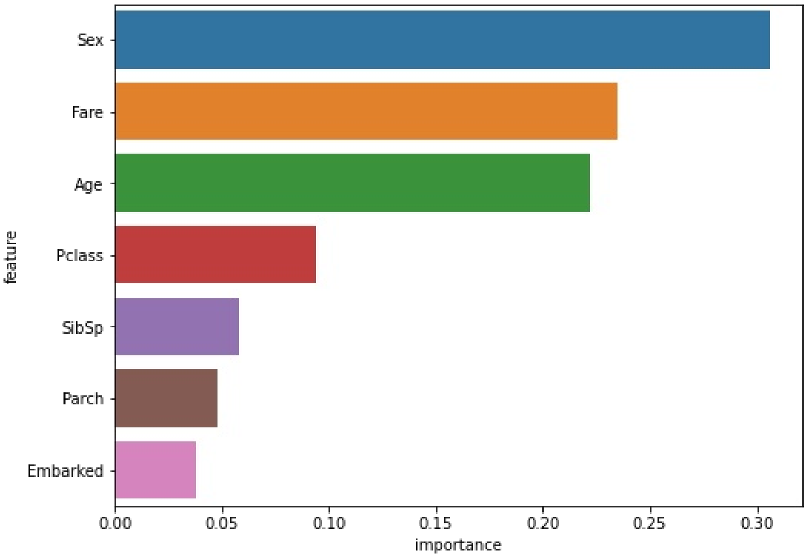
```
#Important columns in model
from sklearn.ensemble import RandomForestClassifier
radm_clf=RandomForestClassifier(max_depth=10,n_estimators=40)
radm_clf.fit(X_train,y_train)
feature_rank=pd.DataFrame({'feature':X_train.columns,'importance':radm_clf.feature_importances_})
```

In [97]:

```
feature_rank=feature_rank.sort_values('importance',ascending=False)
plt.figure(figsize=(8,6))
sns.barplot(y='feature',x='importance',data=feature_rank)
```

Out[97]:

```
<AxesSubplot:xlabel='importance', ylabel='feature'>
```



In [98]:

```
#Using RandomForestClassifier method
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
tuned_parameters=[{'max_depth':[10,20,30,40],'n_estimators':[10,20,30,40,50,60,70,80,90,100]}]
radm_clf=RandomForestClassifier()
clf_grid=GridSearchCV(radm_clf,tuned_parameters,cv=5,scoring='accuracy')
clf_grid.fit(X_train,y_train)
```

Out[98]:

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid=[{'max_depth': [10, 20, 30, 40],
                          'n_estimators': [10, 20, 30, 40, 50, 60, 70, 80, 90,
                                           100]}],
             scoring='accuracy')
```

In [99]:

```
clf_grid.best_params_
```

Out[99]:

```
{'max_depth': 10, 'n_estimators': 90}
```

In [100…

```
clf_grid.best_score_*100
```

Out[100…

```
83.78580645161291
```

In [101…

In [101…

```
#FInal model using best params
radm_clf=RandomForestClassifier(max_depth=10,n_estimators=10,max_features='auto')
radm_clf.fit(X_train,y_train)
```

Out[101…

```
RandomForestClassifier(max_depth=10, n_estimators=10)
```

In [102…

```
y_pred=radm_clf.predict(X_test)
rf=accuracy_score(y_test,y_pred)*100
rf
```

Out[102…

```
79.47761194029852
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js