In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

df = pd.DataFrame({"F1":[1,2,4,1,2,4],
                   "F2":[4,5,6,7,8,9],
                   "F3":[0,0,0,0,0,0],
                   "F4":[1,1,1,1,1,1]})
df
```

Out[1]:

	F1	F2	F3	F4
0	1	4	0	1
1	2	5	0	1
2	4	6	0	1
3	1	7	0	1
4	2	8	0	1
5	4	9	0	1

In [2]:
```
from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=0)
var_thres.fit(df)
```

Out[2]:
```
VarianceThreshold(threshold=0)
```

In [3]:
```
var_thres.get_support()
```

Out[3]:
```
array([ True,  True, False, False])
```

In [4]:
```
df.columns[var_thres.get_support()]
```

Out[4]:
```
Index(['F1', 'F2'], dtype='object')
```

In [5]:
```
#importing libraries
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:
```
#Loading the dataset
df = load_boston()
x = pd.DataFrame(df.data, columns = df.feature_names)
y = df.target
```

In [7]:
```
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=0)

X_train.shape, X_test.shape
```

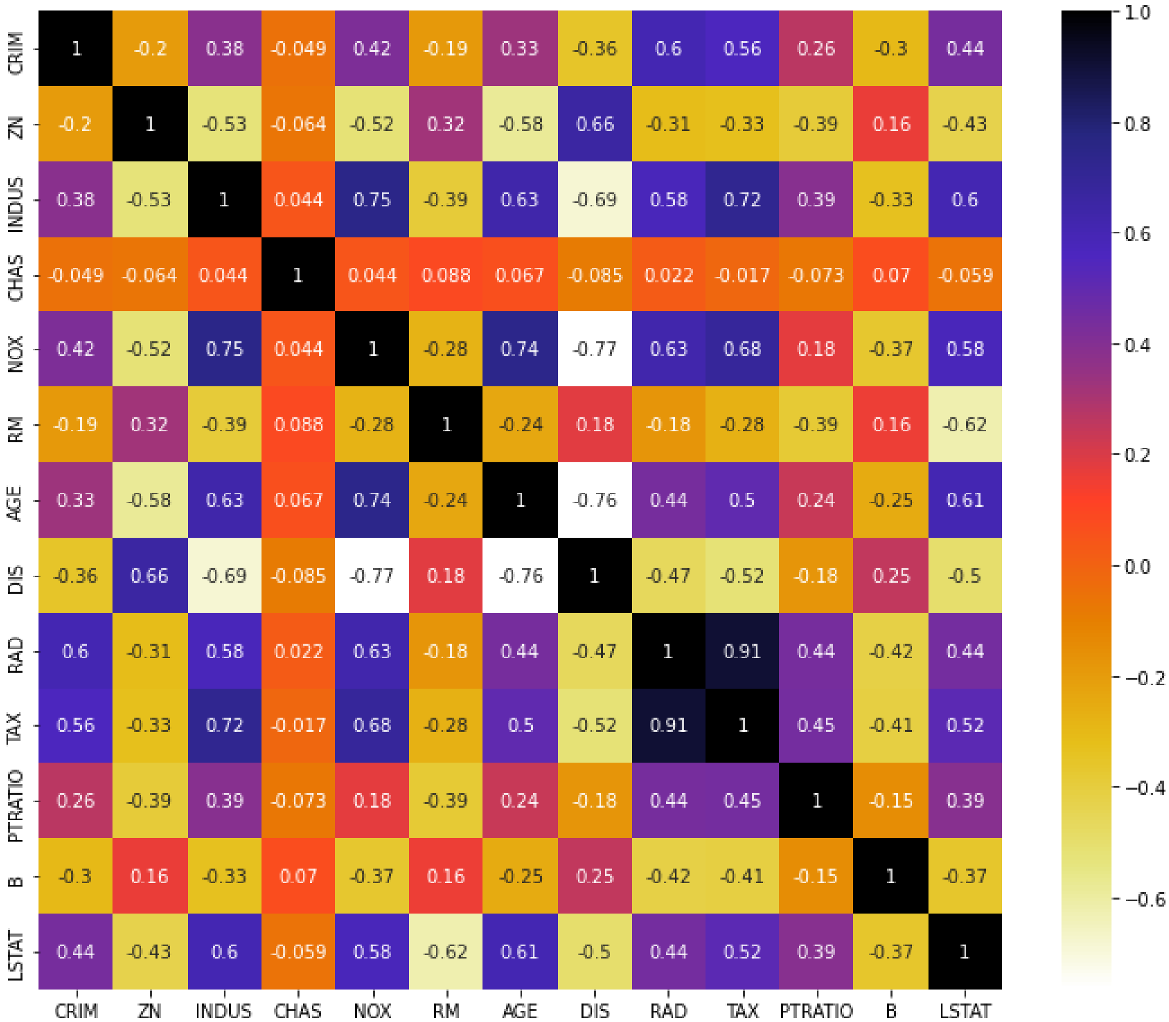
Out[7]: ((354, 13), (152, 13))

In [8]:
```
X_train.corr()
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS
CRIM	1.000000	-0.196172	0.382073	-0.049364	0.416560	-0.188280	0.329927	-0.355840
ZN	-0.196172	1.000000	-0.529392	-0.063863	-0.523572	0.319260	-0.583885	0.658331
INDUS	0.382073	-0.529392	1.000000	0.044224	0.750218	-0.392969	0.629257	-0.686848
CHAS	-0.049364	-0.063863	0.044224	1.000000	0.043748	0.088125	0.067269	-0.085492
NOX	0.416560	-0.523572	0.750218	0.043748	1.000000	-0.279202	0.740052	-0.765753
RM	-0.188280	0.319260	-0.392969	0.088125	-0.279202	1.000000	-0.235839	0.183857
AGE	0.329927	-0.583885	0.629257	0.067269	0.740052	-0.235839	1.000000	-0.761543
DIS	-0.355840	0.658331	-0.686848	-0.085492	-0.765753	0.183857	-0.761543	1.000000
RAD	0.603880	-0.314833	0.578459	0.022338	0.627188	-0.179242	0.440578	-0.467653
TAX	0.560570	-0.327834	0.719038	-0.017156	0.683445	-0.275242	0.502429	-0.519643
PTRATIO	0.264780	-0.392838	0.388353	-0.072683	0.179046	-0.385526	0.239729	-0.176620
B	-0.299525	0.164641	-0.331638	0.069682	-0.369445	0.157459	-0.250416	0.248376
LSTAT	0.439369	-0.429178	0.603374	-0.059060	0.577154	-0.623920	0.606530	-0.501780

In [9]:
```
import seaborn as sns
#Using Pearson Correlation
plt.figure(figsize=(12,10))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap_r)
plt.show()
```



In [10]:
```
# with the following function we can select highly correlated features
# it will remove the first feature that is correlated with anything other feat

def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr
```

In [11]:
```
corr_features = correlation(X_train, 0.7)
len(set(corr_features))
```

Out[11]: 4

In [12]:
```
corr_features
```

Out[12]: {'AGE', 'DIS', 'NOX', 'TAX'}

In [13]:
```
X_train.drop(corr_features,axis=1)
X_test.drop(corr_features,axis=1)
```

Out[13]:

	CRIM	ZN	INDUS	CHAS	RM	RAD	PTRATIO	B	LSTAT
329	0.06724	0.0	3.24	0.0	6.333	4.0	16.9	375.21	7.34
371	9.23230	0.0	18.10	0.0	6.216	24.0	20.2	366.15	9.53
219	0.11425	0.0	13.89	1.0	6.373	5.0	16.4	393.74	10.50
403	24.80170	0.0	18.10	0.0	5.349	24.0	20.2	396.90	19.77
78	0.05646	0.0	12.83	0.0	6.232	5.0	18.7	386.40	12.34
...	...	...	...	...	...	...	...	...	...
4	0.06905	0.0	2.18	0.0	7.147	3.0	18.7	396.90	5.33
428	7.36711	0.0	18.10	0.0	6.193	24.0	20.2	96.73	21.52
385	16.81180	0.0	18.10	0.0	5.277	24.0	20.2	396.90	30.81
308	0.49298	0.0	9.90	0.0	6.635	4.0	18.4	396.90	4.54
5	0.02985	0.0	2.18	0.0	6.430	3.0	18.7	394.12	5.21

152 rows × 9 columns

In [14]:
```
import seaborn as sns
import numpy as np
df=sns.load_dataset('titanic')
df.head()
```

Out[14]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [15]:
```
df.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: 80.7+ KB
```

In [16]:
```
##['sex','embarked','alone','pclass','Survived']
df=df[['sex','embarked','alone','pclass','survived']]
df.head()
```

Out[16]:

	sex	embarked	alone	pclass	survived
0	male	S	False	3	0
1	female	C	False	1	1
2	female	S	True	3	1
3	female	S	False	1	1
4	male	S	True	3	0

In [17]:
```
from sklearn.preprocessing import LabelEncoder
```

In [18]:
```
le = LabelEncoder()

df['embarked']=le.fit_transform(df['embarked'])
df['alone'] = le.fit_transform(df['alone'])
df['sex'] = le.fit_transform(df['sex'])

df
```

Out[18]:

	sex	embarked	alone	pclass	survived
0	1	2	0	3	0
1	0	0	0	1	1
2	0	2	1	3	1
3	0	2	0	1	1
4	1	2	1	3	0
...	...	...	...	...	...
886	1	2	1	2	0
887	0	2	1	1	1
888	0	2	0	3	0
889	1	0	1	1	1
890	1	1	1	3	0

891 rows × 5 columns

In [19]:
```
x = df.iloc[:,:-1]
y= df.iloc[:,-1]
```

In [20]:
```
## Perform chi2 test
### chi2 returns 2 values
### Fscore and the pvalue
from sklearn.feature_selection import chi2
f_p_values=chi2(x,y)

f_p_values
```

Out[20]:
```
(array([92.70244698,  9.75545583, 14.64079273, 30.87369944]),
 array([6.07783826e-22, 1.78791305e-03, 1.30068490e-04, 2.75378563e-08]))
```

In [21]:
```
import pandas as pd
p_values=pd.Series(f_p_values[0])
p_values.index=x.columns
p_values
```

Out[21]:
```
sex          92.702447
embarked      9.755456
alone        14.640793
pclass       30.873699
dtype: float64
```

In [22]:
```
p_values.sort_index(ascending=False)
```

Out[22]:
```
sex         92.702447
pclass      30.873699
embarked     9.755456
alone       14.640793
dtype: float64
```

In [ ]: