In [1]:

```
import numpy as np
import pandas as pd
import sklearn

from sklearn import tree
from pandas import *

df = pd.read_csv("movie_metadata.csv")
df =df.dropna()
```

In [2]:

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer(min_df=1)
```

In [3]:

```
from sklearn.preprocessing import LabelEncoder
X = pd.DataFrame()
df = pd.read_csv("movie_metadata.csv")
df = df.dropna()


columnsToEncode = list(df.select_dtypes(include=['category','object']))
le = LabelEncoder()
for feature in columnsToEncode:
    try:
        df[feature] = le.fit_transform(df[feature])
    except:
        print('Error encoding ' + feature)
df.head()
```

Out[3]:

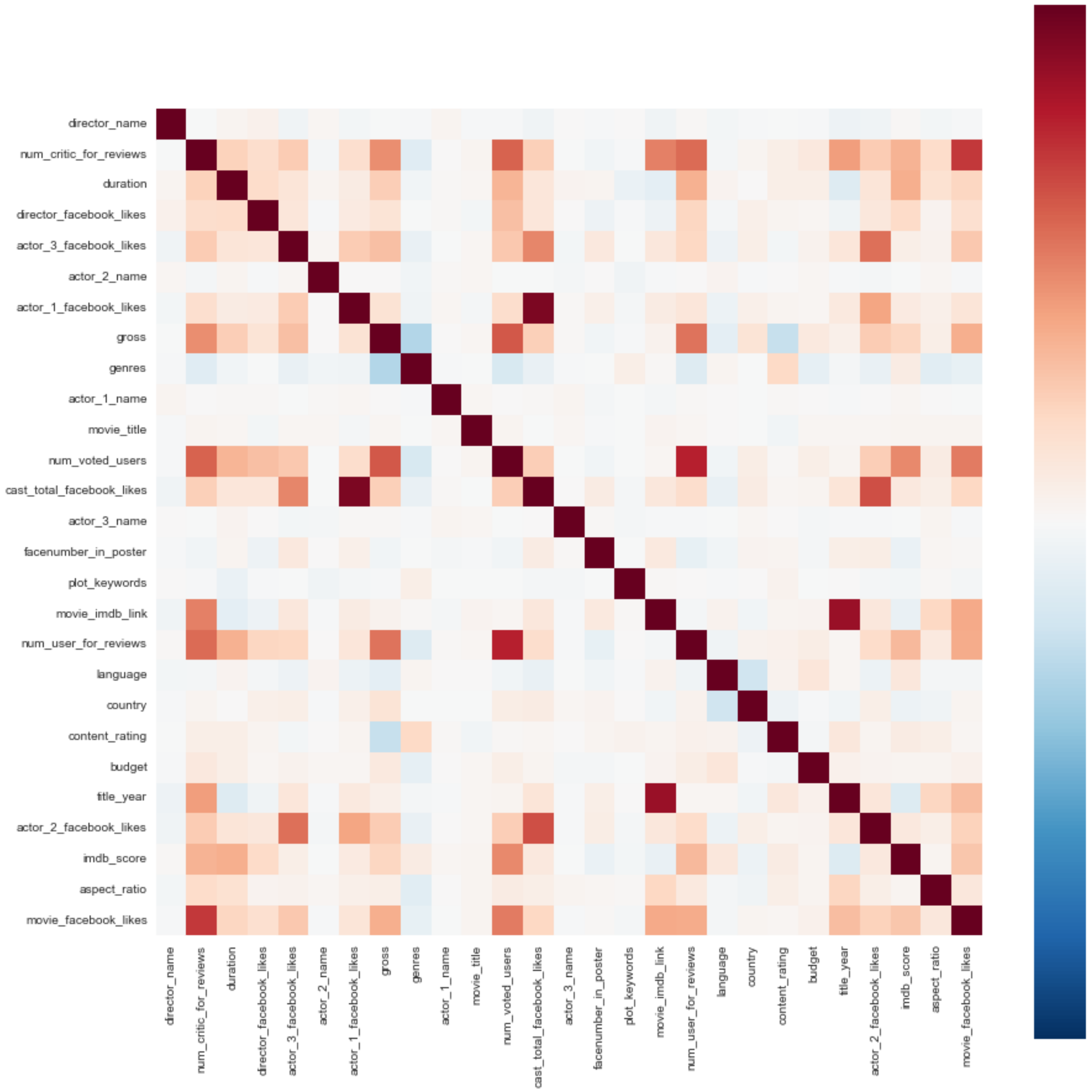
	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	acto
0	1	620	723.0	178.0	0.0	855.(
1	1	538	302.0	169.0	563.0	1000
2	1	1395	602.0	148.0	0.0	161.(
3	1	251	813.0	164.0	22000.0	2300
5	1	62	462.0	132.0	475.0	530.(

5 rows × 28 columns

In [4]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

corr = df.select_dtypes(include = ['float64', 'int64']).iloc[:, 1:].corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr, vmax=1, square=True)
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2923357ba90>
```



In [5]:

```
X=df
y=X['imdb_score']
#y.apply(np.round)
X = X.drop(['imdb_score'], axis = 1)

from sklearn.cross_validation import train_test_split
from sklearn.preprocessing import StandardScaler

scaler=StandardScaler()
X = scaler.fit_transform(X)
y = np.array(y).astype(int)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random
_state =190)
```

Linear and Logistic Regression

In [28]:

```
from sklearn.linear_model import LogisticRegression
LRL2 = LogisticRegression(penalty = 'l2')
LRL2.fit(X_train,y_train)
L2score = LRL2.score(X_test,y_test)
print (L2score)
```

```
0.482697426797
```

In [25]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=4,
normalize=True)
model.fit(X_train,y_train)
print('Accuracy: ', model.score(X_test, y_test))
```

```
Accuracy:  0.371803254538
```

Decision Tree

In [18]:

```
#predict and score
from sklearn import tree
tree_model = tree.DecisionTreeClassifier(max_depth = 11, min_samples_split=90)
tree_model.fit(X_train, y_train)
tree_model.score(X_test, y_test)
```

Out[18]:

```
0.50044365572315885
```

SVM

In [29]:

```
from sklearn import svm

clf = svm.SVC()
clf.set_params(C=10)
clf.fit(X_train, y_train)
clf.predict(X_test)

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
prediction = clf.predict(X_test)
print(accuracy_score(y_test,prediction))
```

```
0.519077196096
```

Linear SVM

In [37]:

```
clfLin = svm.SVC(kernel = 'linear')
clfLin.set_params(C=0.5)
clfLin.fit(X_train,y_train)
clfLin.predict(X_test)

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
prediction = clfLin.predict(X_test)
print(accuracy_score(y_test,prediction))
```

```
0.512866015972
```

KNN

In [34]:

```
from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n_neighbors=12)

neigh.fit(X_train, y_train)
neigh.set_params(p=7)
neigh.predict(X_test)
accuracy_score(y_test, neigh.predict(X_test))
```

Out[34]:

```
0.42236024844720499
```

Random Forest

```
In [35]: from sklearn.ensemble import RandomForestClassifier

         clf = RandomForestClassifier(n_estimators=28)
         clf = clf.fit(X_train, y_train)
         accuracy_score(y_test,clf.predict(X_test))

Out[35]: 0.54835847382431235
```

```
In [ ]:
```