```
# https://www.kaggle.com/iabhishekofficial/mobile-price-classification#train.csv
# https://www.kaggle.com/azzion/svm-for-beginners-tutorial
# https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/
```

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
%matplotlib inline
```

In [2]:

```
df = pd.read_csv('mobileprice.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	0
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	1
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	1
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	1
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	1

5 rows × 21 columns

In [4]:

```
df.isnull().sum()
```

Out[4]:

```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [5]:

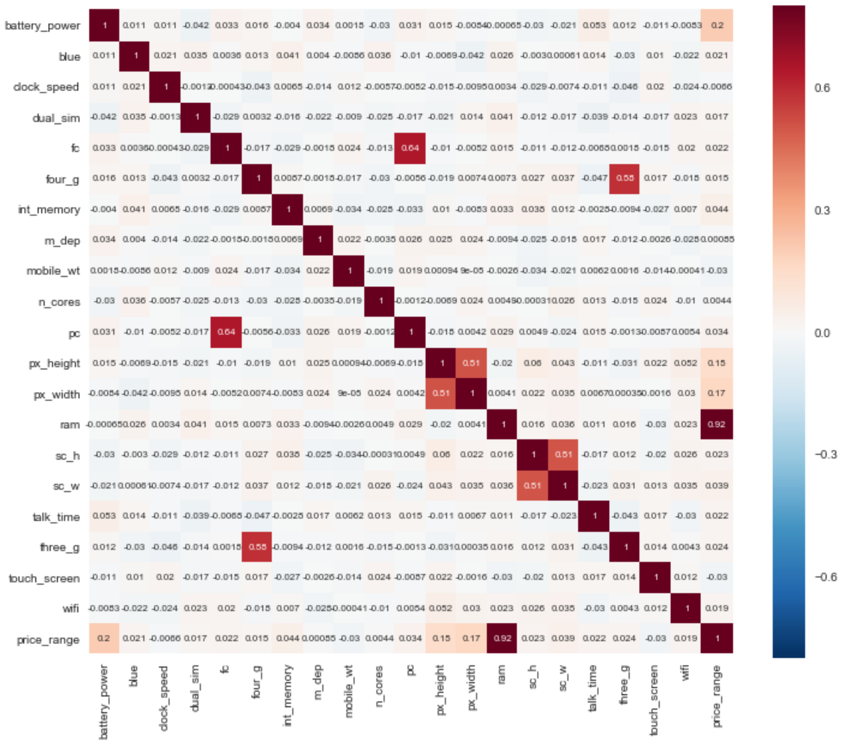
```
df['price_range'].unique()
```

Out[5]:

```
array([1, 2, 3, 0], dtype=int64)
```

In [6]:
```
corrmat = df.corr()
f,ax = plt.subplots(figsize=(12,10))
sns.heatmap(corrmat,vmax=0.8,square=True,annot=True,annot_kws={'size':8})
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x235a04c8f98>
```



In [7]:
```
X = df.drop(['price_range',],axis=1)
X.head()
```

Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1

In [8]:
```
y = df.price_range
y.head()
```

Out[8]:
```
0    1
1    2
2    2
3    2
4    1
Name: price_range, dtype: int64
```

In [9]:
```
#training and testing data:
from sklearn.model_selection import train_test_split
```

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
```

In [11]:
```
X_train.shape
```

Out[11]:
```
(1400, 20)
```

In [12]:
```
X_test.shape
```

Out[12]:
```
(600, 20)
```

In [13]:
```
from sklearn.svm import SVC
```

In [23]:
```
model = SVC(kernel='linear')
```

In [24]:
```
model.fit(X_train,y_train)
```

Out[24]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [25]:

```
y_pred = model.predict(X_test)
```

In [26]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

Out[26]:

```
0.98
```

In [27]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
             precision    recall  f1-score   support

          0       1.00      0.99      1.00       144
          1       0.97      0.97      0.97       147
          2       0.96      0.98      0.97       170
          3       0.99      0.98      0.99       139

avg / total       0.98      0.98      0.98       600
```

```
#Parameters:
C -> for bias variance tradeoff.
large C gives low bias and high variace.
```

```
The most popular kernel functions are :

the linear kernel
the polynomial kernel
the RBF (Gaussian) kernel
the string kernel
```

```
Linear kernel works well with linearly separable data
```

```
https://data-flair.training/blogs/svm-kernel-functions/
```

In [ ]:

```

```