Random Forest, Decision Tree, Gradient Boosting

First of all, let's load the dataset and look at its shape.

In [1]:
```
def prettify_ax(ax):
    ax.spines['right'].set_visible(False)
    ax.spines['top'].set_visible(False)
    ax.spines['left'].set_visible(False)
    ax.spines['bottom'].set_visible(False)
    ax.set_frameon=True
    ax.patch.set_facecolor('#eeeeef')
    ax.grid('on', color='w', linestyle='-', linewidth=1)
    ax.tick_params(direction='out')
    ax.set_axisbelow(True)
```

In [2]:
```
import pandas as pd
import numpy as np
import ast
```

In [3]:

```
data = pd.read_csv('Meta_data_First_All_Cleaned.csv')

data['Action'] = np.zeros(len(data))
data['Adventure'] = np.zeros(len(data))
data['Animation'] = np.zeros(len(data))
data['Comedy'] = np.zeros(len(data))
data['Crime'] = np.zeros(len(data))
data['Documentary'] = np.zeros(len(data))
data['Drama'] = np.zeros(len(data))
data['Family'] = np.zeros(len(data))
data['Fantasy'] = np.zeros(len(data))
data['History'] = np.zeros(len(data))
data['Horror'] = np.zeros(len(data))
data['Music'] = np.zeros(len(data))
data['Mystery'] = np.zeros(len(data))
data['Romance'] = np.zeros(len(data))
data['Science Fiction'] = np.zeros(len(data))
data['Thriller'] = np.zeros(len(data))
data['War'] = np.zeros(len(data))
data['Western'] = np.zeros(len(data))

cols = ['Action', 'Adventure', 'Animation', 'Comedy', 'Crime', 'Document
ary', 'Drama', 'Family', 'Fantasy', 'History',
        'Horror', 'Music', 'Mystery', 'Romance', 'Science Fiction', 'Thr
iller', 'War', 'Western']

for col in cols:
    for i in range(len(data)):
        for j in range(len(ast.literal_eval(data.genres[i]))):
            if ast.literal_eval(data.genres[i])[j]['name'] == col:
                data[col][i] = 1
print data.columns

del data['genres']

print 'Size:', data.shape
```

```
/Users/nisreenshiban/anaconda/lib/python2.7/site-packages/ipykernel/__m
ain__.py:29: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy

Index([u'Unnamed: 0', u'budget', u'director', u'genres', u'id', u'keywo
rds',
       u'overview', u'popularity', u'poster_path', u'releaseyear', u're
venue',
       u'runtime', u'title', u'Action', u'Adventure', u'Animation', u'C
omedy',
       u'Crime', u'Documentary', u'Drama', u'Family', u'Fantasy', u'His
tory',
       u'Horror', u'Music', u'Mystery', u'Romance', u'Science Fiction',
       u'Thriller', u'War', u'Western'],
      dtype='object')
Size: (9988, 30)
```

We see, that there are a lot of columns that have NaN values, we need to somehow deal with that.

Let's explore what columns have NaN values.

In [4]:
```
is_null_column = pd.isnull(data).any()
pd.DataFrame(is_null_column[is_null_column == True].index, columns=['Col
umn name'])
```

Out[4]:

	Column name
0	keywords
1	overview

As we see, two columns have NaN values. Let's see what percentage of values is NaN.

In [5]:
```
print round(100 * pd.isnull(data).values.sum() / float(data.shape[0] * d
ata.shape[1]), 1)
```

```
0.8
```

As we see, a small amount of data is missing. We would need to deal with that later, when we use some model to do prediction. For now, let's keep them.

Let's now look at distributions of values in each column.

As a start, let's draw a histogram for each column and then dig into features that are the most interesting/representative.

In [6]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [7]:
```
%%javascript
IPython.OutputArea.auto_scroll_threshold = 9999; // Prevent ipython from
 suppresing output.
```

We'll start by looking at numerical features.

In [8]:
```
data_n = data.copy()
data_n = data_n.loc[:,:'title']
```

In [9]:
```
idx = 0
i = 1

plt.figure(figsize=(20, 60))
plt.subplots_adjust(hspace=0.2, wspace=0.2)

for column in data_n.columns:
    nonull_rows = data[column].notnull()
    nan_percentage = 100 * (1 - np.sum(nonull_rows) /
float(data.shape[0]))
    if (data.dtypes[idx] == np.int64 or data.dtypes[idx] == np.float64):
        ax = plt.subplot(10, 2, i)

        plt.title(column + '(' + str(int(nan_percentage)) + '% of
NaNs)');
        ax.hist(data[column][nonull_rows], color = '#c0392b', alpha = 0.
7);
        prettify_ax(ax)
        i += 1
    idx += 1
plt.show()
```

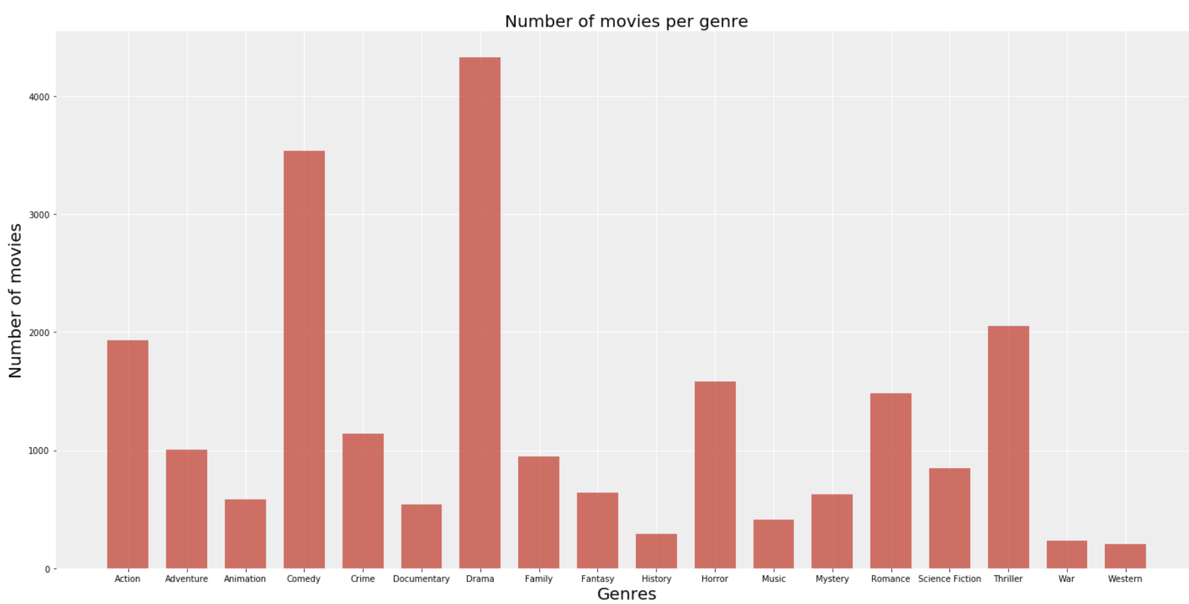
On the other hand, some features mostly contain a single value - those would probably not be informative. But there are outliers that need to be kept in mind.

As we're trying to predict genres, let's look at them closer.

In [10]:
```
genres_cols = []

for i in cols:
    genres_cols.append(sum(data[i]))
```

In [11]:
```
n_groups = len(cols)

fig, ax = plt.subplots(figsize=(20, 10))

index = np.arange(n_groups)
bar_width = 0.7

rects1 = plt.bar(index, genres_cols, bar_width, alpha = 0.7,
                 color='#c0392b')
plt.xlabel('Genres', fontsize=20)
plt.ylabel('Number of movies', fontsize=20)
plt.title('Number of movies per genre', fontsize = 20)
plt.xticks(index, cols)
prettify_ax(ax)
plt.legend()

plt.tight_layout()
plt.show()
```

```
/Users/nisreenshiban/anaconda/lib/python2.7/site-packages/matplotlib/ax
es/_axes.py:545: UserWarning: No labelled objects found. Use label
='...' kwarg on individual plots.
  warnings.warn("No labelled objects found. "
```



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

Out[12]: (9988, 30)

To find out what features are relevant, we'll train Random Forest classifier. It provides off-shelf feature importance mechanism that we'll use in order to understand what feaures are most relevant to prediction task.

For simple random forest model, we'll perform one-hot-encoding on categorical features and change all NaNs to column means (or mode for categorical).

In [27]:
```
del data['id']
del data['Unnamed: 0']
```

In [28]:
```
transformed_data = data.copy()
transformed_data = transformed_data.loc[:,:'title']
```

In [29]:
```
transformed_data.columns
```

Out[29]:
```
Index([u'budget', u'director', u'keywords', u'overview', u'popularity',
       u'poster_path', u'releaseyear', u'revenue', u'runtime', u'titl
e'],
      dtype='object')
```

In [30]:
```
idx = -1

drop_columns = []

for column in transformed_data.columns:
    idx += 1
    isnull = np.where(transformed_data[column].isnull())[0]
    if len(isnull) == 0:
        if not (transformed_data.dtypes[idx] == np.float64 or \
            transformed_data.dtypes[idx] == np.int64 or \
            transformed_data.dtypes[idx] == object):
            drop_columns.append(column)
        continue
    if (transformed_data.dtypes[idx] == np.float64 or transformed_data.d
types[idx] == np.int64):
        transformed_data.iloc[isnull, idx] =
np.nanmean(transformed_data[column])
    elif transformed_data.dtypes[idx] == object:
        transformed_data.iloc[isnull, idx] = 'NaN'
    else:
        drop_columns.append(column)
transformed_data.drop(drop_columns, axis=1, inplace=True)
```

We drop all categorical columns that have > threshold_count unique values.

In [31]:
```
count_threshold = 100
columns = transformed_data.columns[transformed_data.dtypes == object]
ohe_columns = filter(lambda x: len(np.unique(transformed_data[x])) < cou
nt_threshold, columns)
remove_columns = filter(lambda x: len(np.unique(transformed_data[x])) >=
 count_threshold, columns)


transformed_data.drop(remove_columns, axis=1, inplace=True)
transformed_data = pd.get_dummies(transformed_data, columns=ohe_columns)
```

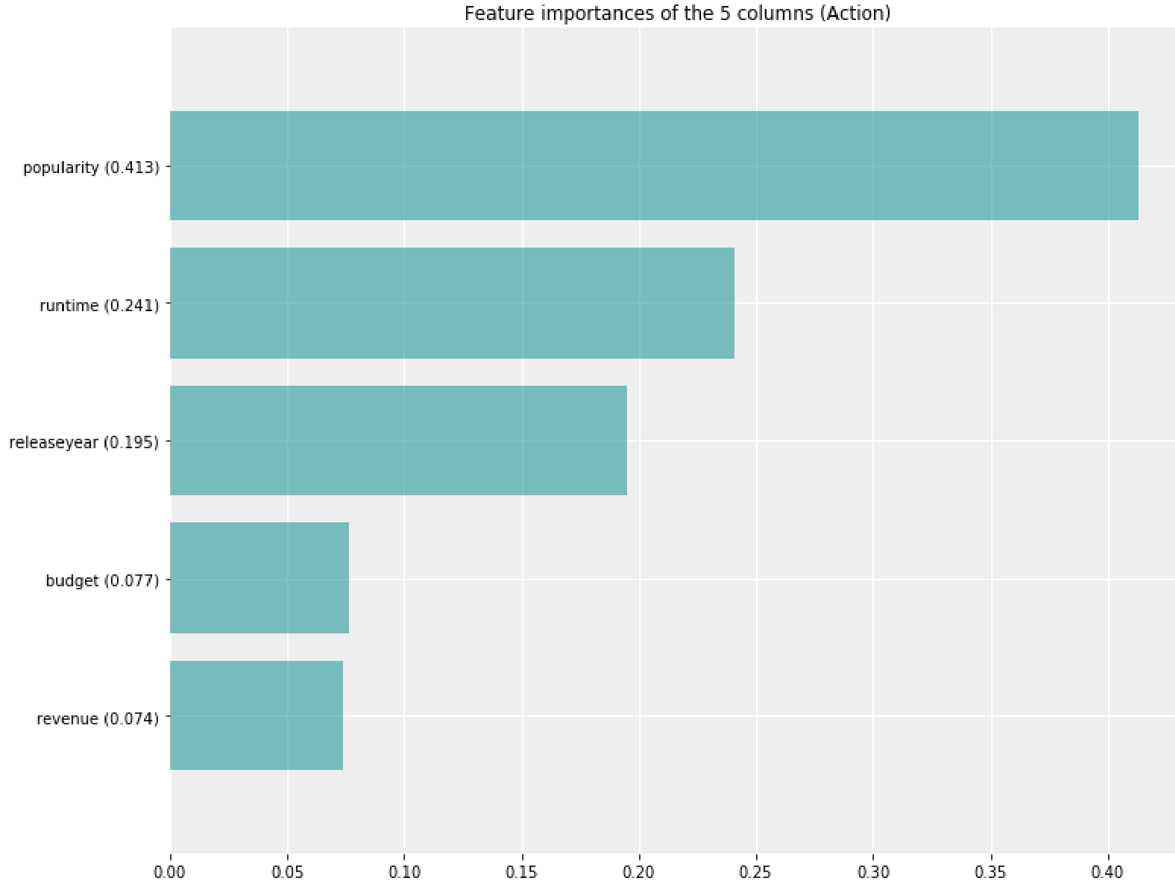
Now we can train random forest on each genre to get feature importances.

Since the dataset is large, we'll take only 10% of it for speed purposes.

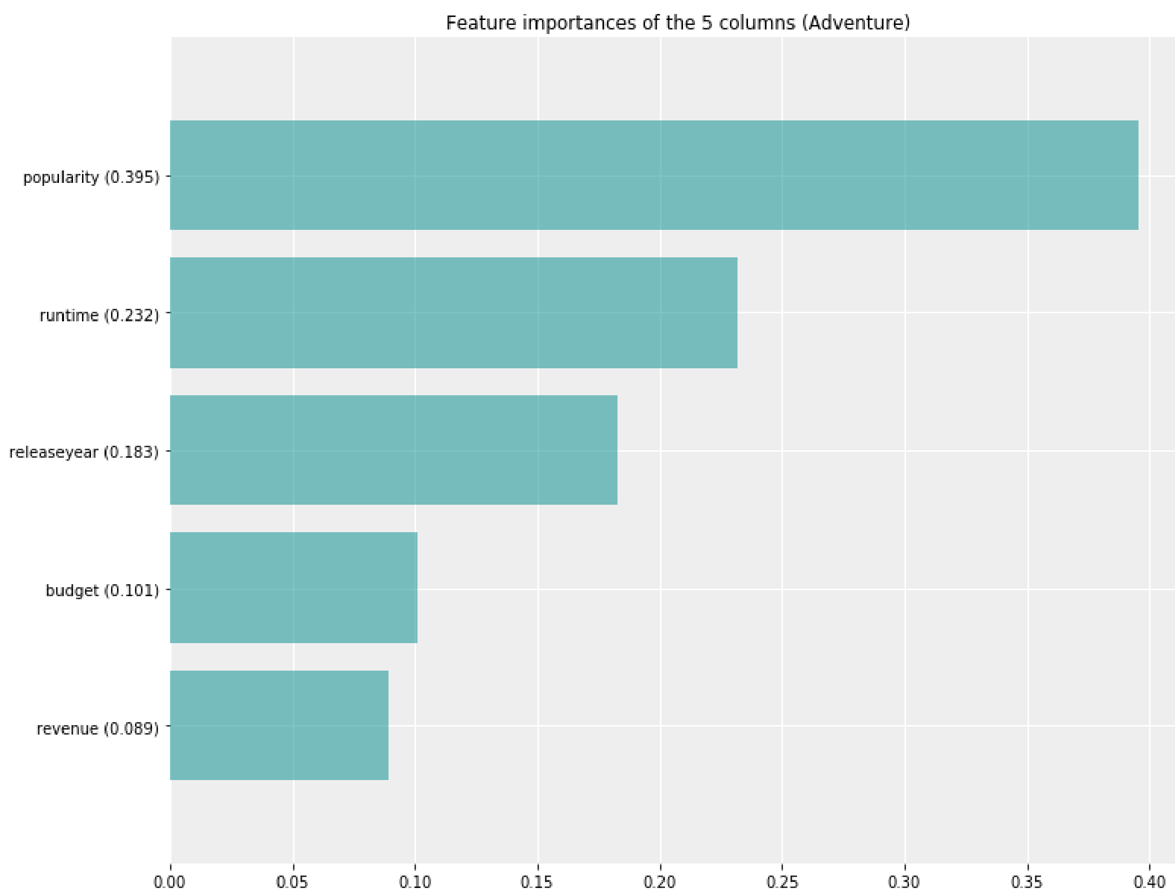
In [32]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.grid_search import GridSearchCV
```

In [34]:
```
for col in cols:
    transformed_y = data[col]

    # Now we can train random forest
    grid_search = GridSearchCV(RandomForestClassifier(n_estimators=100,
random_state=123), \
                            {}, cv=5)

    grid_search.fit(transformed_data, transformed_y)

    print 'The validation accuracy of RF model is: ', grid_search.best_s
core_


    #As we can see, model already works great.
    #At this points, it's not really clear whether we overfit the data
 (since we used KFold as a crossvalidation).
    count = 5
    clf = grid_search.best_estimator_
    importances = clf.feature_importances_
    std = np.std([clf.feature_importances_ for tree in clf.estimators_],
 axis=0)
    indices = np.argsort(importances)[::-1][:count][::-1]

    fig = plt.figure(figsize=(12, 10))
    ax = fig.add_subplot(111)

    plt.title("Feature importances of the " + str(count) + " columns ("
+ col + ")")
    ax.barh(range(count), importances[indices],
           color="darkcyan", alpha= 0.5, yerr=std[indices], align="cente
r")

    column_labels = [transformed_data.columns[idx] + ' (' + str(round(im
portances[idx], 3)) + ')' for idx in indices]

    plt.yticks(range(count), column_labels)
    plt.ylim([-1, count])
    prettify_ax(ax)
    plt.show()
```

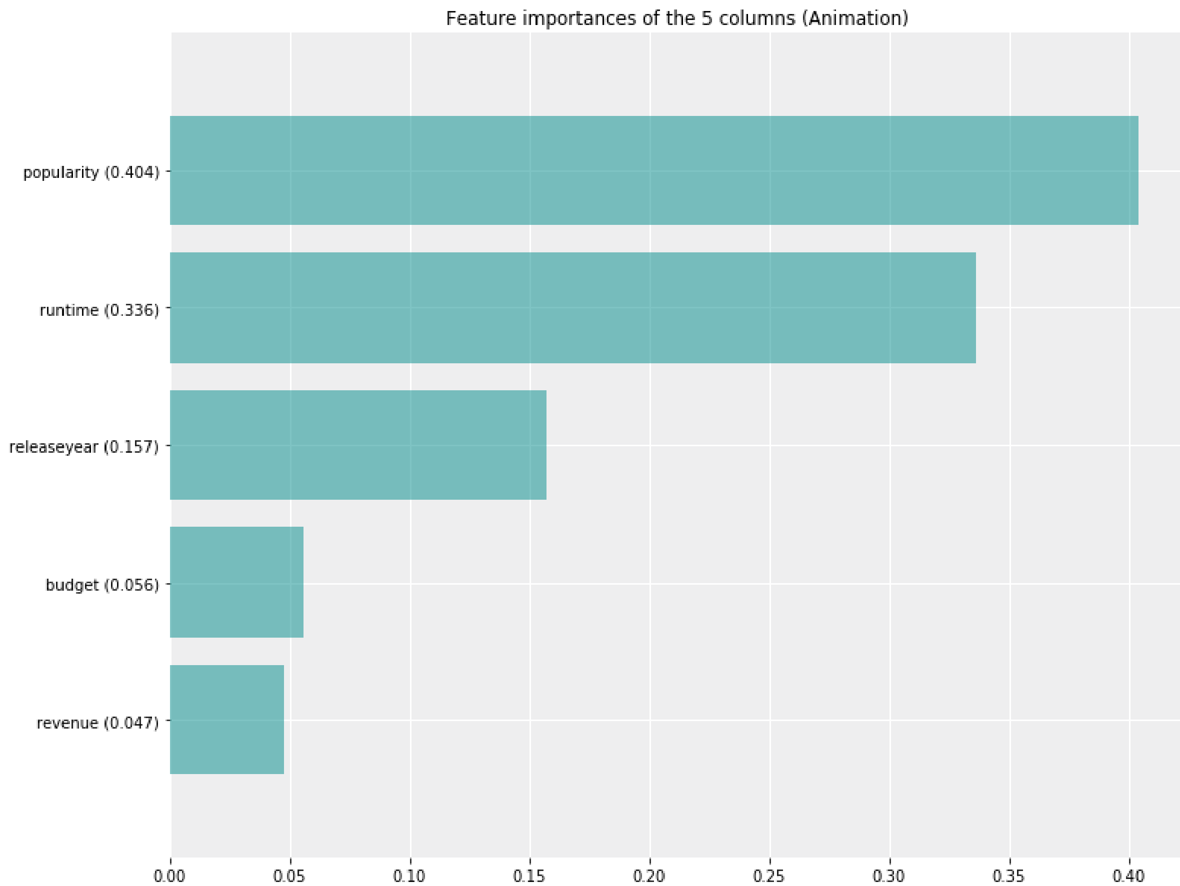
```
The validation accuracy of RF model is:  0.789046856227
```



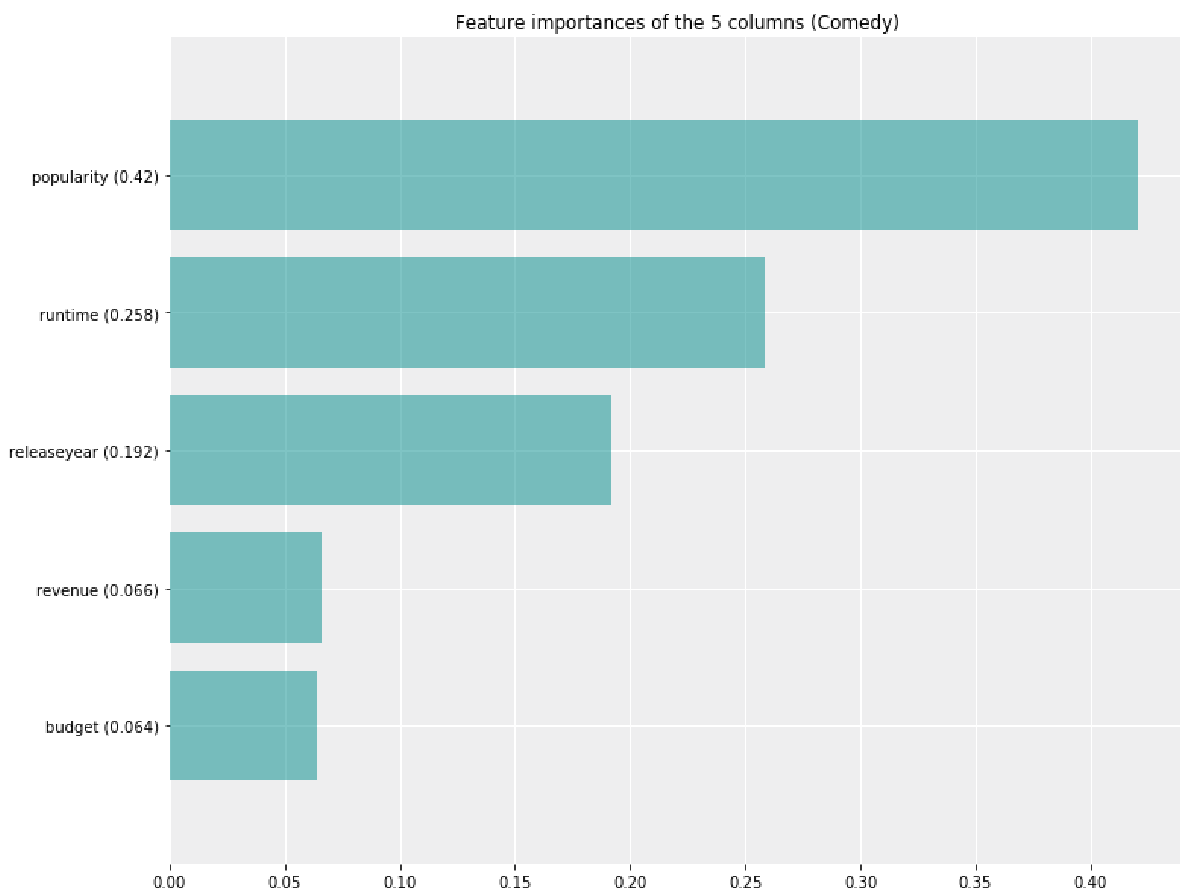

```
The validation accuracy of RF model is:  0.894873848618
```



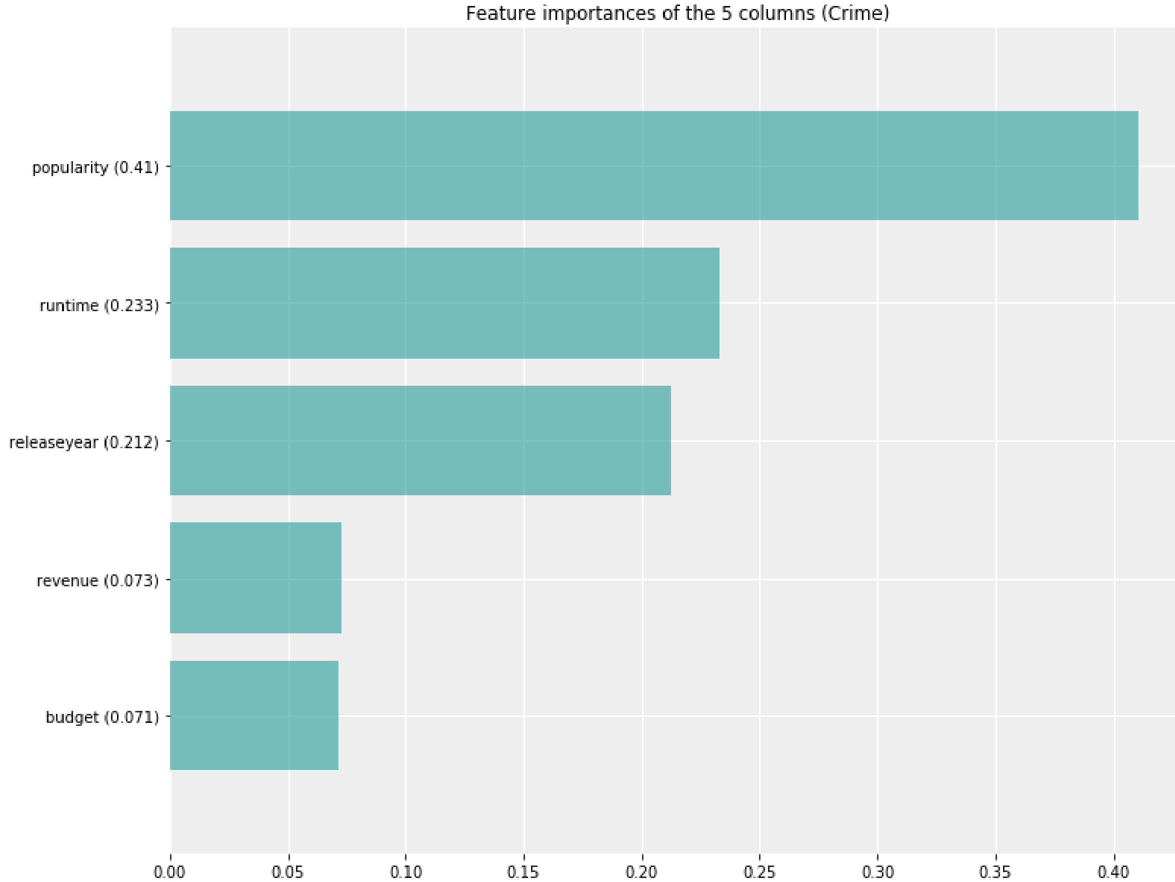
```
The validation accuracy of RF model is:  0.956848217861
```



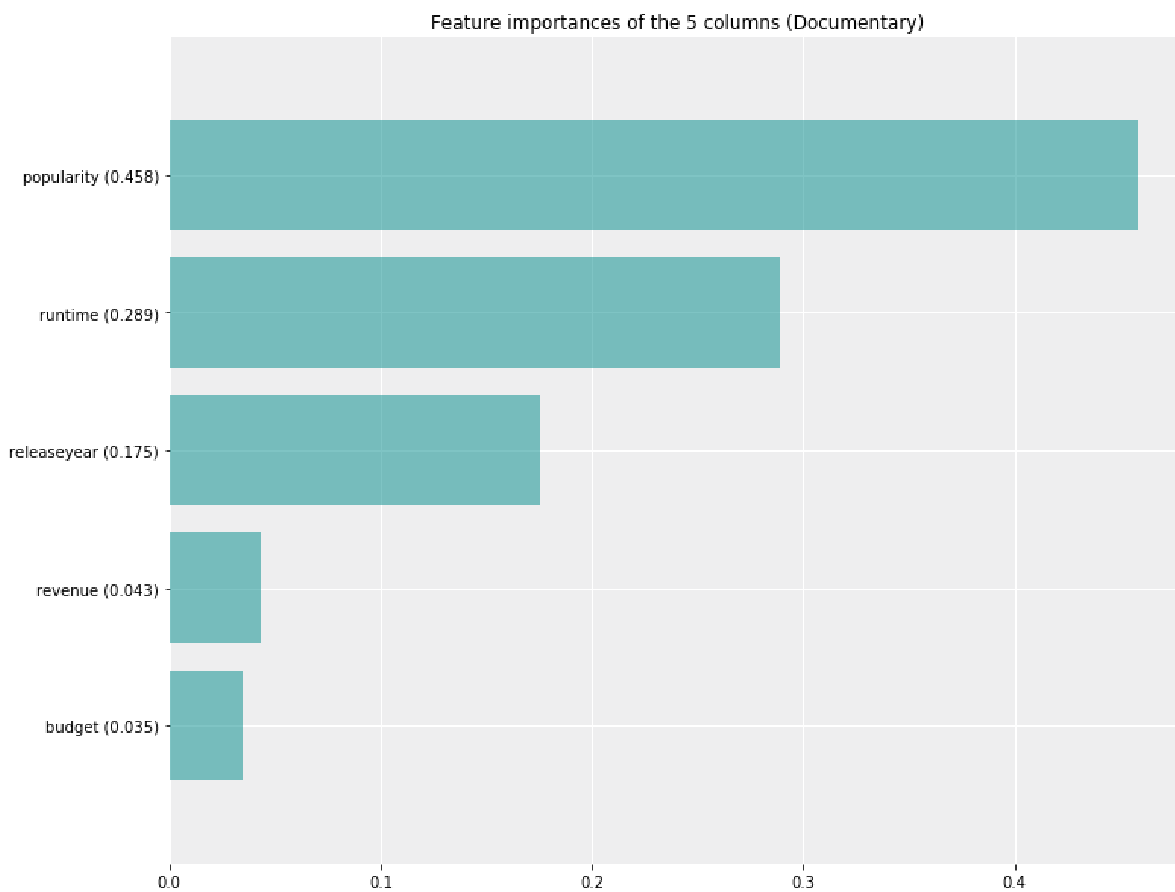

```
The validation accuracy of RF model is:  0.613636363636
```



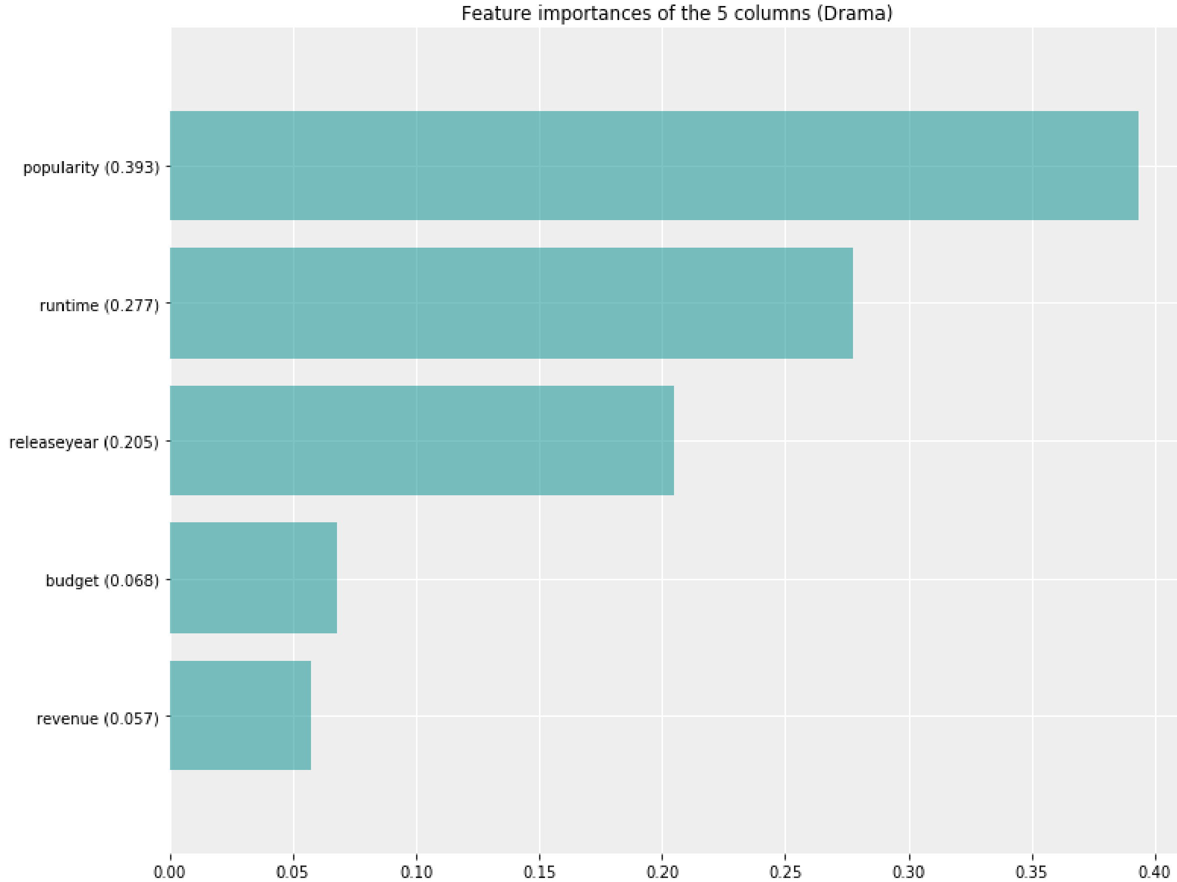
```
The validation accuracy of RF model is:  0.879054865839
```



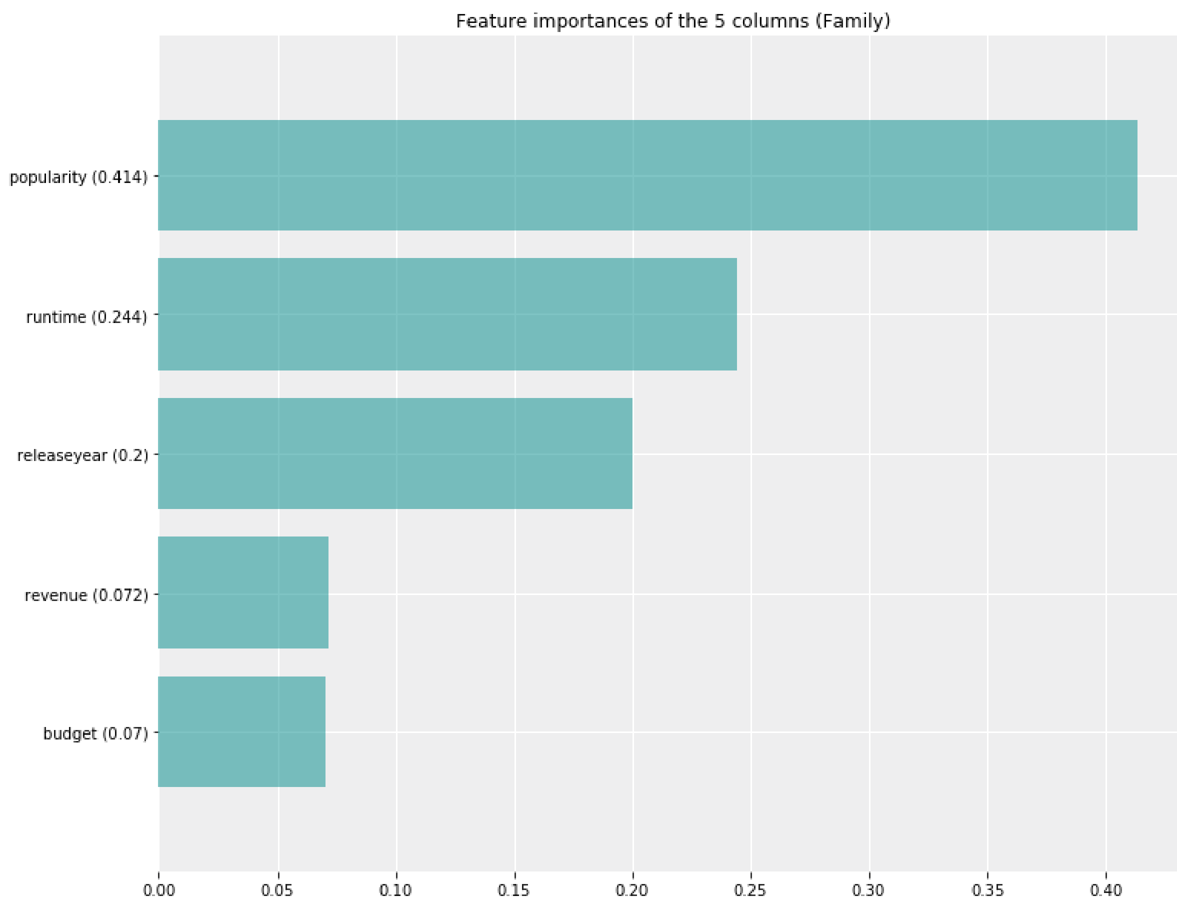

```
The validation accuracy of RF model is:  0.94433319984
```



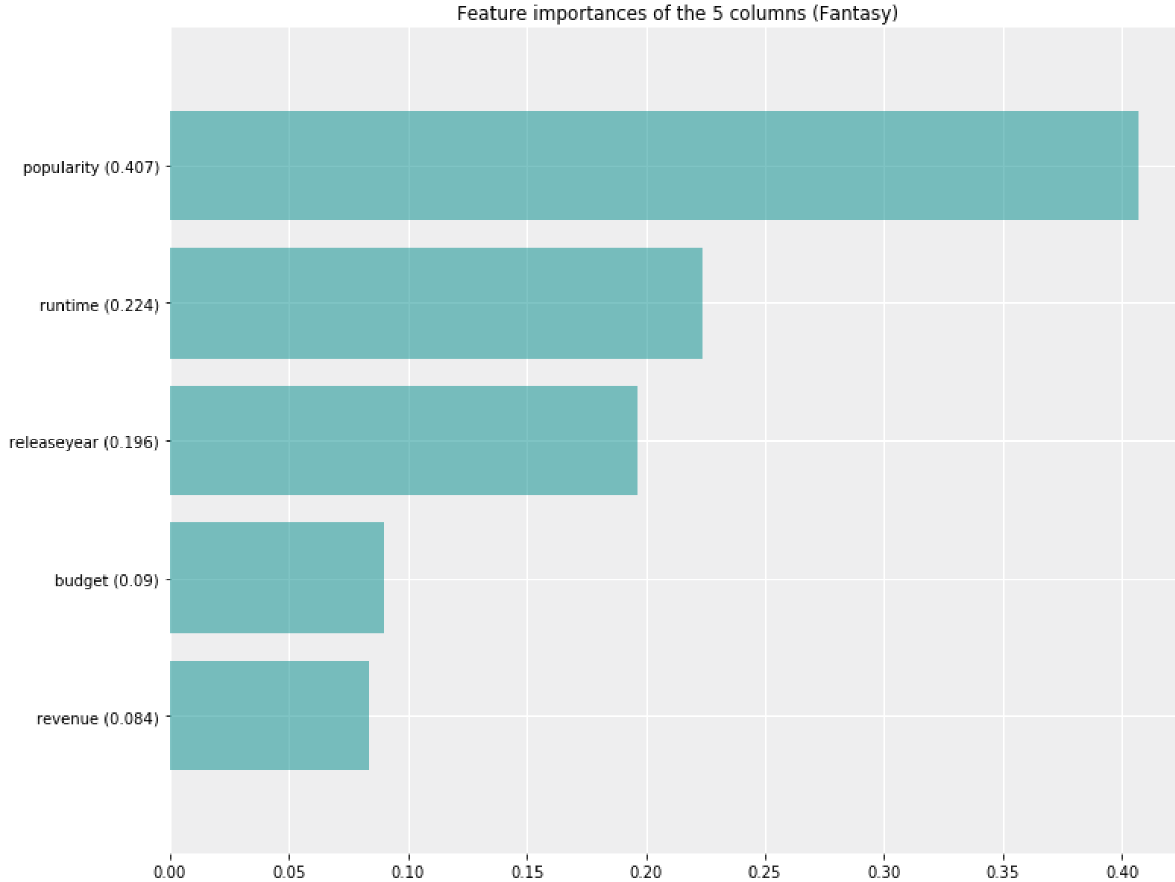
```
The validation accuracy of RF model is:  0.643271926312
```



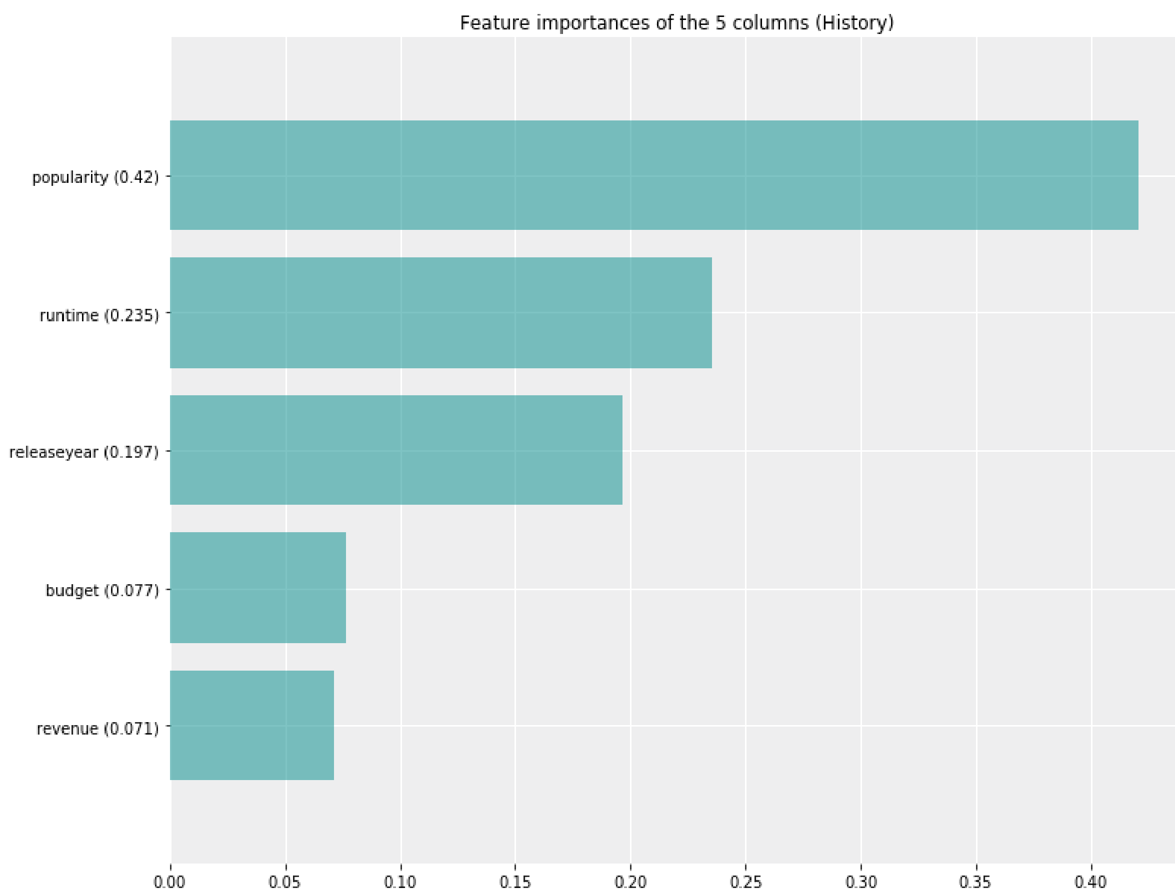

```
The validation accuracy of RF model is:  0.911593912695
```



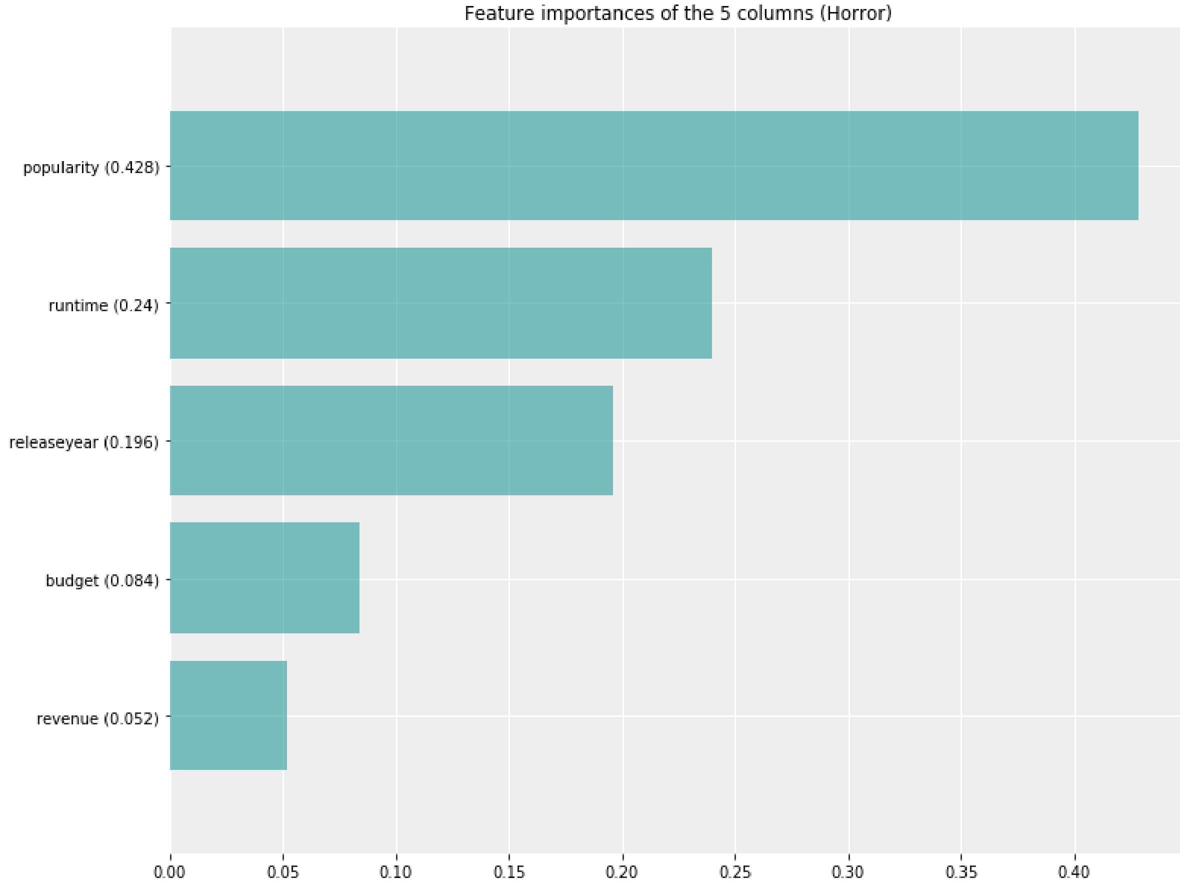
```
The validation accuracy of RF model is:  0.932719263116
```



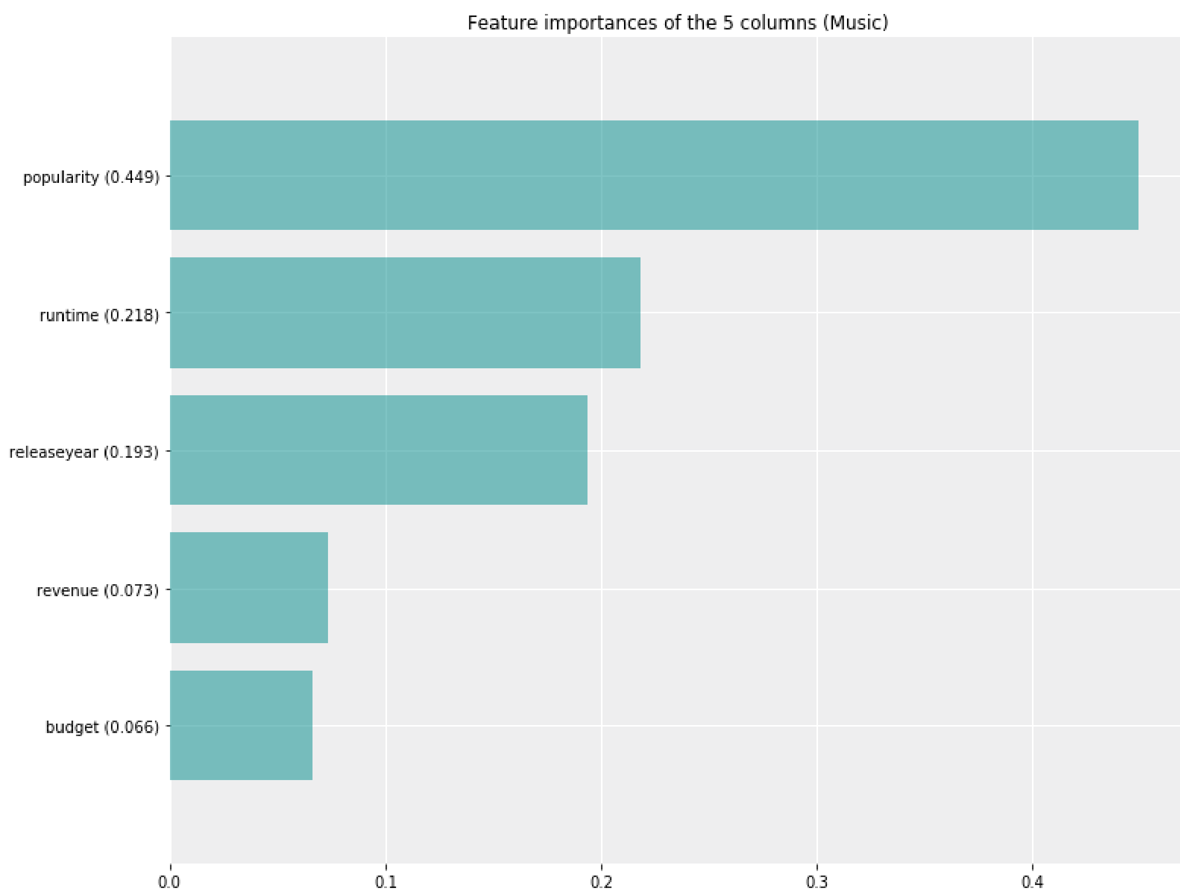

```
The validation accuracy of RF model is:  0.970164197036
```



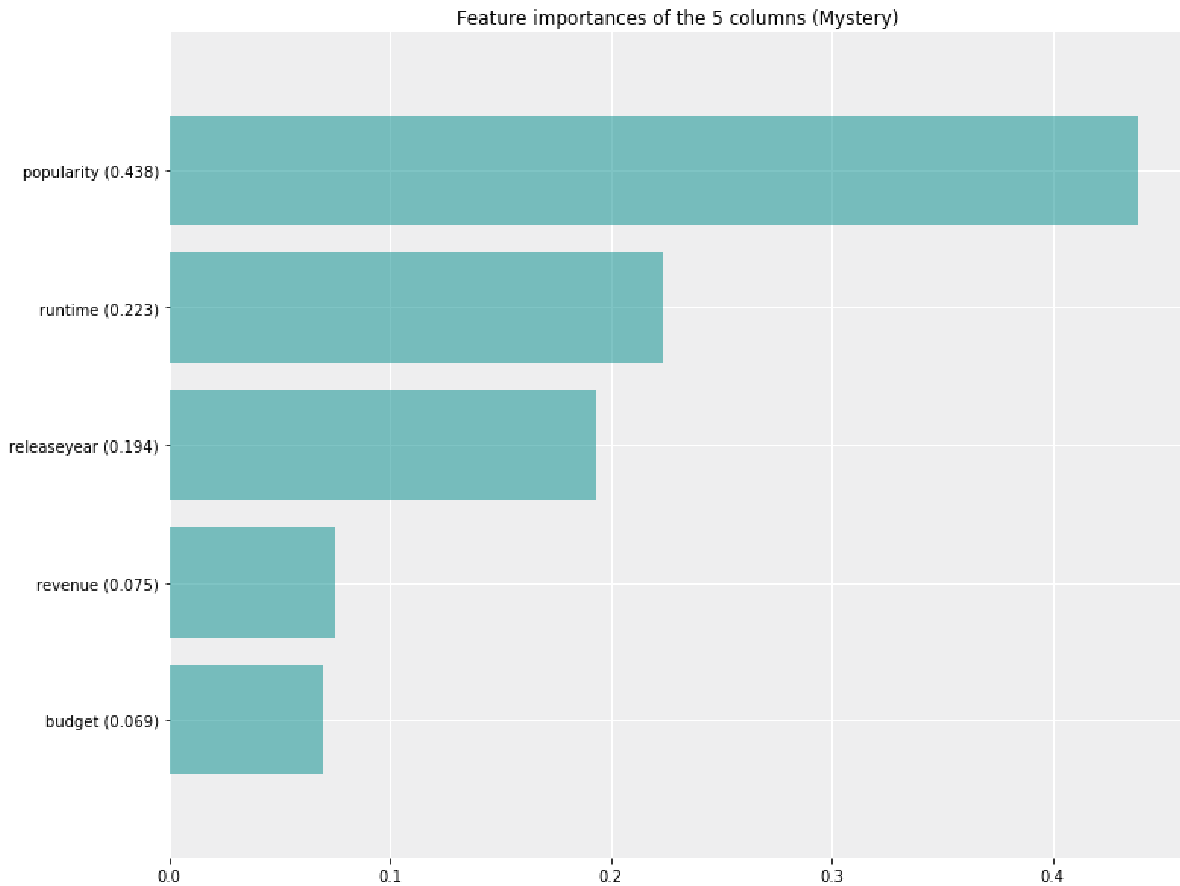
```
The validation accuracy of RF model is:  0.82749299159
```



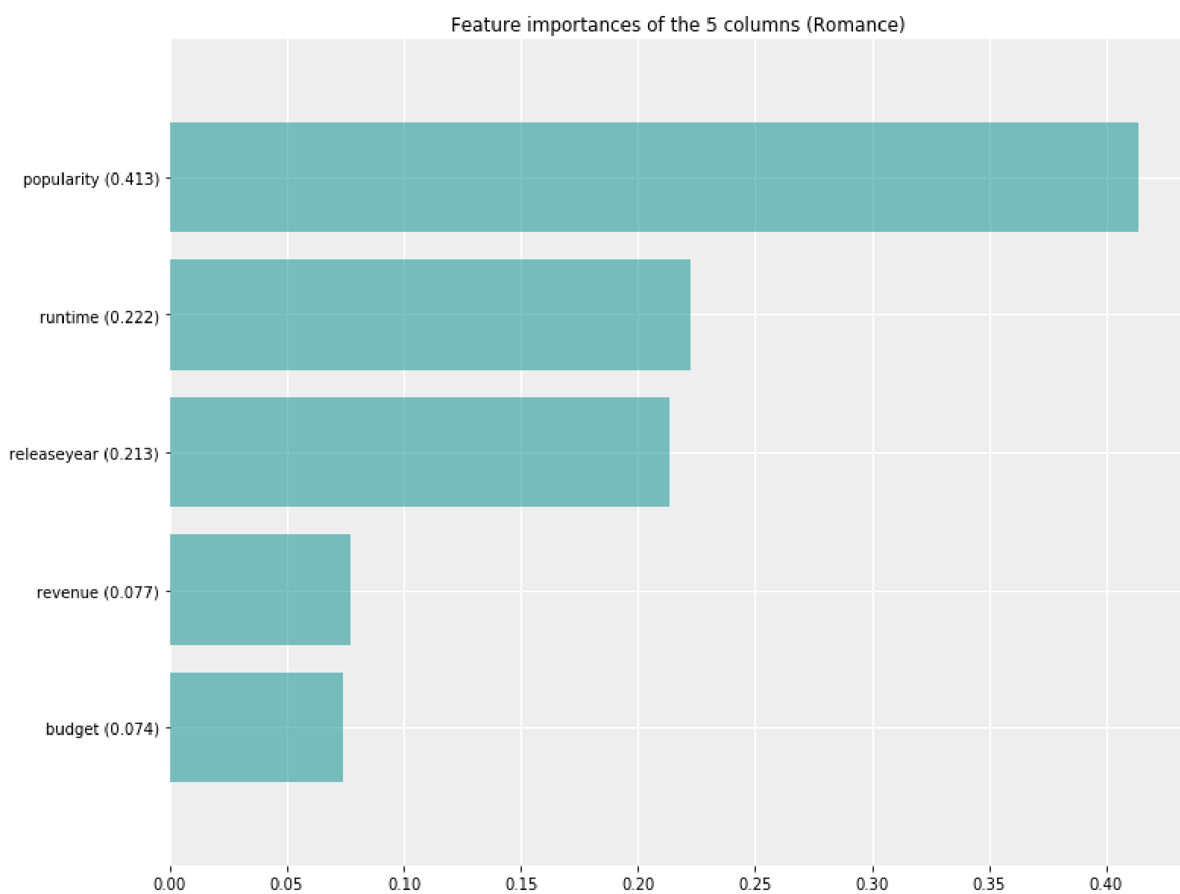

```
The validation accuracy of RF model is:  0.956948338006
```



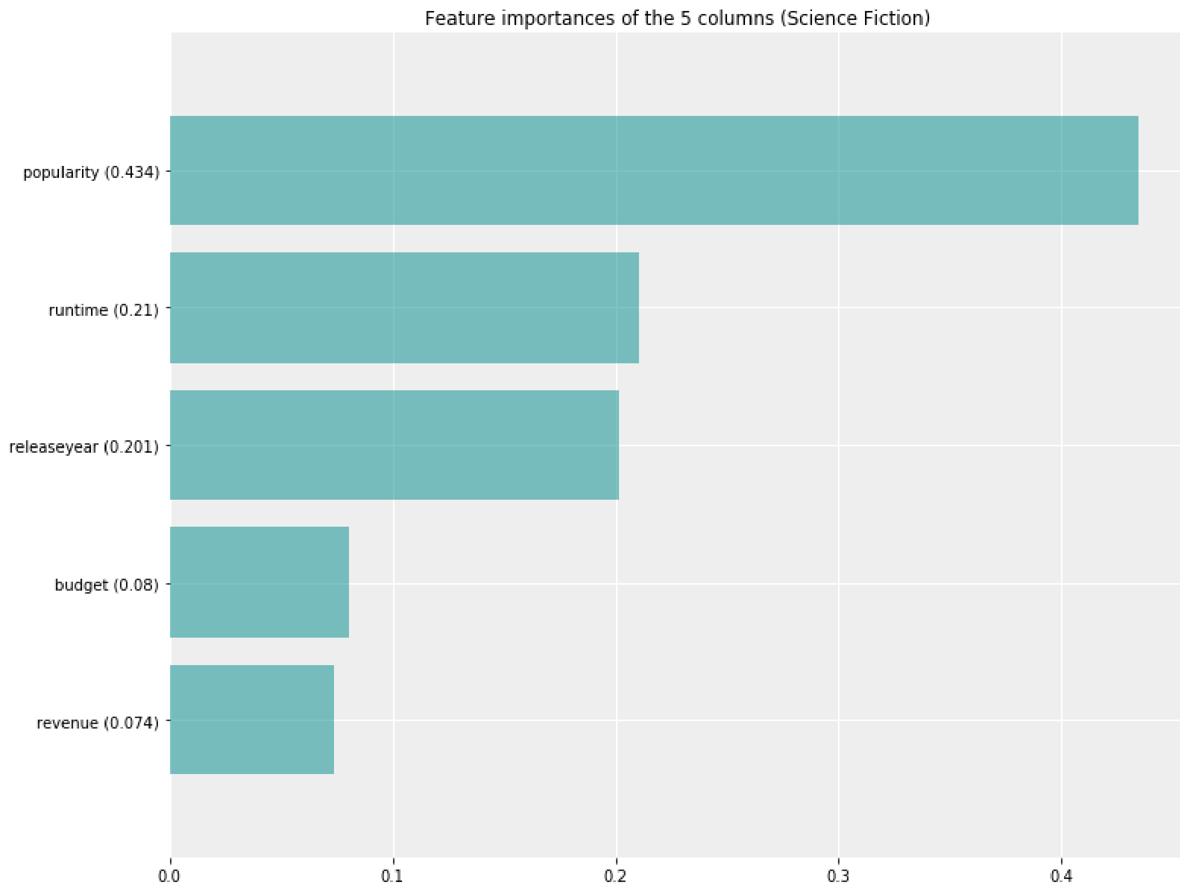
```
The validation accuracy of RF model is:  0.933520224269
```



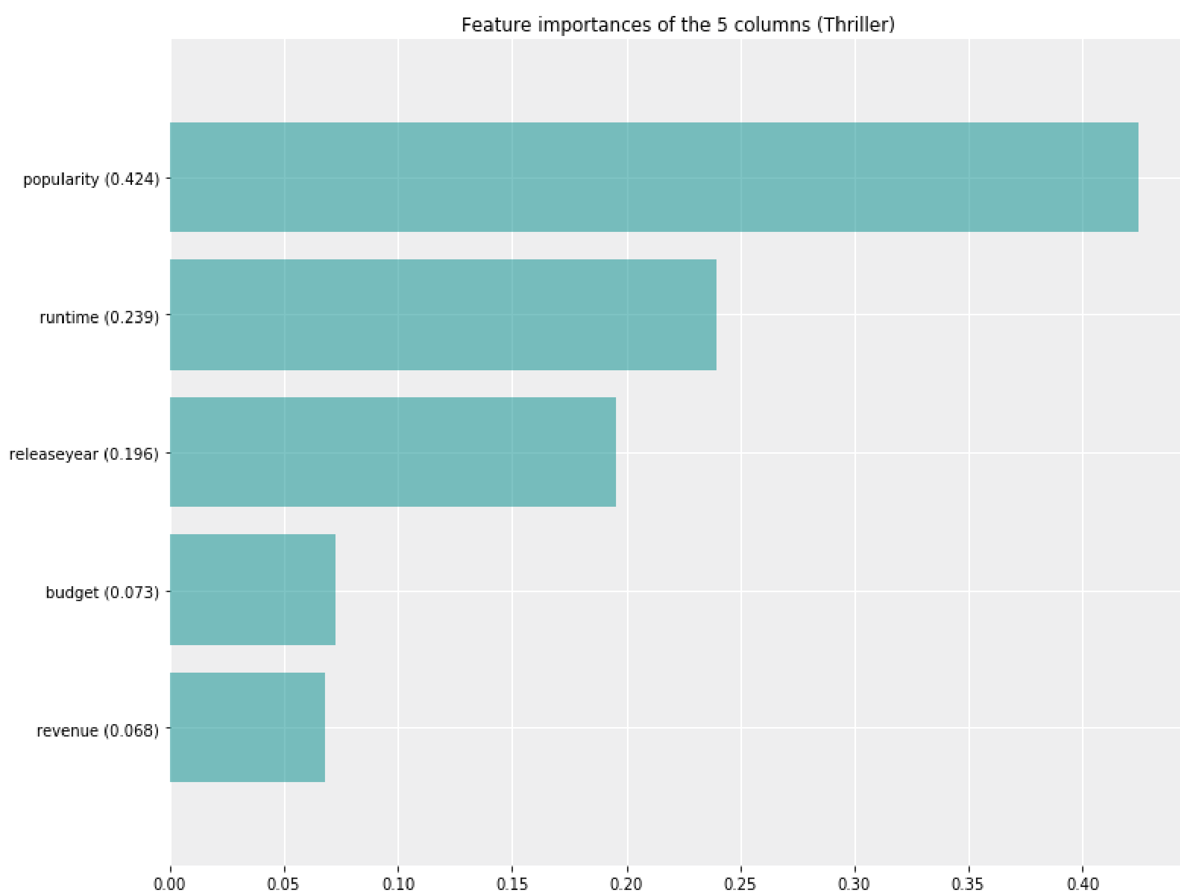

```
The validation accuracy of RF model is:  0.837905486584
```



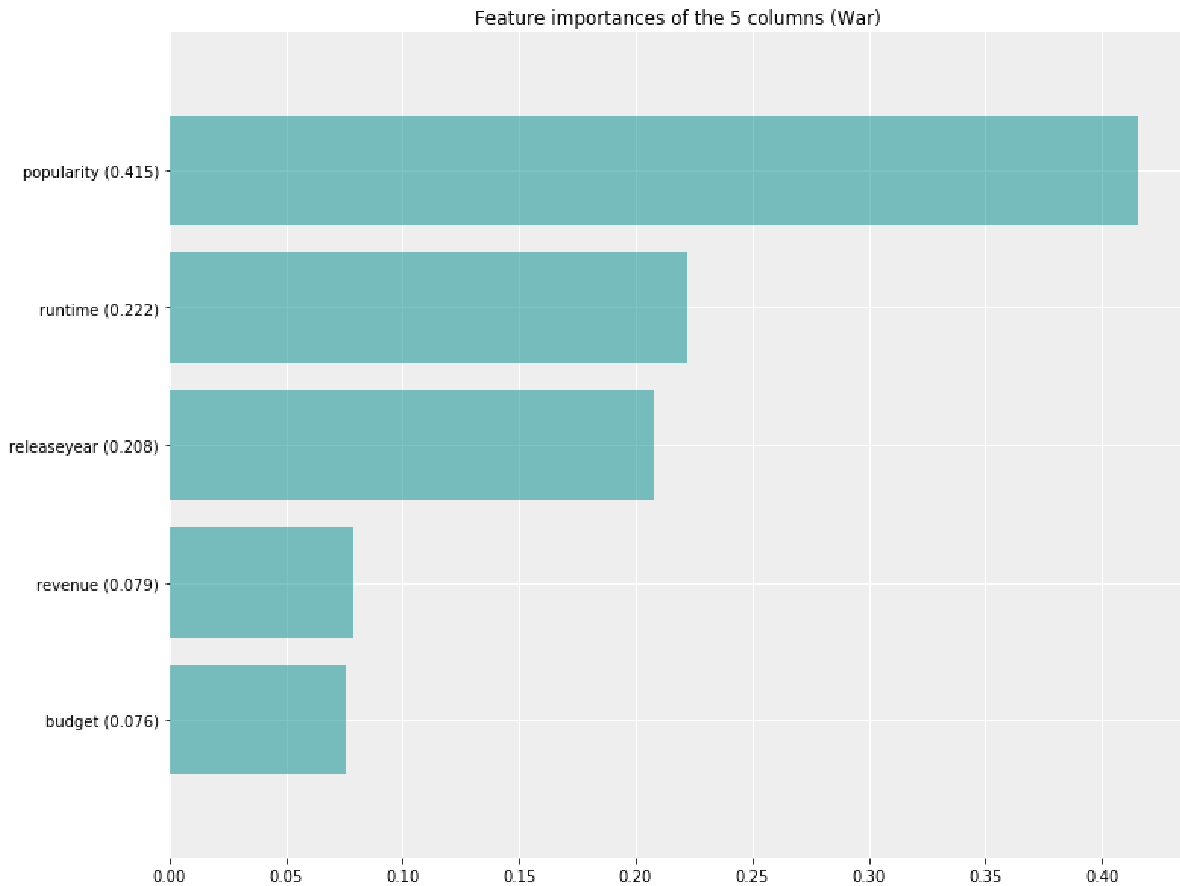
```
The validation accuracy of RF model is:  0.908990788947
```



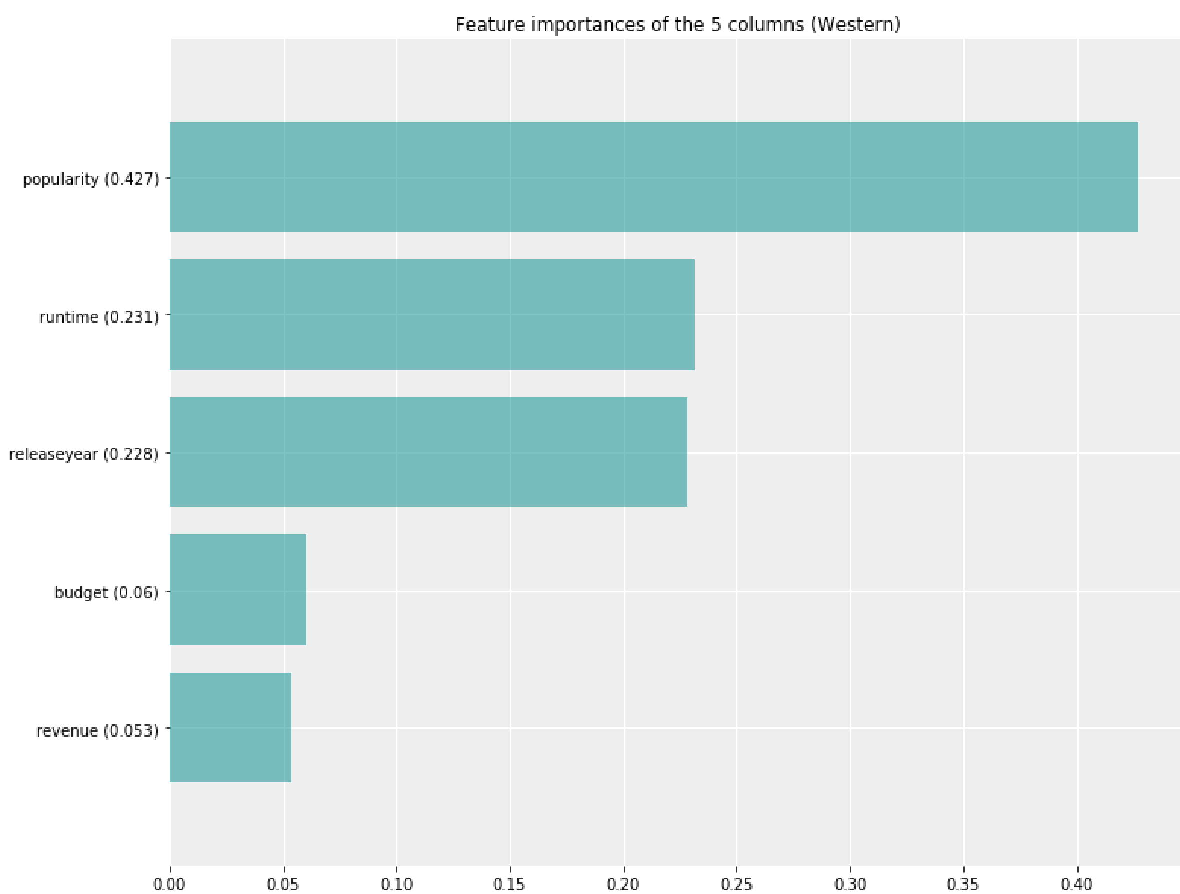

```
The validation accuracy of RF model is:  0.768922707249
```



```
The validation accuracy of RF model is:  0.976071285543
```




```
The validation accuracy of RF model is:  0.979275130156
```



From the plot above, we see, that feature importances drop to zero quite rapidly. Only the first 5 features for each genre (in this simple model, we really need to be skeptical of it) contain useful information.

Recap

After doing exploratory analysis, we did couple of observations and found some relationships, that are summarized here.

- There are not a lot of missing values.
- There are two categorical features. Since most ML algorithms can't deal with them, we performed One-Hot-Encoding to get numerical representations of them. It's worth noting, that this approach did not work on these categorical features, that have small number of unique values. Here, these columns had many unique values. We won't use them in our model.
- Some columns are highly correlated. Usually that's not a problem since in all models we would have some regularization term, that would prevent bad influence of these correlations.
- We fit Random Forest model saw, that there are not many features that this model used to predict an outcome. Also, we explored top relevant features, and detected, that IDs had large impact on prediction. That's a type of overfitting and to avoid it we would remove that columnns.

We run random forest model one more time to get feature importances.

Now, instead of KFold we use hold-out dateset to see whether model overfits.

Random Forest

In [35]:
```
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

for col in cols:

    X_train, X_test, y_train, y_test =
train_test_split(transformed_data, data[col], test_size=0.3, random_stat
e=123)



    rf_model = RandomForestClassifier(n_estimators=10, random_state=123)
    rf_model.fit(X_train, y_train)

    #Now we'll compute accuracy of prediction of test set


    y_test_predicted = rf_model.predict(X_test)

    print 'Validation accuracy for', col, accuracy_score(y_test, y_test_
predicted)

    # We see, that accuracy increased even more.
```

```
Validation accuracy for Action 0.780780780781
Validation accuracy for Adventure 0.885218551885
Validation accuracy for Animation 0.952619285953
Validation accuracy for Comedy 0.609275942609
Validation accuracy for Crime 0.877877877878
Validation accuracy for Documentary 0.941274607941
Validation accuracy for Drama 0.62028695362
Validation accuracy for Family 0.914914914915
Validation accuracy for Fantasy 0.932599265933
Validation accuracy for History 0.969302635969
Validation accuracy for Horror 0.824824824825
Validation accuracy for Music 0.955955955956
Validation accuracy for Mystery 0.932932932933
Validation accuracy for Romance 0.820820820821
Validation accuracy for Science Fiction 0.90990990991
Validation accuracy for Thriller 0.764097430764
Validation accuracy for War 0.974974974975
Validation accuracy for Western 0.976976976977
```

In [36]:
```
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import accuracy_score

def test_score(estimator):
    preds = estimator.predict(X_test)
    return accuracy_score(y_test, preds)
```

Random Forest, Decision Tree, and Gradient Boosting

Random forest has more parameters to set: number of estimators, split criteria, maximum features to account in each tree and max depth of each tree.

In [37]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix

gen_n = np.array([1,0])

def evaluate_model(classifier):
    preds = classifier.predict(X_test)
    print 'F1 score:', f1_score(y_test, preds, average='weighted')
    conf_matrix = pd.DataFrame(confusion_matrix(y_test, preds))
    conf_matrix.columns = gen_n

    conf_matrix.index = gen_n

    conf_matrix.columns.name = 'Predicted Label'
    conf_matrix.index.name = 'True Label'

    return conf_matrix
```

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
import joblib


for col in cols:

    X_train, X_test, y_train, y_test =
train_test_split(transformed_data, data[col], test_size=0.3, random_stat
e=123)
    train_size = X_train.shape[0]

    part_percentage = 0.02
    X_train_part = X_train.ix[:int(train_size * part_percentage), :]
    y_train_part = y_train.ix[:int(train_size * part_percentage)]

    params = {
        'n_estimators' : [10],
        'criterion' : ['gini', 'entropy'],
        'max_features' : ['sqrt', 'log2'],
        'max_depth' : np.arange(5, 25, 2)
    }

    grid = GridSearchCV(RandomForestClassifier(random_state=123),
params, n_jobs=-1, cv=5)
    grid.fit(X_train_part, y_train_part)

    print '------------------------------------------------------------
-----------------------------------'
    print 'Random Forest Classifier'
    print '------------------------------------------------------------
```

```
--------------------------------------'

    print col,":"
    print 'Score:', grid.best_score_
    print 'Params:', grid.best_params_
    print 'Hold-out score:', test_score(grid.best_estimator_)
    rf = grid.best_estimator_

    scores = []
    number_of_estimators = range(10, 100, 10)

    #Once we found best parameters, let's see whether accuracy improves
 when we increase number of estimators.
    for n_est in number_of_estimators:
        model = RandomForestClassifier(n_estimators=n_est, criterion='gi
ni', max_features='sqrt', \
                                       max_depth=21, random_state=123)
        model.fit(X_train_part, y_train_part)
        scores.append(test_score(model))

    print number_of_estimators
    print scores

    confusion = evaluate_model(rf)

    print confusion

    print '------------------------------------------------------------
--------------------------------------'
    print 'Decision Tree Classifier'
    print '------------------------------------------------------------
--------------------------------------'

    #Decision tree seems rational here, since when predicion loan outcom
e
    #we usually need to understand the underlying process of deciding. T
hat's where decision trees are good at
    params = {
        'criterion' : ['gini', 'entropy'],
        'max_features' : ['sqrt', 'log2', None],
        'max_depth' : np.arange(10, 20)
    }

    grid = GridSearchCV(DecisionTreeClassifier(), params, n_jobs=-1,
cv=5)
    grid.fit(X_train_part, y_train_part)
    print col,":"
    print 'Score:', grid.best_score_
    print 'Params:', grid.best_params_
    print 'Hold-out score:', test_score(grid.best_estimator_)
    dtree = grid.best_estimator_

    confusion = evaluate_model(dtree)

    print confusion

    print '------------------------------------------------------------
```

```
------------------------------------'
    print 'Gradient Boosting Classifier'
    print '----------------------------------------------------------------
------------------------------------'

    # Gradient boosting represents more general approach. It's usually m
ore powerful than AdaBoost
    params = {
        'n_estimators' : range(10, 60, 10)
    }

    grid = GridSearchCV(GradientBoostingClassifier(random_state=123), pa
rams, cv=5)
    grid.fit(X_train_part, y_train_part)
    print col,":"
    print 'Score:', grid.best_score_
    print 'Params:', grid.best_params_
    print 'Hold-out score:', test_score(grid.best_estimator_)

    gradboost = grid.best_estimator_

    confusion = evaluate_model(dtree)

    print confusion

    _ = joblib.dump(rf, 'rf_part.joblib', compress=3)
    _ = joblib.dump(dtree, 'dtree_part.joblib', compress=3)
    _ = joblib.dump(gradboost, 'gradboost_part.joblib', compress=3)
```

```
--------------------------------------------------------------------------------------------
Random Forest Classifier
--------------------------------------------------------------------------------------------
Action :
Score: 0.805625
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.807474140807
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.7811144477811145, 0.78712045378712048, 0.7911244577911245, 0.7927927
927927928, 0.79379379379379378, 0.79412746079412744, 0.7931264597931264
6, 0.79379379379379378, 0.79612946279612951]
F1 score: 0.724585560571
Predicted Label     1  0
True Label
1                2415  5
0                 572  5
--------------------------------------------------------------------------------------------
Decision Tree Classifier
--------------------------------------------------------------------------------------------
Action :
Score: 0.786041666667
Params: {'max_features': None, 'criterion': 'entropy', 'max_depth': 10}
Hold-out score: 0.778778778779
F1 score: 0.727092883088
Predicted Label     1    0
True Label
1                2292  128
0                 535   42
--------------------------------------------------------------------------------------------
Gradient Boosting Classifier
--------------------------------------------------------------------------------------------
Action :
Score: 0.806041666667
Params: {'n_estimators': 10}
Hold-out score: 0.807474140807
F1 score: 0.727092883088
Predicted Label     1    0
True Label
1                2292  128
0                 535   42
--------------------------------------------------------------------------------------------
Random Forest Classifier
--------------------------------------------------------------------------------------------
Adventure :
Score: 0.903958333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'entr
opy', 'max_depth': 5}
Hold-out score: 0.892892892893
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.8858858858858859, 0.88655321988655322, 0.88855522188855518, 0.887554
2208875542, 0.88555221888555224, 0.88688688688688688, 0.887887887887887
86, 0.88822155488822152, 0.88788788788788786]
F1 score: 0.845819499061
Predicted Label     1  0
True Label
1                2670  5
0                 316  6
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------
Adventure :
Score: 0.889791666667
Params: {'max_features': 'log2', 'criterion': 'gini', 'max_depth': 10}
Hold-out score: 0.876209542876
F1 score: 0.843250634775
Predicted Label     1   0
True Label
1                2607  68
0                 303  19
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
Adventure :
Score: 0.90375
Params: {'n_estimators': 10}
Hold-out score: 0.893226559893
F1 score: 0.843250634775
Predicted Label     1   0
True Label
1                2607  68
0                 303  19
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Animation :
Score: 0.96
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 9}
Hold-out score: 0.954954954955
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.95261928595261924, 0.95428762095428765, 0.95395395395395399, 0.95362
028695362033, 0.95261928595261924, 0.95395395395395399, 0.9539539539539
5399, 0.95362028695362033, 0.95562228895562229]
F1 score: 0.947636614313
Predicted Label     1   0
True Label
1                2796  22
0                 113  66
------------------------------------------------------------------------
```

```
----------------------------
Decision Tree Classifier
-----------------------------------------------------------------------
----------------------------
Animation :
Score: 0.951041666667
Params: {'max_features': 'sqrt', 'criterion': 'entropy', 'max_depth': 1
1}
Hold-out score: 0.95028361695
F1 score: 0.945052421965
Predicted Label     1   0
True Label
1                2776  42
0                 107  72
-----------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
-----------------------------------------------------------------------
----------------------------
Animation :
Score: 0.96
Params: {'n_estimators': 50}
Hold-out score: 0.956956956957
F1 score: 0.945052421965
Predicted Label     1   0
True Label
1                2776  42
0                 107  72
-----------------------------------------------------------------------
----------------------------
Random Forest Classifier
-----------------------------------------------------------------------
----------------------------
Comedy :
Score: 0.65
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.653319986653
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.59826493159826488, 0.61094427761094428, 0.61261261261261257, 0.61294
627961294623, 0.61828495161828501, 0.61327994661327989, 0.6119452786119
4525, 0.61461461461461464, 0.61895228561895232]
F1 score: 0.544897221145
Predicted Label     1   0
True Label
1                1893  37
0                1002  65
-----------------------------------------------------------------------
----------------------------
Decision Tree Classifier
-----------------------------------------------------------------------
----------------------------
Comedy :
Score: 0.634166666667
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.627627627628
```

```
F1 score: 0.559760510557
Predicted Label     1    0
True Label
1                1739  191
0                 925  142
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
Comedy :
Score: 0.649791666667
Params: {'n_estimators': 40}
Hold-out score: 0.652318985652
F1 score: 0.559760510557
Predicted Label     1    0
True Label
1                1739  191
0                 925  142
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Crime :
Score: 0.885416666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'entr
opy', 'max_depth': 5}
Hold-out score: 0.888555221889
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.87187187187187187, 0.8775442108775442, 0.87887887887887883, 0.879879
87987987992, 0.8808808808808809, 0.88054721388054724, 0.880880880880880
9, 0.8808808808808809, 0.88154821488154822]
F1 score: 0.836121044484
Predicted Label     1  0
True Label
1                2663  0
0                 334  0
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------

/Users/nisreenshiban/anaconda/lib/python2.7/site-packages/sklearn/metri
cs/classification.py:1113: UndefinedMetricWarning: F-score is ill-defin
ed and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
Crime :
Score: 0.87375
Params: {'max_features': 'sqrt', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.878878878879
F1 score: 0.836890289687
Predicted Label     1   0
True Label
1                2624  39
0                 324  10
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
Crime :
Score: 0.886041666667
Params: {'n_estimators': 40}
Hold-out score: 0.887887887888
F1 score: 0.836890289687
Predicted Label     1   0
True Label
1                2624  39
0                 324  10
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Documentary :
Score: 0.943958333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.945612278946
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.93660326993660326, 0.93927260593927264, 0.94160827494160826, 0.94160
827494160826, 0.94194194194194192, 0.94427761094427765, 0.9439439439439
4399, 0.94494494494494496, 0.94494494494494496]
F1 score: 0.919830468434
Predicted Label     1  0
True Label
1                2832  0
0                 163  2
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------
Documentary :
Score: 0.936666666667
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.941274607941
F1 score: 0.926356955779
Predicted Label     1   0
True Label
1                2801  31
```

```
0                  145  20
----------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
----------------------------------------------------------------------------
----------------------------
Documentary :
Score: 0.943541666667
Params: {'n_estimators': 10}
Hold-out score: 0.944944944945
F1 score: 0.926356955779
Predicted Label     1   0
True Label
1                2801  31
0                 145  20
----------------------------------------------------------------------------
----------------------------
Random Forest Classifier
----------------------------------------------------------------------------
----------------------------
Drama :
Score: 0.670833333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 7}
Hold-out score: 0.651985318652
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.61828495161828501, 0.62829496162829501, 0.63363363363363367, 0.63229
896563229893, 0.63630296963630295, 0.63763763763763759, 0.6343009676343
0099, 0.63697030363697027, 0.63697030363697027]
F1 score: 0.643991305161
Predicted Label     1    0
True Label
1                1320  379
0                 664  634
----------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
----------------------------------------------------------------------------
----------------------------
Drama :
Score: 0.643125
Params: {'max_features': 'sqrt', 'criterion': 'gini', 'max_depth': 10}
Hold-out score: 0.637971304638
F1 score: 0.635663876852
Predicted Label     1    0
True Label
1                1211  488
0                 597  701
----------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
----------------------------------------------------------------------------
----------------------------
Drama :
Score: 0.672083333333
Params: {'n_estimators': 40}
Hold-out score: 0.655655655656
```

```
F1 score: 0.635663876852
Predicted Label     1    0
True Label
1                1211  488
0                 597  701
--------------------------------------------------------------------------
----------------------------
Random Forest Classifier
--------------------------------------------------------------------------
----------------------------
Family :
Score: 0.917083333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.91958625292
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.9152485819152486, 0.9185852519185852, 0.91958625291958629, 0.9199199
1991991995, 0.92025358692025361, 0.92058725392058727, 0.921254587921254
59, 0.92092092092092093, 0.92092092092092093]
F1 score: 0.89629027249
Predicted Label     1   0
True Label
1                2710  18
0                 223  46
--------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
--------------------------------------------------------------------------
----------------------------
Family :
Score: 0.903958333333
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.908241574908
F1 score: 0.889963630834
Predicted Label     1   0
True Label
1                2671  57
0                 218  51
--------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
--------------------------------------------------------------------------
----------------------------
Family :
Score: 0.916666666667
Params: {'n_estimators': 10}
Hold-out score: 0.917917917918
F1 score: 0.889963630834
Predicted Label     1   0
True Label
1                2671  57
0                 218  51
--------------------------------------------------------------------------
----------------------------
Random Forest Classifier
--------------------------------------------------------------------------
```

```
----------------------------
Fantasy :
Score: 0.939166666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'entr
opy', 'max_depth': 5}
Hold-out score: 0.933933933934
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.93026359693026361, 0.93226559893226557, 0.93293293293293289, 0.93226
559893226557, 0.93226559893226557, 0.93226559893226557, 0.9322655989322
6557, 0.93259926593259923, 0.93226559893226557]
F1 score: 0.902029358551
Predicted Label     1  0
True Label
1                2799  0
0                 198  0
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------
Fantasy :
Score: 0.929375
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.918918918919
F1 score: 0.898904317624
Predicted Label     1   0
True Label
1                2745  54
0                 189   9
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
Fantasy :
Score: 0.939375
Params: {'n_estimators': 10}
Hold-out score: 0.933600266934
F1 score: 0.898904317624
Predicted Label     1   0
True Label
1                2745  54
0                 189   9
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
History :
Score: 0.971041666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.970637303971
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.96830163496830168, 0.97030363697030364, 0.97030363697030364, 0.97030
363697030364, 0.9706373039706373, 0.97097097097097096, 0.97030363697030
```

```
364, 0.97097097097097096, 0.97097097097097096]
F1 score: 0.956503404861
Predicted Label     1  0
True Label
1                2909  1
0                  87  0
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------
History :
Score: 0.964583333333
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.967300633967
F1 score: 0.957492490792
Predicted Label     1   0
True Label
1                2894  16
0                  82   5
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
History :
Score: 0.970625
Params: {'n_estimators': 10}
Hold-out score: 0.97030363697
F1 score: 0.957492490792
Predicted Label     1   0
True Label
1                2894  16
0                  82   5
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Horror :
Score: 0.850416666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 7}
Hold-out score: 0.844511177845
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.82749416082749416, 0.82849516182849514, 0.82682682682682684, 0.83083
083083083087, 0.83083083083083087, 0.83049716383049721, 0.8308308308308
3087, 0.82949616282949612, 0.8288288288288288]
F1 score: 0.783710516831
Predicted Label     1   0
True Label
1                2512  16
0                 450  19
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
```

```
--------------------------------------------------------------------------------
------------------------------
Horror :
Score: 0.840625
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.823156489823
F1 score: 0.783036937638
Predicted Label     1    0
True Label
1                2422  106
0                 424   45
--------------------------------------------------------------------------------
------------------------------
Gradient Boosting Classifier
--------------------------------------------------------------------------------
------------------------------
Horror :
Score: 0.850833333333
Params: {'n_estimators': 30}
Hold-out score: 0.842509175843
F1 score: 0.783036937638
Predicted Label     1    0
True Label
1                2422  106
0                 424   45
--------------------------------------------------------------------------------
------------------------------
Random Forest Classifier
--------------------------------------------------------------------------------
------------------------------
Music :
Score: 0.960208333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 13}
Hold-out score: 0.956956956957
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.95595595595595595, 0.95628962295628961, 0.95729062395729059, 0.95695
695695695693, 0.95695695695695693, 0.95695695695695693, 0.9569569569569
5693, 0.95695695695695693, 0.95695695695695693]
F1 score: 0.936235126857
Predicted Label     1  0
True Label
1                2868  1
0                 128  0
--------------------------------------------------------------------------------
------------------------------
Decision Tree Classifier
--------------------------------------------------------------------------------
------------------------------
Music :
Score: 0.948333333333
Params: {'max_features': 'sqrt', 'criterion': 'gini', 'max_depth': 10}
Hold-out score: 0.947614280948
F1 score: 0.933087644823
Predicted Label     1   0
True Label
```

```
1                2837  32
0                 125   3
-----------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
-----------------------------------------------------------------------
----------------------------
Music :
Score: 0.959375
Params: {'n_estimators': 10}
Hold-out score: 0.957290623957
F1 score: 0.933087644823
Predicted Label     1   0
True Label
1                2837  32
0                 125   3
-----------------------------------------------------------------------
----------------------------
Random Forest Classifier
-----------------------------------------------------------------------
----------------------------
Mystery :
Score: 0.938541666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'entr
opy', 'max_depth': 5}
Hold-out score: 0.936269602936
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.93426760093426764, 0.93593593593593594, 0.9346012679346013, 0.935268
60193526862, 0.9346012679346013, 0.93493493493493496, 0.934601267934601
3, 0.9346012679346013, 0.93493493493493496]
F1 score: 0.90609858523
Predicted Label     1  0
True Label
1                2806  2
0                 189  0
-----------------------------------------------------------------------
----------------------------
Decision Tree Classifier
-----------------------------------------------------------------------
----------------------------
Mystery :
Score: 0.92875
Params: {'max_features': 'sqrt', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.931931931932
F1 score: 0.906815446563
Predicted Label     1   0
True Label
1                2788  20
0                 184   5
-----------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
-----------------------------------------------------------------------
----------------------------
Mystery :
Score: 0.9375
```

```
Params: {'n_estimators': 10}
Hold-out score: 0.936603269937
F1 score: 0.906815446563
Predicted Label     1   0
True Label
1                2788  20
0                 184   5
----------------------------------------------------------------------
----------------------------
Random Forest Classifier
----------------------------------------------------------------------
----------------------------
Romance :
Score: 0.855208333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.840840840841
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.82582582582582587, 0.82916249582916246, 0.82782782782782782, 0.82982
982982982978, 0.83049716383049721, 0.83249916583249917, 0.8314981648314
9819, 0.83083083083083087, 0.83083083083083087]
F1 score: 0.769412535783
Predicted Label     1  0
True Label
1                2518  2
0                 475  2
----------------------------------------------------------------------
----------------------------
Decision Tree Classifier
----------------------------------------------------------------------
----------------------------
Romance :
Score: 0.838541666667
Params: {'max_features': 'sqrt', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.832499165832
F1 score: 0.775678937167
Predicted Label     1   0
True Label
1                2474  46
0                 456  21
----------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
----------------------------------------------------------------------
----------------------------
Romance :
Score: 0.855416666667
Params: {'n_estimators': 10}
Hold-out score: 0.840840840841
F1 score: 0.775678937167
Predicted Label     1   0
True Label
1                2474  46
0                 456  21
----------------------------------------------------------------------
----------------------------
```

```
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Science Fiction :
Score: 0.914791666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.912579245913
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.90590590590590592, 0.90990990990990994, 0.91124457791124458, 0.91124
457791124458, 0.91157824491157824, 0.91157824491157824, 0.9119119119119
119, 0.91124457791124458, 0.91157824491157824]
F1 score: 0.870866796082
Predicted Label     1  0
True Label
1                2735  0
0                 262  0
------------------------------------------------------------------------
----------------------------
Decision Tree Classifier
------------------------------------------------------------------------
----------------------------
Science Fiction :
Score: 0.901041666667
Params: {'max_features': 'sqrt', 'criterion': 'gini', 'max_depth': 10}
Hold-out score: 0.904571237905
F1 score: 0.870821567382
Predicted Label     1   0
True Label
1                2704  31
0                 255   7
------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
------------------------------------------------------------------------
----------------------------
Science Fiction :
Score: 0.914375
Params: {'n_estimators': 10}
Hold-out score: 0.912579245913
F1 score: 0.870821567382
Predicted Label     1   0
True Label
1                2704  31
0                 255   7
------------------------------------------------------------------------
----------------------------
Random Forest Classifier
------------------------------------------------------------------------
----------------------------
Thriller :
Score: 0.791041666667
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.798798798799
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.76042709376042705, 0.76643309976643315, 0.77277277277277279, 0.77544
```

```
210877544206, 0.773773773773773 77, 0.77544210877544206, 0.7744411077744
4108, 0.77310643977310645, 0.77610944277610949]
F1 score: 0.709450686078
Predicted Label     1  0
True Label
1                2394  0
0                 603  0
-----------------------------------------------------------------------
----------------------------
Decision Tree Classifier
-----------------------------------------------------------------------
----------------------------
Thriller :
Score: 0.768958333333
Params: {'max_features': 'log2', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.775108441775
F1 score: 0.719557797714
Predicted Label     1    0
True Label
1                2278  116
0                 558   45
-----------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
-----------------------------------------------------------------------
----------------------------
Thriller :
Score: 0.789791666667
Params: {'n_estimators': 10}
Hold-out score: 0.798798798799
F1 score: 0.719557797714
Predicted Label     1    0
True Label
1                2278  116
0                 558   45
-----------------------------------------------------------------------
----------------------------
Random Forest Classifier
-----------------------------------------------------------------------
----------------------------
War :
Score: 0.976458333333
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'entr
opy', 'max_depth': 9}
Hold-out score: 0.975975975976
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.97597597597597596, 0.9756423089756423, 0.9756423089756423, 0.9759759
7597597596, 0.97630964297630962, 0.97630964297630962, 0.976643309976643
28, 0.97597597597597596, 0.97597597597597596]
F1 score: 0.964439616922
Predicted Label     1  0
True Label
1                2925  1
0                  71  0
-----------------------------------------------------------------------
----------------------------
```

```
Decision Tree Classifier
--------------------------------------------------------------------------------------------
War :
Score: 0.96875
Params: {'max_features': 'sqrt', 'criterion': 'entropy', 'max_depth': 1
0}
Hold-out score: 0.96963630297
F1 score: 0.963152907965
Predicted Label     1   0
True Label
1                2902  24
0                  67   4
--------------------------------------------------------------------------------------------
Gradient Boosting Classifier
--------------------------------------------------------------------------------------------
War :
Score: 0.975416666667
Params: {'n_estimators': 10}
Hold-out score: 0.975975975976
F1 score: 0.963152907965
Predicted Label     1   0
True Label
1                2902  24
0                  67   4
--------------------------------------------------------------------------------------------
Random Forest Classifier
--------------------------------------------------------------------------------------------
Western :
Score: 0.978125
Params: {'max_features': 'sqrt', 'n_estimators': 10, 'criterion': 'gin
i', 'max_depth': 5}
Hold-out score: 0.978978978979
[10, 20, 30, 40, 50, 60, 70, 80, 90]
[0.97831164497831169, 0.97797797797797803, 0.97764431097764426, 0.97831
164497831169, 0.97831164497831169, 0.97864531197864535, 0.9786453119786
4535, 0.97864531197864535, 0.97897897897897901]
F1 score: 0.968580112738
Predicted Label     1  0
True Label
1                2934  0
0                  63  0
--------------------------------------------------------------------------------------------
Decision Tree Classifier
--------------------------------------------------------------------------------------------
Western :
Score: 0.968958333333
Params: {'max_features': 'log2', 'criterion': 'gini', 'max_depth': 10}
Hold-out score: 0.967634300968
F1 score: 0.965157117708
Predicted Label     1   0
```

```
True Label
1                2894  40
0                  57   6
-------------------------------------------------------------------------
----------------------------
Gradient Boosting Classifier
-------------------------------------------------------------------------
----------------------------
Western :
Score: 0.977083333333
Params: {'n_estimators': 10}
Hold-out score: 0.977644310978
F1 score: 0.965157117708
Predicted Label     1   0
True Label
1                2894  40
0                  57   6
```

The above plots look like a straight line, so we can choose number of estimators equal to 10.

We'll save all the models so that we don't need to refit them.

This part loads models from file. We uncomment if we already trained models.

In [ ]:

```
# import joblib
# rf = joblib.load('rf_part.joblib')
# dtree = joblib.load('dtree_part.joblib')
# gradboost = joblib.load('gradboost_part.joblib')
```

Model Comparison and Evaluation

Now, we have all this models trained. There are number of parameters that influence model:

- Speed (training and prediction time)
- Memory consumption
- Metrics score
- Interpretability

1. **Speed**

   Since this report was created, all proposed models can be built in a reasonable time on a single machine. As for testing time, all provided models are much quicker to test than train, so it's also affordable.
2. **Memory**

   Number of parameters of each model is very small compared to size of dataset, so after training it consumes very little memory.
3. **Metrics score**

   That is the main characteristic of the model. All others are just binary criterias - we won't take a model if it doesn't fit any of them. Among all models that satisfy all other criterias, we pick the model with highest metrics score.

   For each real problem we create own metrics to optimize. We propose two choices of metrics:

   - Accuracy. That means we conider all examples are weighted equally. That is usually not true, and it's not true in our problem, so that is not the best metric to choose.
   - F1-score. This metric account for class imbalance. As seen before, some classes have small number of examples. That gives the efffect that each class is weighted inverse proportional to class size. This metric is also not designed to this task. Though it would give more reasonable results, since we'll penalize for not paying attention to some class,
4. **Interpretability**

   In genre prediction task, it would be great if human can evaluate the model and see how it makes its decisions, so that he could find maybe unreasonable decicsions/logic in algorithm's thinking. Not all the models provide a nice interpretation.
5. **Variance**

   How much different can results be when we train an algorithm mulitple times. That gives us the confidence about the prediction. We prefer variance to be small, but there's a bias-variance tradeoff.

As final metrics we'll take F1 score. We'll also plot confusion matrix to understand where algorithm makes mistakes.

Let's now evaluate each algorithm and choose the best one to our task.

Random Forest

- Memory: memory consumption is pretty small.
- Interpretability: since RF is an ensemble method, it combines a lot of decision trees together and that's why it's not well interpreted.
- Variance: Since we average a lot of predictions, variance for this method is low.
- Metrics score: as we see, random forest produces very high F1-score.

Decision Tree

- Memory: memory consumption is pretty small.
- Interpretability: decision tree is great at understanding the underlying progress. We can see what decisions were made at each step of the algorithm.
- Variance: The exact tree shape highly depends on the training data. Even small changes can result into big changes in final model. This model has high variance.
- Metrics score: as we see, decision trees give us pretty high score. That means we can predict each person's loan status by "asking" series of questions about values of features.

Gradient Boosting

GB is another to combining classifiers. It also lacks interpretability.

But as we see, it has high F1 score, comparable to random forest model with makes it also possible for a best model.

Results

The final choice of the model is not obvious. As stated above, it relies on the specific goal we want to accomplish.

For sure we need high accuracy and F1-score, and we used the following models:

- Random Forest
- Decision Tree
- Gradient Boosting

Scores of these models are comparable one to another. But decision tree has the property that it's interpretable, that can be critical in this problem. Also, the training time for this model is lowest.