Project Overview

Create a model that predicts whether a KIVA loan request is from a male vs female.

This is the first of several KIVA analyses and models. Further models (separate documents) will predict which loans are most likely to be fulfilled based on country and/or activity.

Summary

Loans to made in the countries of: India, Liberia, Pakistan, Philipines, and Sierra Leon, for a loan Activity of: Fruits/Vegetables, Fish Selling, Food Market, and General Store are most likely being made to a woman.

Data Cleansing

- Use MinMaxScaler to standardize values in columns with siginificant variance
- Create dummy variables
- Drop small values for Country, Activity, and Number of Borrowers
- Drop null values/strip white spaces and unnecessary characters
- Change column names for easier readability
- Change Gender to binary

Analysis and Results

- Split data into train and test (80%, 20%)
- Target variable > ‘Gender’
- Ran following models:

1. Extra Trees Classifier to determine feature ranking
   - Highest ranked: Armenia, Burkina Faso, Cambodia, Cameroon
   - These results do not match what is seen in logistic regression model
   - Based on crosstab data, it does not appear that Extra Trees Classifier provided correctly ranked Countries or Activities according to females (to be further investigated)

2. Random Forest Classifer
   - Train score = .91
   - Test score = .73 (low score unexpected)

3. Logistic Regression CV
   - Training data score = .75
   - Test data score = .73

4. Logistic Regression
   - Test data Accuracy Score = .71
   - Test data AUC = .80

In [1]:

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

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	Id	Name	Country	Activity	Number of Borrowers	Paid/Raised	Loan Amount	Gender
0	1233677	Robert	Armenia	Agriculture	1	1,500	1,500	Male
1	1232922	Aleksan	Armenia	Agriculture	1	1,500	1,500	Male
2	1242118	Elina	Armenia	Agriculture	1	3,000	3,000	Female
3	1236073	Aquilino	Bolivia	Agriculture	1	300	300	Male
4	1241034	Teeltaaba De Zongo Group	Burkina Faso	Agriculture	5	825	825	Female

In [4]:

```
df.shape
```

Out[4]:

```
(3000, 8)
```

In [5]:

```
df.dtypes
```

Out[5]:

```
Id                      int64
Name                   object
Country                object
Activity               object
Number of Borrowers     int64
Paid/Raised            object
Loan Amount            object
Gender                 object
dtype: object
```

In [6]:

```
df.columns=['Id', 'Name', 'Country', 'Activity', 'NumBorrowers',
            'Raised', 'LoanAmt', 'Gender']
```

In [7]:

```
df.head(2)
```

Out[7]:

	Id	Name	Country	Activity	NumBorrowers	Raised	LoanAmt	Gender
0	1233677	Robert	Armenia	Agriculture	1	1,500	1,500	Male
1	1232922	Aleksan	Armenia	Agriculture	1	1,500	1,500	Male

In [8]:

```
df['Raised'] = df['Raised'].str.replace(',','')
```

In [9]:

```
df['Raised'].value_counts()
```

Out[9]:

```
200     279
100     231
300     209
```

175	174
500	140
125	132
400	120
150	105
600	97
225	89
1000	81
250	75
325	70
425	55
450	52
1500	52
350	50
275	48
625	44
550	41
375	40
75	36
700	35
800	33
475	31
575	30
50	30
925	29
750	28
525	27
...	...
3500	1
4175	1
2100	1
4625	1
2675	1
4750	1
2600	1
3750	1
2450	1
5450	1
1900	1
4775	1
2175	1
1850	1
4125	1
2625	1
3575	1
2425	1
5250	1
8575	1
4550	1
4200	1
2825	1
3075	1
3525	1

```
2750      1
1325      1
2900      1
3825      1
4300      1
Name: Raised, dtype: int64
```

In [10]:

```
df['LoanAmt'] = df['LoanAmt'].str.replace(',','')
```

In [11]:

```
df['LoanAmt'].value_counts()
```

Out[11]:

```
200     279
100     231
300     209
175     174
500     140
125     132
400     120
150     105
600      97
225      89
1000     81
250      75
325      70
425      55
450      52
1500     52
350      50
275      48
625      44
550      41
375      40
75       36
700      35
800      33
475      31
575      30
50       30
925      29
750      28
525      27
        ...
3500      1
4175      1
2100      1
4625      1
2675      1
4750      1
```

```
2600      1
3750      1
2450      1
5450      1
1900      1
4775      1
2175      1
1850      1
4125      1
2625      1
3575      1
2425      1
5250      1
8575      1
4550      1
4200      1
2825      1
3075      1
3525      1
2750      1
1325      1
2900      1
3825      1
4300      1
Name: LoanAmt, dtype: int64
```

In [12]:

```
df['Country'] = df['Country'].str.replace(' ','')
df['Country'] = df['Country'].str.strip(' ')
```

In [13]:

```
df['Country'] = df['Country'].str.strip(',')
df['Country'] = df['Country'].str.strip("'")
```

In [14]:

```
df['Activity'] = df['Activity'].str.replace(' ','')
df['Activity'] = df['Activity'].str.strip(' ')
```

In [15]:

```
df.dtypes
```

Out[15]:

```
Id               int64
Name            object
Country         object
Activity        object
NumBorrowers     int64
Raised          object
LoanAmt         object
Gender          object
dtype: object
```

In [16]:

```
df['Gender'] = df['Gender'].map({'Male':0, 'Female':1})
```

In [17]:

```
df.head()
```

Out[17]:

	Id	Name	Country	Activity	NumBorrowers	Raised	LoanAmt	Gender
0	1233677	Robert	Armenia	Agriculture	1	1500	1500	0
1	1232922	Aleksan	Armenia	Agriculture	1	1500	1500	0
2	1242118	Elina	Armenia	Agriculture	1	3000	3000	1
3	1236073	Aquilino	Bolivia	Agriculture	1	300	300	0
4	1241034	Teeltaaba De Zongo Group	BurkinaFaso	Agriculture	5	825	825	1

In [18]:

```
df['Raised'] = df['Raised'].astype(int)
df['LoanAmt'] = df['LoanAmt'].astype(int)
```

In [19]:

```
df.dtypes
```

Out[19]:

```
Id               int64
Name            object
Country         object
Activity        object
NumBorrowers     int64
Raised           int64
LoanAmt          int64
Gender           int64
dtype: object
```

In [20]:

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

Out[20]:

```
Id              0
Name            0
Country         0
Activity        0
NumBorrowers    0
Raised          0
LoanAmt         0
Gender          0
dtype: int64
```

In [21]:

```
#crossCA = pd.crosstab[df.Country, df.Activity, margins=True]
countryact = pd.crosstab(index=df['Country'],
                         columns=df['Activity'])

countryact
```

Out[21]:

Activity	Agriculture	AnimalSales	Arts	AutoRepair	Bakery	BalutMaking	BarberShop	Bea
Country								
Albania	0	0	0	0	0	0	0	0
Armenia	3	0	0	0	0	0	0	0
Bolivia	1	0	0	0	0	0	0	0
Brazil	0	0	0	0	0	0	0	0
BurkinaFaso	1	0	0	0	0	0	0	1
Burundi	0	0	0	0	0	0	0	0
Cambodia	1	1	0	0	0	0	0	0

In [22]:

```
countrygen = pd.crosstab(index=df['Country'],
                         columns=df['Gender'])

countrygen
```

Out[22]:

Gender	0	1
Country		
Albania	4	0
Armenia	15	4
Bolivia	8	0
Brazil	4	0
BurkinaFaso	18	7
Burundi	4	0
Cambodia	164	137
Cameroon	20	9

In [23]:

```
activitygen = pd.crosstab(index=df['Activity'],
                          columns=df['Gender'])

activitygen
```

Out[23]:

Gender	0	1
Activity		
Agriculture	97	25
AnimalSales	15	12
Arts	0	1
AutoRepair	11	2
Bakery	11	8
BalutMaking	0	1
BarberShop	8	0
BeautySalon	5	10

In [24]:

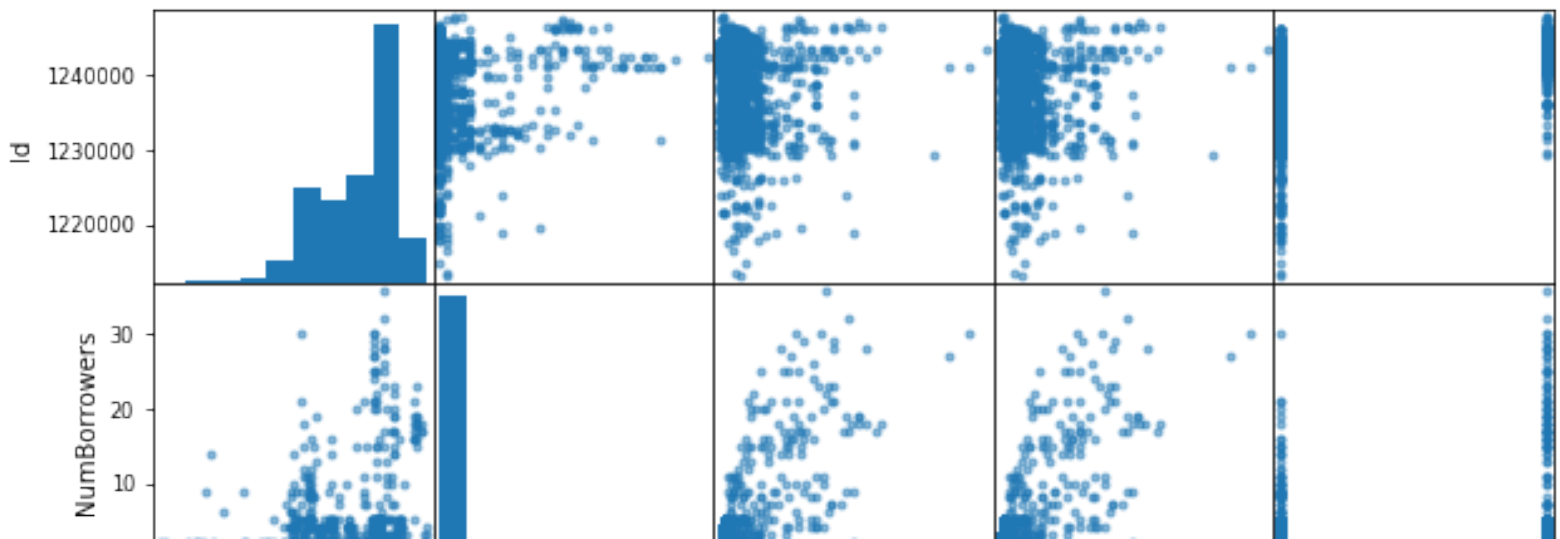
```
import matplotlib.pyplot as plt
%matplotlib inline
```

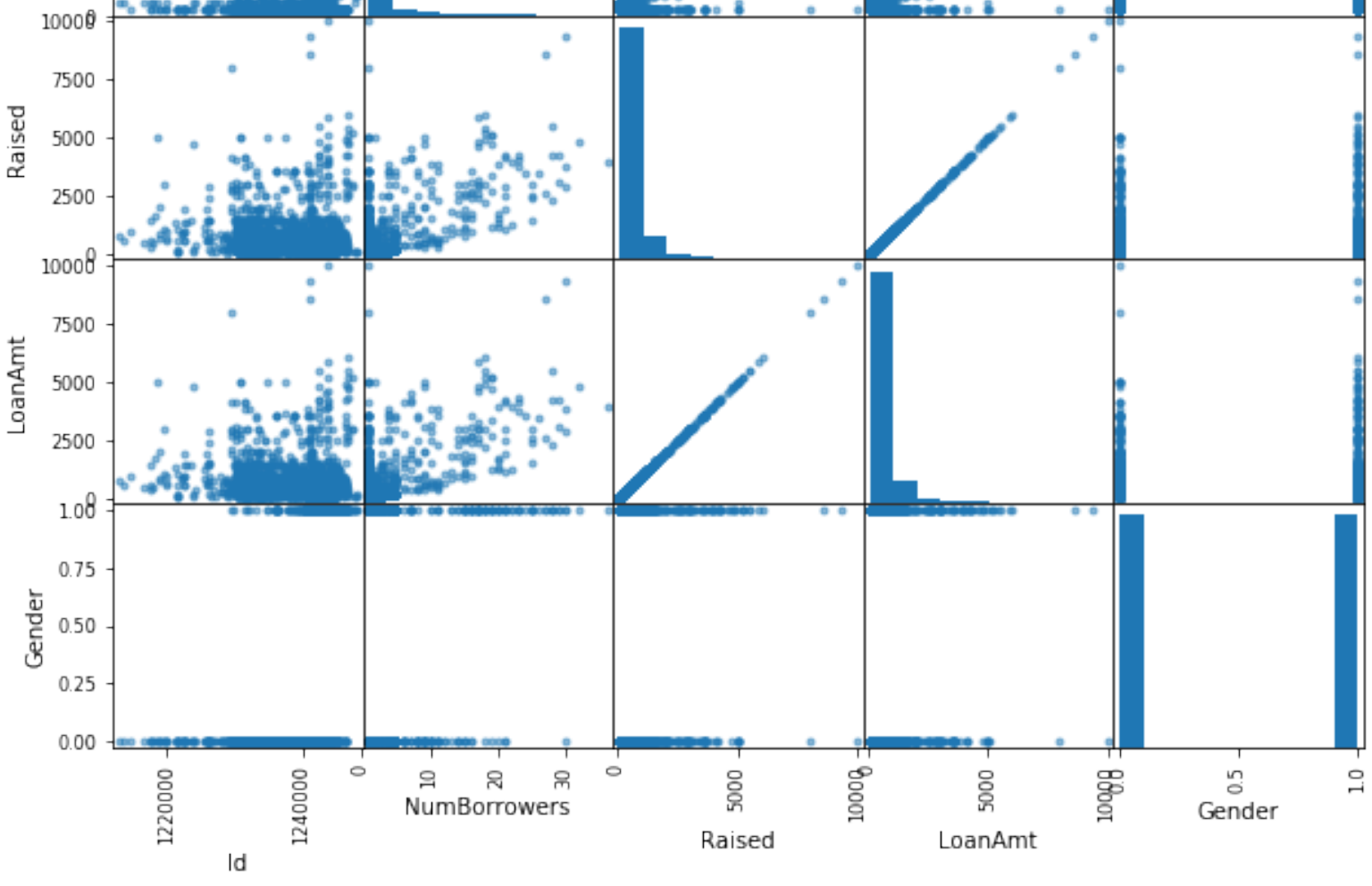
In [25]:

```
pd.scatter_matrix(df, figsize=(10,10))
```

Out[25]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x119750110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b956dd0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11b9dfd50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11ba5d110>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bae2150>]
,
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11bb48910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bbd5a10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bc49050>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bcd0210>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bd37b90>]
,
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11bdbae50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bd525d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11beb4c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11beefc50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11bfa1e50>]
,
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11c029e90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c09ce10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c122e50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c197710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c21d8d0>]
,
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11c292290>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c315550>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c2a2b10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c3f1a90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11c477910>]
], dtype=object)
```





In [26]:

```
df.corr()
```

Out[26]:

	Id	NumBorrowers	Raised	LoanAmt	Gender
Id	1.000000	0.027806	-0.110374	-0.110374	0.697676
NumBorrowers	0.027806	1.000000	0.565802	0.565802	0.064726
Raised	-0.110374	0.565802	1.000000	1.000000	-0.095921
LoanAmt	-0.110374	0.565802	1.000000	1.000000	-0.095921
Gender	0.697676	0.064726	-0.095921	-0.095921	1.000000

In [27]:

```
df.describe()
```

Out[27]:

	Id	NumBorrowers	Raised	LoanAmt	Gender
count	3.000000e+03	3000.000000	3000.000000	3000.000000	3000.000000
mean	1.238881e+06	1.922667	598.991667	598.991667	0.500000
std	5.088642e+03	3.347998	781.940483	781.940483	0.500083
min	1.212925e+06	1.000000	50.000000	50.000000	0.000000
25%	1.235032e+06	1.000000	200.000000	200.000000	0.000000
50%	1.240805e+06	1.000000	350.000000	350.000000	0.500000
75%	1.242654e+06	1.000000	700.000000	700.000000	1.000000
max	1.247890e+06	36.000000	10000.000000	10000.000000	1.000000

In [29]:

```
# Now preparing df for modeling: remove unnecessary columns, scale NumBorrowers and
# make dummy var for Country and Activity, remove some of the smaller values from C
# and Activity
```

In [30]:

```
df.head()
```

Out[30]:

	Id	Name	Country	Activity	NumBorrowers	Raised	LoanAmt	Gender
0	1233677	Robert	Armenia	Agriculture	1	1500	1500	0
1	1232922	Aleksan	Armenia	Agriculture	1	1500	1500	0
2	1242118	Elina	Armenia	Agriculture	1	3000	3000	1
3	1236073	Aquilino	Bolivia	Agriculture	1	300	300	0
4	1241034	Teeltaaba De Zongo Group	BurkinaFaso	Agriculture	5	825	825	1

In [31]:

```
df.drop(df.columns[[0,1,5]], axis=1, inplace=True)
```

In [85]:

```
df.head(2)
```

Out[85]:

	Country	Activity	NumBorrowers	LoanAmt	Gender
0	Armenia	Agriculture	0.0	0.145729	0
1	Armenia	Agriculture	0.0	0.145729	0

In [33]:

```
df['Country'].value_counts()
```

Out[33]:

```
Philippines          822
Cambodia             301
Kenya                243
ElSalvador           163
Ecuador              146
Uganda               120
Peru                 119
Tajikistan            74
Paraguay              61
Lebanon               57
Nicaragua             56
Colombia              55
Pakistan              49
DemRepCongo           46
Madagascar            42
India                 37
Palestine             36
Honduras              35
Vietnam               33
Egypt                 30
Cameroon              29
Indonesia             27
SierraLeone           27
Mexico                27
Kyrgyzstan            26
BurkinaFaso           25
Tanzania              23
Liberia               21
Armenia               19
Togo                  19
                     ...
Mali                  13
Guatemala             12
Mozambique            11
Malawi                10
TimorLeste             9
```

```
Bolivia                 8
Moldova                 8
Myanmar                 8
Thailand                8
Yemen                   7
Rwanda                  7
UnitedStates            7
SolomonIslands          6
CostaRica               5
Albania                 4
Brazil                  4
Georgia                 4
Burundi                 4
Zimbabwe                4
Kosovo                  3
SouthAfrica             2
Lesotho                 1
Samoa                   1
Turkey                  1
NegrosOccidental        1
Zambia                  1
PuertoRico              1
Ukraine                 1
Jalalabatregion         1
Nepal                   1
Name: Country, dtype: int64
```

In [34]:

```
threshold = 20
for col in df.columns:
    value_counts = df['Country'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['Country'].replace(to_remove, inplace=True)
print df['Country'].value_counts()
```

```
Philippines    841
Cambodia       309
Kenya          268
ElSalvador     201
Ecuador        152
Uganda         148
Peru           122
Tajikistan      78
Madagascar      77
Paraguay        62
Lebanon         61
Nicaragua       58
Colombia        56
DemRepCongo     54
India           51
Vietnam         51
Pakistan        49
Tanzania        40
SierraLeone     37
Palestine       36
Honduras        35
Mexico          33
Kyrgyzstan      33
Egypt           32
Cameroon        31
Indonesia       29
BurkinaFaso     26
Liberia         26
Armenia          4
Name: Country, dtype: int64
```

In [35]:

```
df['Activity'].value_counts()
```

Out[35]:

```
HomeAppliances                292
Farming                       283
GeneralStore                  235
Pigs                          151
Agriculture                   122
Highereducationcosts          122
PersonalHousingExpenses        92
FoodProductionSales            83
```

PrimarySecondaryschoolcosts	79
HomeEnergy	75
Retail	71
Cattle	65
FruitsVegetables	64
Livestock	63
Food	59
FishSelling	58
PersonalMedicalExpenses	56
Fishing	54
MotorcycleTransport	54
ClothingSales	52
GroceryStore	49
Educationprovider	48
Poultry	46
FoodMarket	37
Crafts	31
Services	29
FoodStall	29
AnimalSales	27
Cereals	26
Tailoring	25
...	
ConsumerGoods	2
MusicalInstruments	2
VeterinarySales	2
Pub	2
TimberSales	2
Cobbler	2
RestaurantCaterer	1
WasteManagement	1
SocialEnterprise	1
Knitting	1
Cement	1
SportingGoodSales	1
Games	1
Technology	1
BalutMaking	1
Communications	1
FuneralExpenses	1
Tourism	1
MachineryRental	1
CheeseMaking	1
NaturalMedicines	1
Health	1
Other	1
RenewableEnergyProducts	1
Arts	1
BicycleRepair	1
OfficeSupplies	1
CleaningServices	1
Blacksmith	1
WeddingExpenses	1

```
Name: Activity, dtype: int64
```

In [36]:

```
threshold = 20
for col in df.columns:
    value_counts = df['Activity'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['Activity'].replace(to_remove, inplace=True)
print df['Activity'].value_counts()
```

```
HomeAppliances                292
Farming                       283
GeneralStore                  235
Pigs                          154
Highereducationcosts          122
Agriculture                   122
AnimalSales                   116
PrimarySecondaryschoolcosts   106
HomeEnergy                    103
Livestock                     101
Tailoring                      99
PersonalHousingExpenses        92
PersonalMedicalExpenses        91
ClothingSales                  84
FoodProductionSales            83
Educationprovider              79
Retail                         74
FruitsVegetables               72
Cattle                         66
Fishing                        63
MotorcycleTransport            63
Cereals                        61
GroceryStore                   59
Food                           59
FishSelling                    58
Poultry                        46
Sewing                         42
FoodMarket                     37
Crafts                         31
FoodStall                      29
Services                       29
FurnitureMaking                25
Dairy                          24
Name: Activity, dtype: int64
```

In [37]:

```
df['NumBorrowers'].value_counts()
```

Out[37]:

```
1     2465
2      181
4       84
5       69
3       65
9       11
8       10
15       9
7        9
10       9
16       8
6        8
17       8
11       8
18       6
21       6
19       6
14       5
20       5
25       4
30       3
28       3
13       3
23       3
12       2
29       2
22       2
27       2
24       1
26       1
36       1
32       1
Name: NumBorrowers, dtype: int64
```

In [38]:

```
threshold = 40
for col in df.columns:
    value_counts = df['NumBorrowers'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['NumBorrowers'].replace(to_remove, inplace=True)
print df['NumBorrowers'].value_counts()
```

```
1    2578
2     192
4      86
5      77
3      67
Name: NumBorrowers, dtype: int64
```

In [39]:

```
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
```

In [40]:

```
df[['NumBorrowers']] = scale.fit_transform(df[['NumBorrowers']].as_matrix())
df[['LoanAmt']] = scale.fit_transform(df[['LoanAmt']].as_matrix())
```

```
/Users/krys/anaconda2/lib/python2.7/site-packages/sklearn/utils/valida
tion.py:429: DataConversionWarning: Data with input dtype int64 was co
nverted to float64 by MinMaxScaler.
  warnings.warn(msg, _DataConversionWarning)
```

In [41]:

```
dummy_country = pd.get_dummies(df[['Country']], prefix='Country')
dummy_country.astype(float)
print dummy_country.head()
```

```
   Country_Armenia  Country_BurkinaFaso  Country_Cambodia  Country_Cam
eroon  \
0                1                    0                 0
0
1                1                    0                 0
0
2                1                    0                 0
0
3                1                    0                 0
0
4                0                    1                 0
0

   Country_Colombia  Country_DemRepCongo  Country_Ecuador  Country_Egy
pt  \
0                 0                    0                 0
```

```
0
1                    0                      0                    0
0
2                    0                      0                    0
0
3                    0                      0                    0
0
4                    0                      0                    0
0

   Country_ElSalvador  Country_Honduras        ...          Country_Paki
stan  \
0                     0                    0       ...
0
1                     0                    0       ...
0
2                     0                    0       ...
0
3                     0                    0       ...
0
4                     0                    0       ...
0

   Country_Palestine  Country_Paraguay  Country_Peru  Country_Philippi
nes  \
0                    0                   0             0
0
1                    0                   0             0
0
2                    0                   0             0
0
3                    0                   0             0
0
4                    0                   0             0
0

   Country_SierraLeone  Country_Tajikistan  Country_Tanzania  Country_
Uganda  \
0                     0                    0                    0
0
1                     0                    0                    0
0
2                     0                    0                    0
0
3                     0                    0                    0
0
4                     0                    0                    0
0

   Country_Vietnam
0                0
1                0
2                0
```

```
3                  0
4                  0

[5 rows x 29 columns]
```

In [42]:

```
dummy_country.dtypes
```

Out[42]:

```
Country_Armenia        uint8
Country_BurkinaFaso    uint8
Country_Cambodia       uint8
Country_Cameroon       uint8
Country_Colombia       uint8
Country_DemRepCongo    uint8
Country_Ecuador        uint8
Country_Egypt          uint8
Country_ElSalvador     uint8
Country_Honduras       uint8
Country_India          uint8
Country_Indonesia      uint8
Country_Kenya          uint8
Country_Kyrgyzstan     uint8
Country_Lebanon        uint8
Country_Liberia        uint8
Country_Madagascar     uint8
Country_Mexico         uint8
Country_Nicaragua      uint8
Country_Pakistan       uint8
Country_Palestine      uint8
Country_Paraguay       uint8
Country_Peru           uint8
Country_Philippines    uint8
Country_SierraLeone    uint8
Country_Tajikistan     uint8
Country_Tanzania       uint8
Country_Uganda         uint8
Country_Vietnam        uint8
dtype: object
```

In [43]:

```
dummy_act = pd.get_dummies(df[['Activity']], prefix='Act')
dummy_act.astype(int)
print dummy_act.head()
```

```
   Act_Agriculture  Act_AnimalSales  Act_Cattle  Act_Cereals  \
0                1                0           0            0
1                1                0           0            0
2                1                0           0            0
3                1                0           0            0
4                1                0           0            0
```

```
   Act_ClothingSales  Act_Crafts  Act_Dairy  Act_Educationprovider  \
0                  0           0          0                      0
1                  0           0          0                      0
2                  0           0          0                      0
3                  0           0          0                      0
4                  0           0          0                      0

   Act_Farming  Act_FishSelling      ...        Act_MotorcycleTranspor
t  \
0            0                0      ...
0
1            0                0      ...
0
2            0                0      ...
0
3            0                0      ...
0
4            0                0      ...
0

   Act_PersonalHousingExpenses  Act_PersonalMedicalExpenses  Act_Pigs
\
0                            0                            0         0
1                            0                            0         0
2                            0                            0         0
3                            0                            0         0
4                            0                            0         0

   Act_Poultry  Act_PrimarySecondaryschoolcosts  Act_Retail  Act_Servi
ces  \
0            0                                0           0
0
1            0                                0           0
0
2            0                                0           0
0
3            0                                0           0
0
4            0                                0           0
0

   Act_Sewing  Act_Tailoring
0           0              0
1           0              0
2           0              0
3           0              0
4           0              0

[5 rows x 33 columns]
```

In [44]:

```
data = pd.concat([dummy_country,
                  dummy_act,
                  df[['NumBorrowers']],
                 df[['LoanAmt']],
                 df[['Gender']]],
                 axis=1, join='inner')

print data.head()
```

```
   Country_Armenia  Country_BurkinaFaso  Country_Cambodia  Country_Cam
eroon  \
0                1                    0                 0
0
1                1                    0                 0
0
2                1                    0                 0
0
3                1                    0                 0
0
4                0                    1                 0
0

   Country_Colombia  Country_DemRepCongo  Country_Ecuador  Country_Egy
pt  \
0                 0                    0                0
0
1                 0                    0                0
0
2                 0                    0                0
0
3                 0                    0                0
0
4                 0                    0                0
0

   Country_ElSalvador  Country_Honduras   ...    Act_Pigs  Act_Poultry
\
0                   0                 0   ...           0            0
1                   0                 0   ...           0            0
2                   0                 0   ...           0            0
3                   0                 0   ...           0            0
4                   0                 0   ...           0            0

   Act_PrimarySecondaryschoolcosts  Act_Retail  Act_Services  Act_Sewi
ng  \
0                                0           0             0
0
1                                0           0             0
0
2                                0           0             0
0
3                                0           0             0
0
4                                0           0             0
```

0

	Act_Tailoring	NumBorrowers	LoanAmt	Gender
0	0	0.0	0.145729	0
1	0	0.0	0.145729	0
2	0	0.0	0.296482	1
3	0	0.0	0.025126	0
4	0	1.0	0.077889	1

[5 rows x 65 columns]

In [45]:

```
data.dtypes
```

Out[45]:

```
Country_Armenia                uint8
Country_BurkinaFaso            uint8
Country_Cambodia               uint8
Country_Cameroon               uint8
Country_Colombia               uint8
Country_DemRepCongo            uint8
Country_Ecuador                uint8
Country_Egypt                  uint8
Country_ElSalvador             uint8
Country_Honduras               uint8
Country_India                  uint8
Country_Indonesia              uint8
Country_Kenya                  uint8
Country_Kyrgyzstan             uint8
Country_Lebanon                uint8
Country_Liberia                uint8
Country_Madagascar             uint8
Country_Mexico                 uint8
Country_Nicaragua              uint8
Country_Pakistan               uint8
Country_Palestine              uint8
Country_Paraguay               uint8
Country_Peru                   uint8
Country_Philippines            uint8
Country_SierraLeone            uint8
Country_Tajikistan             uint8
Country_Tanzania               uint8
Country_Uganda                 uint8
Country_Vietnam                uint8
Act_Agriculture                uint8
                               ...
Act_Dairy                      uint8
Act_Educationprovider          uint8
Act_Farming                    uint8
Act_FishSelling                uint8
Act_Fishing                    uint8
```

```
Act_Food                          uint8
Act_FoodMarket                    uint8
Act_FoodProductionSales           uint8
Act_FoodStall                     uint8
Act_FruitsVegetables              uint8
Act_FurnitureMaking               uint8
Act_GeneralStore                  uint8
Act_GroceryStore                  uint8
Act_Highereducationcosts          uint8
Act_HomeAppliances                uint8
Act_HomeEnergy                    uint8
Act_Livestock                     uint8
Act_MotorcycleTransport           uint8
Act_PersonalHousingExpenses       uint8
Act_PersonalMedicalExpenses       uint8
Act_Pigs                          uint8
Act_Poultry                       uint8
Act_PrimarySecondaryschoolcosts   uint8
Act_Retail                        uint8
Act_Services                      uint8
Act_Sewing                        uint8
Act_Tailoring                     uint8
NumBorrowers                    float64
LoanAmt                         float64
Gender                            int64
dtype: object
```

In [46]:

```
from sklearn.cross_validation import train_test_split
train, test = train_test_split(data, train_size=.80, test_size=.20)
```

```
/Users/krys/anaconda2/lib/python2.7/site-packages/sklearn/cross_valida
tion.py:44: DeprecationWarning: This module was deprecated in version
0.18 in favor of the model_selection module into which all the refacto
red classes and functions are moved. Also note that the interface of t
he new CV iterators are different from that of this module. This modul
e will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [47]:

```
print train.shape
print test.shape
```

```
(2400, 65)
(600, 65)
```

In [48]:

```
train_X = train.iloc[:,:-1]
train_y = train['Gender']
test_X = test.iloc[:,:-1]
test_y = test['Gender']
```

In [49]:

```
from sklearn.datasets import make_classification
from sklearn.ensemble import ExtraTreesClassifier
```

In [50]:

```
forest = ExtraTreesClassifier(n_estimators=64,
                              random_state=0)

forest.fit(train_X, train_y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]
```

In [51]:

```
print("Feature ranking:")

for f in range(train_X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature 63 (0.349935)
2. feature 23 (0.126598)
3. feature 62 (0.028828)
4. feature 46 (0.028130)
5. feature 8 (0.020599)
6. feature 52 (0.017801)
7. feature 27 (0.017205)
8. feature 44 (0.015995)
9. feature 10 (0.013752)
10. feature 37 (0.013618)
11. feature 29 (0.012302)
12. feature 22 (0.012289)
13. feature 36 (0.012048)
14. feature 48 (0.011813)
15. feature 57 (0.011377)
16. feature 33 (0.010847)
17. feature 6 (0.010810)
18. feature 12 (0.010254)
19. feature 50 (0.009864)
20. feature 42 (0.009794)
21. feature 18 (0.009203)
22. feature 15 (0.008852)
```

```
23. feature 16 (0.008809)
24. feature 61 (0.008517)
25. feature 24 (0.008417)
26. feature 30 (0.008361)
27. feature 4 (0.008226)
28. feature 47 (0.008087)
29. feature 14 (0.007957)
30. feature 38 (0.007655)
31. feature 43 (0.007631)
32. feature 40 (0.007624)
33. feature 55 (0.007604)
34. feature 19 (0.007388)
35. feature 25 (0.007294)
36. feature 5 (0.006958)
37. feature 60 (0.006544)
38. feature 51 (0.006430)
39. feature 53 (0.006283)
40. feature 41 (0.005981)
41. feature 54 (0.005910)
42. feature 31 (0.005764)
43. feature 21 (0.005650)
44. feature 3 (0.005649)
45. feature 2 (0.005582)
46. feature 59 (0.005475)
47. feature 58 (0.005212)
48. feature 32 (0.005176)
49. feature 45 (0.004874)
50. feature 28 (0.004863)
51. feature 9 (0.004832)
52. feature 56 (0.004356)
53. feature 7 (0.004242)
54. feature 34 (0.004224)
55. feature 26 (0.004004)
56. feature 11 (0.003888)
57. feature 20 (0.003615)
58. feature 39 (0.003554)
59. feature 35 (0.003511)
60. feature 49 (0.003445)
61. feature 17 (0.002797)
62. feature 1 (0.002766)
63. feature 13 (0.002497)
64. feature 0 (0.000431)
```

In [52]:

```
forest = ExtraTreesClassifier(n_estimators=64,
                              random_state=0)

forest.fit(test_X, test_y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]
```

In [53]:

```
print("Feature ranking:")

for f in range(test_X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature 63 (0.285348)
2. feature 23 (0.108702)
3. feature 22 (0.035037)
4. feature 27 (0.022873)
5. feature 8 (0.021630)
6. feature 24 (0.020118)
7. feature 62 (0.019774)
8. feature 12 (0.017790)
9. feature 31 (0.016590)
10. feature 51 (0.016043)
11. feature 46 (0.015843)
12. feature 29 (0.015640)
13. feature 52 (0.015532)
14. feature 42 (0.014036)
15. feature 15 (0.013672)
16. feature 48 (0.013395)
17. feature 39 (0.012951)
18. feature 14 (0.012720)
19. feature 35 (0.011257)
20. feature 47 (0.011051)
21. feature 59 (0.010895)
22. feature 37 (0.010838)
23. feature 33 (0.010676)
24. feature 3 (0.010518)
25. feature 32 (0.010477)
26. feature 38 (0.010371)
27. feature 55 (0.009955)
28. feature 25 (0.009559)
29. feature 16 (0.009429)
30. feature 6 (0.009326)
31. feature 10 (0.009311)
32. feature 2 (0.009193)
33. feature 18 (0.008658)
34. feature 43 (0.008403)
35. feature 28 (0.008045)
36. feature 36 (0.007932)
37. feature 13 (0.007837)
38. feature 34 (0.007259)
39. feature 41 (0.006782)
40. feature 40 (0.006576)
41. feature 5 (0.006368)
42. feature 21 (0.006259)
43. feature 57 (0.006174)
44. feature 30 (0.006038)
45. feature 17 (0.006030)
```

```
46. feature 61 (0.005998)
47. feature 44 (0.005939)
48. feature 50 (0.005862)
49. feature 1 (0.005785)
50. feature 11 (0.005758)
51. feature 53 (0.005714)
52. feature 9 (0.005676)
53. feature 56 (0.005589)
54. feature 4 (0.005460)
55. feature 20 (0.004961)
56. feature 49 (0.004674)
57. feature 26 (0.004669)
58. feature 58 (0.004503)
59. feature 60 (0.003963)
60. feature 45 (0.003930)
61. feature 7 (0.003109)
62. feature 19 (0.002972)
63. feature 54 (0.001846)
64. feature 0 (0.000678)
```

In [54]:

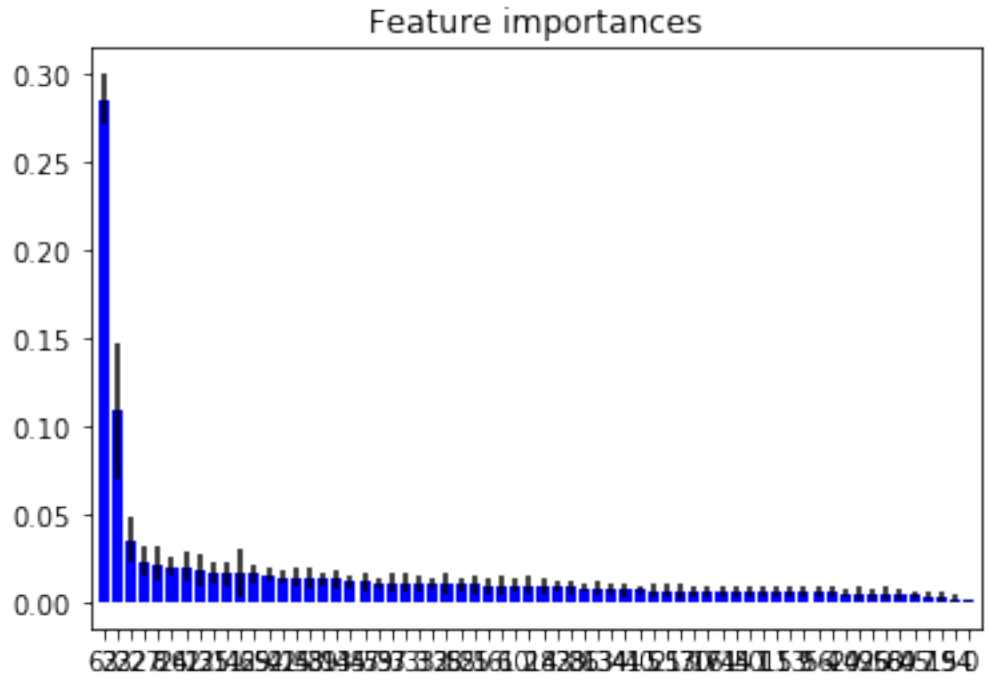
```
cols = test_X.columns
for feat in range(test_X.shape[1]):
    print("%d. feature %s (%f)" % (feat + 1, cols[feat], importances[indices[feat]])
```

```
1. feature Country_Armenia (0.285348)
2. feature Country_BurkinaFaso (0.108702)
3. feature Country_Cambodia (0.035037)
4. feature Country_Cameroon (0.022873)
5. feature Country_Colombia (0.021630)
6. feature Country_DemRepCongo (0.020118)
7. feature Country_Ecuador (0.019774)
8. feature Country_Egypt (0.017790)
9. feature Country_ElSalvador (0.016590)
10. feature Country_Honduras (0.016043)
11. feature Country_India (0.015843)
12. feature Country_Indonesia (0.015640)
13. feature Country_Kenya (0.015532)
14. feature Country_Kyrgyzstan (0.014036)
15. feature Country_Lebanon (0.013672)
16. feature Country_Liberia (0.013395)
17. feature Country_Madagascar (0.012951)
18. feature Country_Mexico (0.012720)
19. feature Country_Nicaragua (0.011257)
20. feature Country_Pakistan (0.011051)
21. feature Country_Palestine (0.010895)
22. feature Country_Paraguay (0.010838)
23. feature Country_Peru (0.010676)
24. feature Country_Philippines (0.010518)
25. feature Country_SierraLeone (0.010477)
26. feature Country_Tajikistan (0.010371)
27. feature Country_Tanzania (0.009955)
28. feature Country_Uganda (0.009559)
```

29. feature Country_Vietnam (0.009429)
30. feature Act_Agriculture (0.009326)
31. feature Act_AnimalSales (0.009311)
32. feature Act_Cattle (0.009193)
33. feature Act_Cereals (0.008658)
34. feature Act_ClothingSales (0.008403)
35. feature Act_Crafts (0.008045)
36. feature Act_Dairy (0.007932)
37. feature Act_Educationprovider (0.007837)
38. feature Act_Farming (0.007259)
39. feature Act_FishSelling (0.006782)
40. feature Act_Fishing (0.006576)
41. feature Act_Food (0.006368)
42. feature Act_FoodMarket (0.006259)
43. feature Act_FoodProductionSales (0.006174)
44. feature Act_FoodStall (0.006038)
45. feature Act_FruitsVegetables (0.006030)
46. feature Act_FurnitureMaking (0.005998)
47. feature Act_GeneralStore (0.005939)
48. feature Act_GroceryStore (0.005862)
49. feature Act_Highereducationcosts (0.005785)
50. feature Act_HomeAppliances (0.005758)
51. feature Act_HomeEnergy (0.005714)
52. feature Act_Livestock (0.005676)
53. feature Act_MotorcycleTransport (0.005589)
54. feature Act_PersonalHousingExpenses (0.005460)
55. feature Act_PersonalMedicalExpenses (0.004961)
56. feature Act_Pigs (0.004674)
57. feature Act_Poultry (0.004669)
58. feature Act_PrimarySecondaryschoolcosts (0.004503)
59. feature Act_Retail (0.003963)
60. feature Act_Services (0.003930)
61. feature Act_Sewing (0.003109)
62. feature Act_Tailoring (0.002972)
63. feature NumBorrowers (0.001846)
64. feature LoanAmt (0.000678)

In [55]:

```
plt.figure()
plt.title("Feature importances")
plt.bar(range(test_X.shape[1]), importances[indices],
       color="b", yerr=std[indices], align="center")
plt.xticks(range(test_X.shape[1]), indices)
plt.xlim([-1, test_X.shape[1]])
plt.show()
```



In [56]:

```
forest.score(test_X,test_y)
```

Out[56]:

```
0.94333333333333336
```

In [57]:

```
from sklearn.ensemble import RandomForestClassifier
randfor = RandomForestClassifier(n_estimators=64)
```

In [58]:

```
randfor = randfor.fit(train_X,train_y)
```

In [59]:

```
randfor.score(train_X,train_y)
```

Out[59]:

```
0.91500000000000004
```

In [60]:

```
randfor.predict_proba(train_X)
```

Out[60]:

```
array([[ 0.55034401,  0.44965599],
       [ 0.29166667,  0.70833333],
       [ 0.        ,  1.        ],
       ...,
       [ 0.953125  ,  0.046875  ],
       [ 0.26761533,  0.73238467],
       [ 0.953125  ,  0.046875  ]])
```

In [61]:

```
randfor.score(test_X,test_y)
```

Out[61]:

```
0.72833333333333339
```

In [62]:

```
y_predicted = randfor.predict(test_X)
print y_predicted
```

```
[0 0 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0 0 1 1
1 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 0 1 1 0 1 0 0
1 0
 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 0
1 1
 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0
0 1
 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 1
1 1
 0 1 0 0 0 0 0 1 1 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0
1 0
 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1
0 0
 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 1 1 0 1 0
0 0
 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 1
1 1
 0 1 0 0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1
0 0
 0 1 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 1 0 1 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0
1 0
 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0
0 1
 0 1 0 0 1 1 0 0 0 1 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0
0 0
 0 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 0
0 0
 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0
1 1
 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1
0 0
 1 1 1 1 0 0 1 0]
```

In [64]:

```
preds = randfor.predict(test_X)
pd.crosstab(test_y, preds, rownames=['actual'], colnames=['preds'])
```

Out[64]:

preds	0	1
actual		
0	245	72
1	91	192

In [65]:

```
from sklearn import linear_model
from sklearn.linear_model import LogisticRegressionCV
logit = linear_model.LogisticRegressionCV()
```

In [66]:

```
train.head()
```

Out[66]:

	Country_Armenia	Country_BurkinaFaso	Country_Cambodia	Country_Cameroon	C
1840	0	0	1	0	0
1604	0	0	0	0	0
1448	0	0	0	0	0
865	0	0	0	0	0
963	0	0	0	0	0

5 rows × 65 columns

In [67]:

```
X = train.iloc[:,:-1]
y = train['Gender']
```

In [68]:

```
logit.fit(X,y)
```

Out[68]:

```
LogisticRegressionCV(Cs=10, class_weight=None, cv=None, dual=False,
           fit_intercept=True, intercept_scaling=1.0, max_iter=100,
           multi_class='ovr', n_jobs=1, penalty='l2', random_state=Non
e,
           refit=True, scoring=None, solver='lbfgs', tol=0.0001, verbo
se=0)
```

In [73]:

```
print logit.intercept_
print logit.coef_
```

```
[-0.31891428]
[[ 0.69474796 -1.15002929  0.38722554  0.19742237 -1.32470553 -0.32025
107
   0.49666807 -1.40694841 -1.11186866 -1.64231893  2.12386434  0.53343
435
   0.01178297 -0.00901513 -1.11105883  2.53636527 -1.13869579 -0.98270
802
  -1.42041095  2.0643975   0.5345276   0.74093873 -0.88159302  1.49261
213
   1.7254793   0.14674508 -0.16785429 -1.34824822  0.28838747 -0.98849
304
  -0.01598812 -0.47926153  0.91434531  0.88155808 -1.06746485  0.28896
451
  -1.32206175 -0.66330467  1.99640994  0.50381458  0.88771373  1.30248
056
   1.44941053  0.90108505  2.26725978 -5.6810423   1.3229409   0.48497
978
  -0.56765942 -0.53545173 -0.06501613 -0.17300625 -1.72179541  0.33094
913
   0.03075125  0.34658618  0.17554673  0.28805461 -0.26583354 -1.98696
5
   0.9617425   0.15764314  0.50565712 -0.57582539]]
```

In [74]:

```
print logit.score(X,y)
```

```
0.745833333333
```

In [75]:

```
test_X = test.iloc[:,:-1]
test_y = test['Gender']
```

In [76]:

```
test_X.shape
```

Out[76]:

```
(600, 64)
```

In [77]:

```
logit.score(test_X,test_y)
```

Out[77]:

```
0.72833333333333339
```

In [78]:

```
y_testpred = logit.predict(test_X)
print y_testpred
```

```
[0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1
0 1
 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1 0 0 1 0 0
1 0
 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 1 0
1 0
 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0
0 1
 0 1 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1
1 1
 0 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0
0 0
 1 0 0 1 0 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 1 1 1 1
1 0
 0 0 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 1 0
0 0
 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1
1 1
 1 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 1 1
0 0
 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 1 1 0 0
1 0
 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 1 1 0 1 0 1 0 1 1
0 1
 0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1
0 0
 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0
0 0
 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 0
1 1
 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1
0 0
 1 1 1 1 0 0 1 0]
```

In [79]:

```
print logit.intercept_
print logit.coef_
```

```
[-0.31891428]
[[ 0.69474796 -1.15002929  0.38722554  0.19742237 -1.32470553 -0.32025
107
   0.49666807 -1.40694841 -1.11186866 -1.64231893  2.12386434  0.53343
435
   0.01178297 -0.00901513 -1.11105883  2.53636527 -1.13869579 -0.98270
802
  -1.42041095  2.0643975   0.5345276   0.74093873 -0.88159302  1.49261
213
   1.7254793   0.14674508 -0.16785429 -1.34824822  0.28838747 -0.98849
304
  -0.01598812 -0.47926153  0.91434531  0.88155808 -1.06746485  0.28896
451
  -1.32206175 -0.66330467  1.99640994  0.50381458  0.88771373  1.30248
056
   1.44941053  0.90108505  2.26725978 -5.6810423   1.3229409   0.48497
978
  -0.56765942 -0.53545173 -0.06501613 -0.17300625 -1.72179541  0.33094
913
   0.03075125  0.34658618  0.17554673  0.28805461 -0.26583354 -1.98696
5
   0.9617425   0.15764314  0.50565712 -0.57582539]]
```

In [140]:

```
coef_list = pd.DataFrame(zip(test_X.columns, np.transpose(logit.coef_)))
coef_list.values.tolist()
```

Out[140]:

```
[['Country_Armenia', array([ 0.09248569])],
 ['Country_BurkinaFaso', array([-0.49042576])],
 ['Country_Cambodia', array([ 0.20005138])],
 ['Country_Cameroon', array([-0.32949822])],
 ['Country_Colombia', array([-0.66983506])],
 ['Country_DemRepCongo', array([-0.02990461])],
 ['Country_Ecuador', array([ 0.67734474])],
 ['Country_Egypt', array([-0.66526186])],
 ['Country_ElSalvador', array([-0.81764888])],
 ['Country_Honduras', array([-0.75711038])],
 ['Country_India', array([ 1.17848116])],
 ['Country_Indonesia', array([-0.23736426])],
 ['Country_Kenya', array([ 0.07961847])],
 ['Country_Kyrgyzstan', array([-0.28722417])],
 ['Country_Lebanon', array([-0.81124608])],
 ['Country_Liberia', array([ 1.36470332])],
 ['Country_Madagascar', array([-0.40440215])],
 ['Country_Mexico', array([-0.45488202])],
 ['Country_Nicaragua', array([-1.10010541])],
```

```
['Country_Pakistan', array([ 1.33086162])],
['Country_Palestine', array([ 0.11889887])],
['Country_Paraguay', array([ 0.70465963])],
['Country_Peru', array([-1.167701])],
['Country_Philippines', array([ 1.475781])],
['Country_SierraLeone', array([ 1.33681285])],
['Country_Tajikistan', array([ 0.19334877])],
['Country_Tanzania', array([ 0.09073955])],
['Country_Uganda', array([-1.03668475])],
['Country_Vietnam', array([ 0.4159469])],
['Act_Agriculture', array([-0.87244955])],
['Act_AnimalSales', array([-0.15717138])],
['Act_Cattle', array([-0.95250599])],
['Act_Cereals', array([ 0.91020399])],
['Act_ClothingSales', array([ 0.65884189])],
['Act_Crafts', array([-0.48900776])],
['Act_Dairy', array([ 0.04493263])],
['Act_Educationprovider', array([-0.82152602])],
['Act_Farming', array([-0.6187559])],
['Act_FishSelling', array([ 0.98320154])],
['Act_Fishing', array([ 0.10225407])],
['Act_Food', array([ 0.42850501])],
['Act_FoodMarket', array([ 0.97156707])],
['Act_FoodProductionSales', array([ 0.87263359])],
['Act_FoodStall', array([ 0.51175913])],
['Act_FruitsVegetables', array([ 1.33037039])],
['Act_FurnitureMaking', array([-0.79062738])],
['Act_GeneralStore', array([ 0.94393579])],
['Act_GroceryStore', array([ 0.21344061])],
['Act_Highereducationcosts', array([-0.65845773])],
['Act_HomeAppliances', array([-0.24449435])],
['Act_HomeEnergy', array([-0.10672053])],
['Act_Livestock', array([-0.26989209])],
['Act_MotorcycleTransport', array([-1.45863516])],
['Act_PersonalHousingExpenses', array([ 0.06002704])],
['Act_PersonalMedicalExpenses', array([-0.10047796])],
['Act_Pigs', array([ 0.02806523])],
['Act_Poultry', array([-0.31054807])],
['Act_PrimarySecondaryschoolcosts', array([ 0.15630364])],
['Act_Retail', array([-0.13243916])],
['Act_Services', array([-1.02493836])],
['Act_Sewing', array([ 0.6433515])],
['Act_Tailoring', array([ 0.14969362])],
['NumBorrowers', array([ 0.20184228])],
['LoanAmt', array([-0.31806696])]]
```

In [80]:

```
#Logistic Regression
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression()
```

In [81]:

```
logreg.fit(train_X,train_y)
```

Out[81]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs
=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0
001,
          verbose=0, warm_start=False)
```

In [82]:

```
#TEST data - Accuracy
seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = LogisticRegression()
scoring = 'accuracy'
results = model_selection.cross_val_score(model, test_X, test_y,
                                          cv=kfold, scoring=scoring)
print('Accuracy: %.3f (%.3f)') % (results.mean(), results.std())
```

```
Accuracy: 0.710 (0.075)
```

In [83]:

```
#TEST data - AUC

seed = 7
kfold = model_selection.KFold(n_splits=10, random_state=seed)
model = LogisticRegression()
scoring = 'roc_auc'
results = model_selection.cross_val_score(model, test_X, test_y,
                                          cv=kfold, scoring=scoring)
print('AUC: %.3f (%.3f)') % (results.mean(), results.std())
```

```
AUC: 0.804 (0.062)
```

In [ ]:

```

```