Feature Selection By Using The Chi2 Test.

- High diamensional data in input usualy not good for classification due to curse of the diamensionality .It significantly increases the time and space complexity for processing the data.
- Lets suppose you having huge dimension of your data then definetly it will take more time and more space to test and train a model moreover in presence of irrelevant and redudant features learning method tends to overfit and becomes less interpretable.A common way to resolve this problem is a feature selection which reduces the diamensionality by selection the subset of feature from the input feature set and it is often used to reduce the computational cost and remove irrrevant and redudant features for the problem with the high diamensional data.
- One of the method in filter method is a Fisher Score which can be calculated by doing the chi2(chi sqaure).

- **Fisher score use on numerical and chi2 method is apply only on the categorical data.**
- That categorical data that has to be a finite set of the data so the basical it computes the distributions and its frequency.
- Distribution and frequncy is basically define accordingly with the help of mode and median.
- So the Fisher Score is one of the most widely used supervised feature selection method however it select each features independently accordingly their score and the fisher criteria which can lead suboptimal subset of the features.
- In univaraint selection that the feature are been selected individualy by considering the effect of other features we may endup by selecting a suboptimal feature set.

What is the Fisher Score and Chi2(x2) Test.

- Chi2(x2) Test :- It is a stastical hypothesis test where the sampling distribution of the stastic of the chi2 distribution and chi2 test is used to determine whether there is significant diffrence between expected frequency and observed frequency in one or more categories thats means one or more features.It is applied only and only categorical datasets.
- Chi2 test is measure the diffrences between the stochastic variable,so using this function "Weeds out" the features that are most likely to be independed of the class and therefore irrelevant for classifiaction.
- That we are removing those feature which are the irrevant for the classification or not depended to the target output.

Understand With an Example.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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

In [2]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel,chi2,SelectKBest,SelectPercentile
from sklearn.ensemble import RandomForestClassifier
```

In [3]:
```
from sklearn import datasets
```

In [4]:
```
data = sns.load_dataset('titanic')
```

In [5]:
```
data.head()
```

Out[5]:

	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

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
data.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: 80.6+ KB
```

In [8]:
```
data.describe()
```

Out[8]:

	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

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
```

In [10]:
```
data['age'].mean()
```

Out[10]:
```
29.69911764705882
```

In [11]:
```
data['age'].median()
```

Out[11]:
```
28.0
```

In [12]:
```
data['age'].fillna(data['age'].median(),inplace=True)
```

In [13]:
```
data['deck'].value_counts()
```

Out[13]:
```
C    59
B    47
D    33
E    32
A    15
F    13
G     4
Name: deck, dtype: int64
```

In [14]:
```
data['deck'].isnull().sum()/len(data.index)*100
```

Out[14]:
```
77.21661054994388
```

In [15]:
```
del data['deck']
```

In [16]:
```
data.head()
```

Out[16]:

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

In [17]:
```
data['embarked'].value_counts()
```

Out[17]:
```
S    644
C    168
Q     77
Name: embarked, dtype: int64
```

In [18]:
```
data['embarked'].fillna('S',inplace=True)
```

In [19]:
```
lable = LabelEncoder()
data['sex']=lable.fit_transform(data['sex'])
```

In [20]:
```
d = {'S':0,'C':1,'Q':2}

data['embarked']=data['embarked'].map(d)
```

In [21]:
```
xa = {'First':0,'Second':1,'Third':2}
data['class']=data['class'].map(xa)
```

In [22]:
```
lable = LabelEncoder()
data['who']=lable.fit_transform(data['who'])
```

In [23]:
```
lable = LabelEncoder()
data['who']=lable.fit_transform(data['who'])
```

In [24]:
```
lable = LabelEncoder()
data['adult_male']=lable.fit_transform(data['adult_male'])
```

In [25]:
```
lable = LabelEncoder()
data['alive']=lable.fit_transform(data['alive'])
```

In [26]:
```
lable = LabelEncoder()
data['alone']=lable.fit_transform(data['alone'])
```

In [27]:
```
data.head()
```

Out[27]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	1	22.0	1	0	7.2500	0	2	1	1	Southampton	0	0
1	1	1	0	38.0	1	0	71.2833	1	0	2	0	Cherbourg	1	0
2	1	3	0	26.0	0	0	7.9250	0	2	2	0	Southampton	1	1
3	1	1	0	35.0	1	0	53.1000	0	0	2	0	Southampton	1	0
4	0	3	1	35.0	0	0	8.0500	0	2	1	1	Southampton	0	1

In [28]:
```
data['embark_town'].value_counts()
```

Out[28]:
```
Southampton    644
Cherbourg      168
Queenstown      77
Name: embark_town, dtype: int64
```

In [29]:
```
data['embark_town'].isnull().sum()
```

Out[29]:
```
2
```

In [30]:
```
data['embark_town'].fillna('Southampton',inplace=True)
```

In [31]:
```
data['embark_town']=lable.fit_transform(data['embark_town'])
```

In [32]:
```
data.head()
```

Out[32]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	1	22.0	1	0	7.2500	0	2	1	1	2	0	0
1	1	1	0	38.0	1	0	71.2833	1	0	2	0	0	1	0
2	1	3	0	26.0	0	0	7.9250	0	2	2	0	2	1	1
3	1	1	0	35.0	1	0	53.1000	0	0	2	0	2	1	0
4	0	3	1	35.0	0	0	8.0500	0	2	1	1	2	0	1

In [33]:
```
x = data.drop(columns=['alive'])
y = data['alive']
```

In [34]:
```
x.shape,y.shape
```

Out[34]:
```
((891, 13), (891,))
```

In [35]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,stratify=y,random_state=42)
```

In [36]:
```
x_train.shape,y_train.shape
```

Out[36]:
```
((712, 13), (712,))
```

Chi2_test

In [40]:
```
f_score,pvalue = chi2(x_train,y_train) #it will return us the f_score and pvalues
f_score
```

Out[40]:
```
array([4.39000000e+02, 2.59119322e+01, 7.63019940e+01, 2.08599242e+01,
       1.10666572e+00, 9.06416124e+00, 3.91708928e+03, 1.58304323e+01,
       4.57073139e+01, 2.24512976e+01, 8.93946763e+01, 1.06126527e+01,
       1.29187387e+01])
```

In [41]:
```
pvalue
```

Out[41]:
```
array([1.78683855e-97, 3.57353388e-07, 2.43437961e-18, 4.94125395e-06,
       2.92807896e-01, 2.60668078e-03, 0.00000000e+00, 6.92792453e-05,
       1.37309133e-11, 2.15539829e-06, 3.23405685e-21, 1.12316440e-03,
       3.25308982e-04])
```

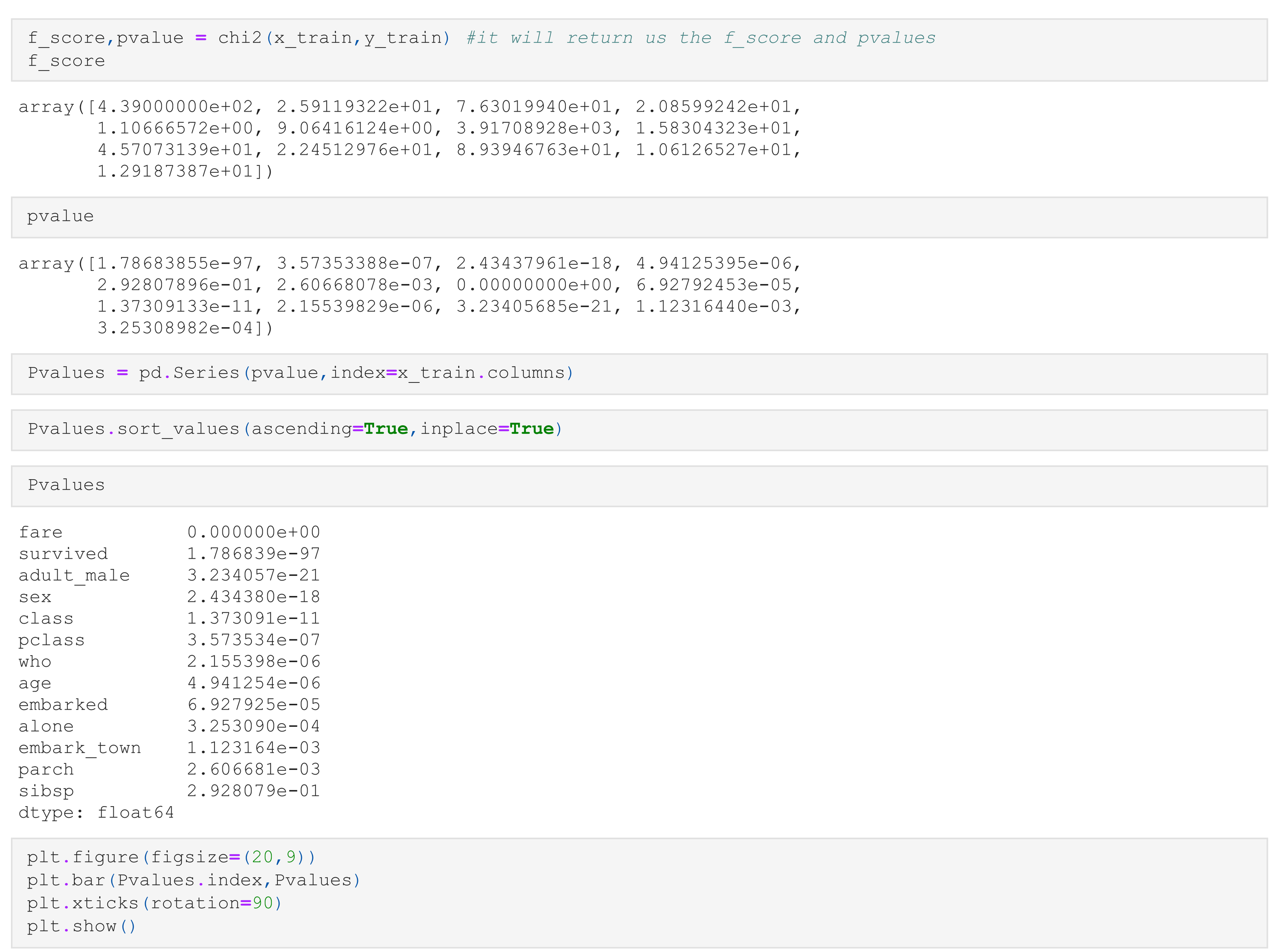
In [42]:
```
Pvalues = pd.Series(pvalue,index=x_train.columns)
```

In [43]:
```
Pvalues.sort_values(ascending=True,inplace=True)
```

In [44]:
```
Pvalues
```

Out[44]:
```
fare           0.000000e+00
survived       1.786839e-97
adult_male     3.234057e-21
sex            2.434380e-18
class          1.373091e-11
pclass         3.573534e-07
who            2.155398e-06
age            4.941254e-06
embarked       6.927925e-05
alone          3.253090e-04
embark_town    1.123164e-03
parch          2.606681e-03
sibsp          2.928079e-01
dtype: float64
```

In [45]:
```
plt.figure(figsize=(20,9))
plt.bar(Pvalues.index,Pvalues)
plt.xticks(rotation=90)
plt.show()
```



If We see the Pvalues according Feature.Those which having the low pvalus which are highly important for us to get the maximum prediction.

In [55]:
```
x_train_6 = x_train[['fare','survived','adult_male','sex','class','pclass','age','alone','who','embark_town']]
x_test_6  = x_test[['fare','survived','adult_male','sex','class','pclass','age','alone','who','embark_town']]
```

In [56]:
```
x_train_6.shape,x_test_6.shape
```

Out[56]:
```
((712, 10), (179, 10))
```

In [57]:
```
def RandomForest(x_train,x_test,y_train,y_test):
    clf = RandomForestClassifier(n_estimators=100,)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print('Accuracy :-',accuracy_score(y_test,y_pred))
```

In [58]:
```
RandomForest(x_train_6,x_test_6,y_train,y_test)
```

```
Accuracy :- 1.0
```

Thanks !!!