In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pylab as plt
```

executed in 41.3s, finished 17:03:20 2019-07-10

In [2]:

```
train = pd.read_csv('train_technidus_clf.csv')
test = pd.read_csv('test_technidus_clf.csv')
```

executed in 722ms, finished 17:03:20 2019-07-10

1 Data Preprocessing

In [3]:

```
train.describe().T
```

executed in 446ms, finished 17:03:21 2019-07-10

Out[3]:

	count	mean	std	min	25%	50%	75%
CustomerID	7685.0	18763.213403	4832.429369	11001.0	14676.0	18493.0	22393.
HomeOwnerFlag	7685.0	0.696682	0.459721	0.0	0.0	1.0	1.
NumberCarsOwned	7685.0	1.569031	1.177871	0.0	1.0	2.0	2.
NumberChildrenAtHome	7685.0	1.230839	1.650189	0.0	0.0	0.0	2.
TotalChildren	7685.0	2.151464	1.727375	0.0	1.0	2.0	4.
YearlyIncome	7685.0	82323.659206	40368.317414	10063.0	50784.0	82158.0	109389.
AveMonthSpend	7685.0	76.383995	29.367313	26.0	53.0	71.0	89.
BikeBuyer	7685.0	0.496291	0.500019	0.0	0.0	0.0	1.

In [4]:

```
train.drop('CustomerID',axis=1).duplicated().sum()
```

executed in 257ms, finished 17:03:22 2019-07-10

Out[4]:

```
0
```

In [5]:

```
1  train.info()
```

executed in 82ms, finished 17:03:23 2019-07-10

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7685 entries, 0 to 7684
Data columns (total 25 columns):
CustomerID              7685 non-null int64
Title                   41 non-null object
FirstName               7685 non-null object
MiddleName              4457 non-null object
LastName                7685 non-null object
Suffix                  1 non-null object
AddressLine1            7685 non-null object
AddressLine2            135 non-null object
City                    7685 non-null object
StateProvinceName       7685 non-null object
CountryRegionName       7685 non-null object
PostalCode              7685 non-null object
PhoneNumber             7685 non-null object
BirthDate               7685 non-null object
Education               7685 non-null object
Occupation              7685 non-null object
Gender                  7685 non-null object
MaritalStatus           7685 non-null object
HomeOwnerFlag           7685 non-null int64
NumberCarsOwned         7685 non-null int64
NumberChildrenAtHome    7685 non-null int64
TotalChildren           7685 non-null int64
YearlyIncome            7685 non-null int64
AveMonthSpend           7685 non-null int64
BikeBuyer               7685 non-null int64
dtypes: int64(8), object(17)
memory usage: 1.5+ MB
```

In [6]:

```
1  #Print out all the columns that has less than 30% null values
2  nn_cols=[col for col in train.columns if train[col].count()<=0.7*len(train)]
3  print(nn_cols)
```

executed in 63ms, finished 17:03:24 2019-07-10

```
['Title', 'MiddleName', 'Suffix', 'AddressLine2']
```

In [7]:

```
1  train.drop(nn_cols,axis=1,inplace=True)
2  test.drop(nn_cols,axis=1,inplace=True)
```

executed in 50ms, finished 17:03:24 2019-07-10

In [8]:

```
1 train.isnull().sum()
```

executed in 51ms, finished 17:03:25 2019-07-10

Out[8]:

```
CustomerID              0
FirstName               0
LastName                0
AddressLine1            0
City                    0
StateProvinceName       0
CountryRegionName       0
PostalCode              0
PhoneNumber             0
BirthDate               0
Education               0
Occupation              0
Gender                  0
MaritalStatus           0
HomeOwnerFlag           0
NumberCarsOwned         0
NumberChildrenAtHome    0
TotalChildren           0
YearlyIncome            0
AveMonthSpend           0
BikeBuyer               0
dtype: int64
```

In [9]:

```
1 train.nunique()
```

executed in 307ms, finished 17:03:26 2019-07-10

Out[9]:

```
CustomerID              7685
FirstName                606
LastName                 296
AddressLine1            6581
City                     250
StateProvinceName         45
CountryRegionName          6
PostalCode               301
PhoneNumber             3882
BirthDate               5173
Education                  5
Occupation                 5
Gender                     2
MaritalStatus              2
HomeOwnerFlag              2
NumberCarsOwned            5
NumberChildrenAtHome       6
TotalChildren              6
YearlyIncome            7489
AveMonthSpend            148
BikeBuyer                  2
dtype: int64
```

In [10]:

```
uninformative=['FirstName','LastName','CustomerID','PhoneNumber','AddressLine1','Post
```

executed in 12ms, finished 17:03:26 2019-07-10

In [11]:

```
train.drop(uninformative,axis=1,inplace=True)
test.drop(uninformative,axis=1,inplace=True)
```

executed in 31ms, finished 17:03:27 2019-07-10

In [12]:

```
#Convert BirthDate to Year,Month
train['BirthYear']=pd.to_datetime(train['BirthDate']).dt.year;
train.drop(['BirthDate'],axis=1,inplace=True)

test['BirthYear']=pd.to_datetime(test['BirthDate']).dt.year;
test.drop(['BirthDate'],axis=1,inplace=True)
```

executed in 6.23s, finished 17:03:34 2019-07-10

In [13]:

```
train.nunique()
```

executed in 90ms, finished 17:03:34 2019-07-10

Out[13]:

```
City                    250
StateProvinceName        45
CountryRegionName         6
Education                 5
Occupation                5
Gender                    2
MaritalStatus             2
HomeOwnerFlag             2
NumberCarsOwned           5
NumberChildrenAtHome      6
TotalChildren             6
YearlyIncome           7489
AveMonthSpend           148
BikeBuyer                 2
BirthYear                64
dtype: int64
```

In [14]:

```
cat_col=[col for col in train.columns if train[col].nunique()<10] + ['StateProvinceNa
num_col=list(set(list(train.columns))-set(cat_col))
```

executed in 43ms, finished 17:03:35 2019-07-10

In [15]:

```
print('Categorical features are:',cat_col)
print('')
print('Numerical features are:',num_col)
```

executed in 24ms, finished 17:03:35 2019-07-10

```
Categorical features are: ['CountryRegionName', 'Education', 'Occupation',
'Gender', 'MaritalStatus', 'HomeOwnerFlag', 'NumberCarsOwned', 'NumberChildr
enAtHome', 'TotalChildren', 'BikeBuyer', 'StateProvinceName', 'City']

Numerical features are: ['BirthYear', 'AveMonthSpend', 'YearlyIncome']
```

2 EDA

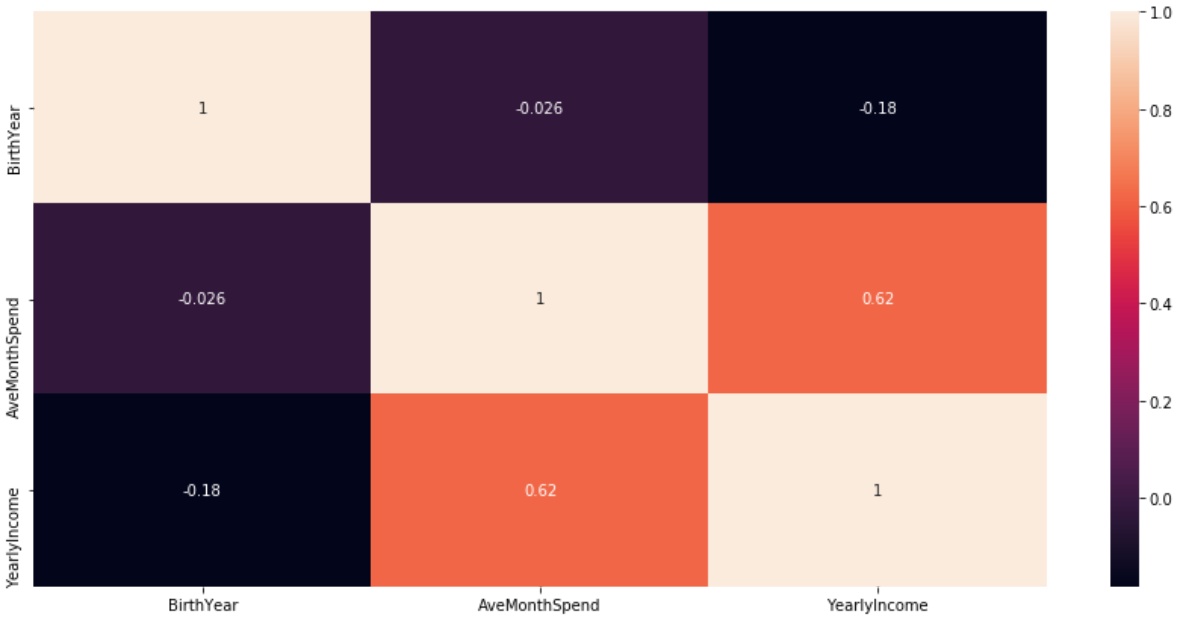
In [16]:

```
plt.figure(figsize=(15,7))
sns.heatmap(train[num_col].corr(),annot=True)
```

executed in 2.59s, finished 17:03:38 2019-07-10

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14cbff43f28>
```



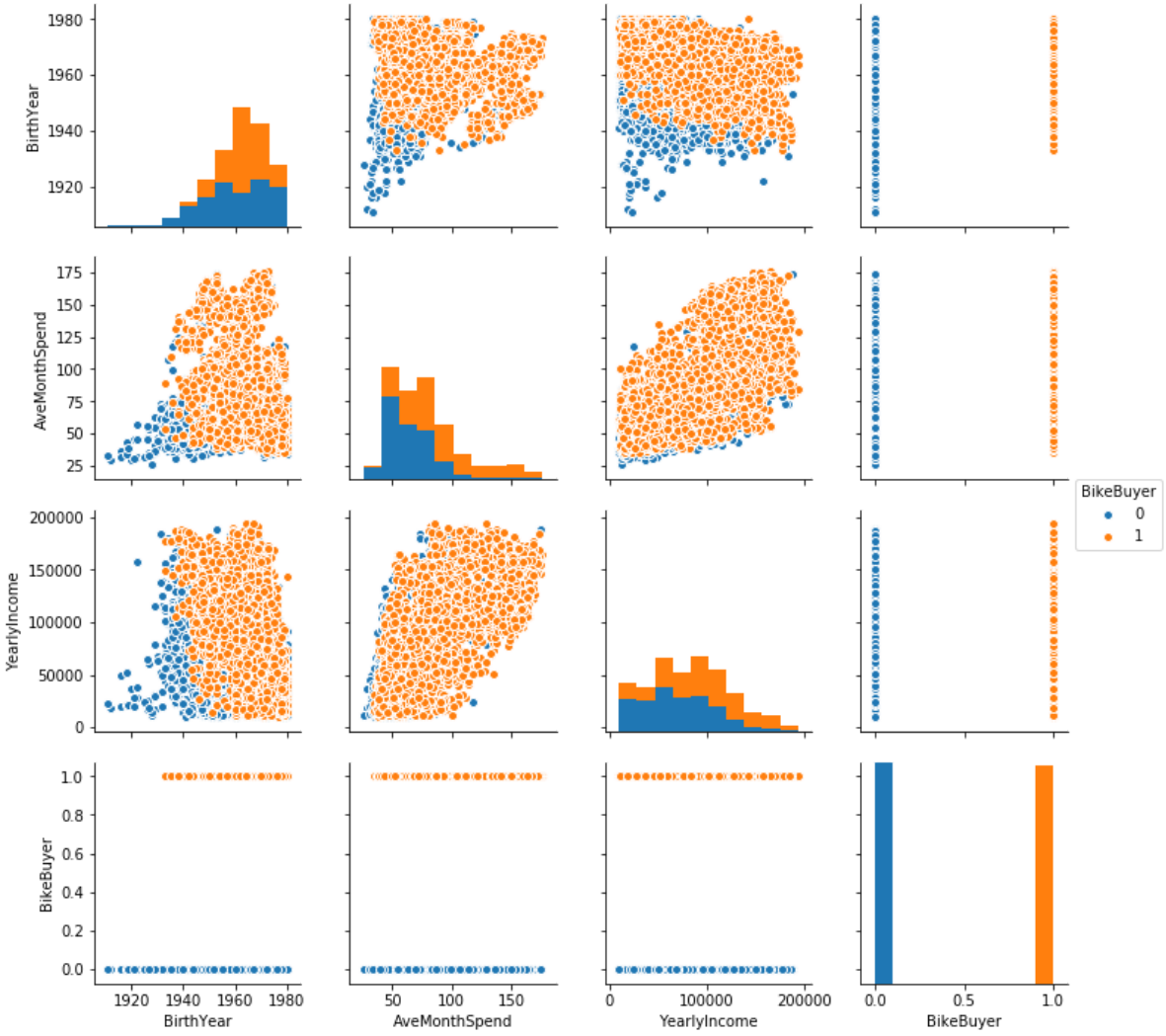
In [17]:

```
1  sc=num_col + ['BikeBuyer']
2  sns.pairplot(train[sc],hue='BikeBuyer')
```

executed in 10.7s, finished 17:03:49 2019-07-10

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x14cc02dce80>
```



In [18]:

```
for col in cat_col:
    print(train[col].value_counts())
    print('')
```

executed in 185ms, finished 17:03:50 2019-07-10

```
United States     3226
Australia         1537
United Kingdom     820
France             731
Germany            726
Canada             645
Name: CountryRegionName, dtype: int64

Bachelors              2294
Partial College        2117
High School            1361
Graduate Degree        1324
Partial High School     589
Name: Education, dtype: int64

Professional      2485
Skilled Manual    1845
Management        1330
Clerical          1095
Manual             930
Name: Occupation, dtype: int64

M    4024
F    3661
Name: Gender, dtype: int64

M    3939
S    3746
Name: MaritalStatus, dtype: int64

1    5354
0    2331
Name: HomeOwnerFlag, dtype: int64

2    2521
1    2030
0    1681
3     826
4     627
Name: NumberCarsOwned, dtype: int64

0    4171
1     962
2     825
3     611
4     566
5     550
Name: NumberChildrenAtHome, dtype: int64

0    1851
2    1368
1    1353
4    1190
3     965
```

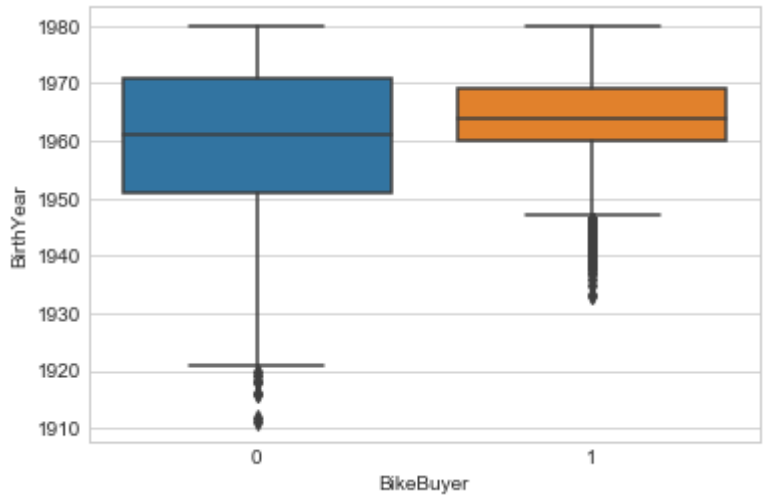
```
5     958
Name: TotalChildren, dtype: int64

0    3871
1    3814
Name: BikeBuyer, dtype: int64

California             1826
Washington              939
England                 820
New South Wales         656
British Columbia        641
Oregon                  443
Victoria                382
Queensland              343
Saarland                177
Nordrhein-Westfalen     164
Hessen                  154
Seine (Paris)           150
Seine Saint Denis       130
Hamburg                 123
Nord                    108
South Australia         106
Bayern                   95
Yveline                  77
Hauts de Seine           75
Essonne                  55
Tasmania                 50
Moselle                  25
Loiret                   22
Seine et Marne           20
Brandenburg              13
Garonne (Haute)          12
Val d'Oise               11
Charente-Maritime        11
Val de Marne             10
Pas de Calais             9
Somme                     9
Loir et Cher              7
Alberta                   4
Illinois                  4
Ohio                      2
New York                  2
Texas                     2
Wyoming                   1
Georgia                   1
South Carolina            1
North Carolina            1
Mississippi               1
Arizona                   1
Missouri                  1
Virginia                  1
Name: StateProvinceName, dtype: int64

London                 183
Paris                  150
Beaverton               93
Concord                 85
Beverly Hills           84
Burien                  81
Cliffside               79
```

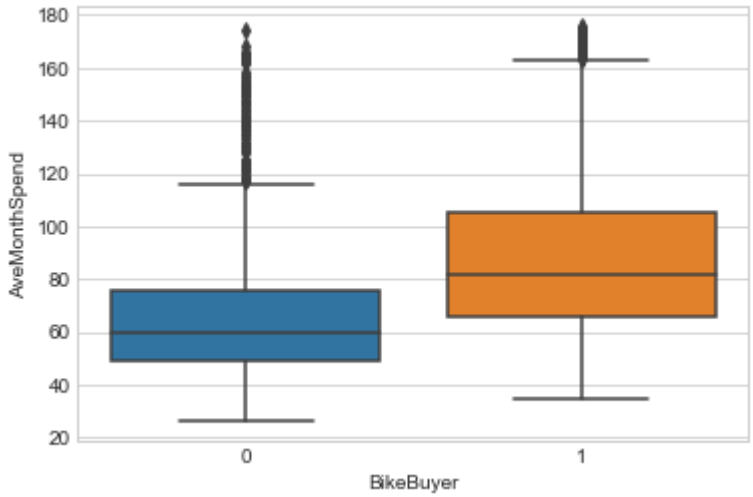
```
Bellflower              79
Bremerton               79
Berkeley                78
Burbank                 76
Chula Vista             75
Berlin                  72
Burlingame              72
Colma                   72
Bellingham              70
Cranbourne              54
Brisbane                52
Grossmont               51
Glendale                50
Hobart                  50
Hervey Bay              50
York                    49
Shawnee                 48
Warrnambool             48
Lake Oswego             47
Coronado                47
El Cajon                47
Port Hammond            46
Yakima                  46
                       ...
Roissy en Brie           9
Grevenbroich             8
Sèvres                   8
Boulogne-Billancourt     6
Leipzig                  6
Bothell                  4
Calgary                  4
Chicago                  3
Cincinnati               2
Clackamas                2
Newport Hills            1
Biloxi                   1
Chantilly                1
Branson                  1
Cerritos                 1
Carrollton               1
Kenmore                  1
Bluffton                 1
Camarillo                1
College Station          1
Charlotte                1
Byron                    1
Casper                   1
Chandler                 1
Sammamish                1
Cheektowaga              1
Chehalis                 1
Bellevue                 1
Carol Stream             1
Clay                     1
Name: City, Length: 250, dtype: int64
```

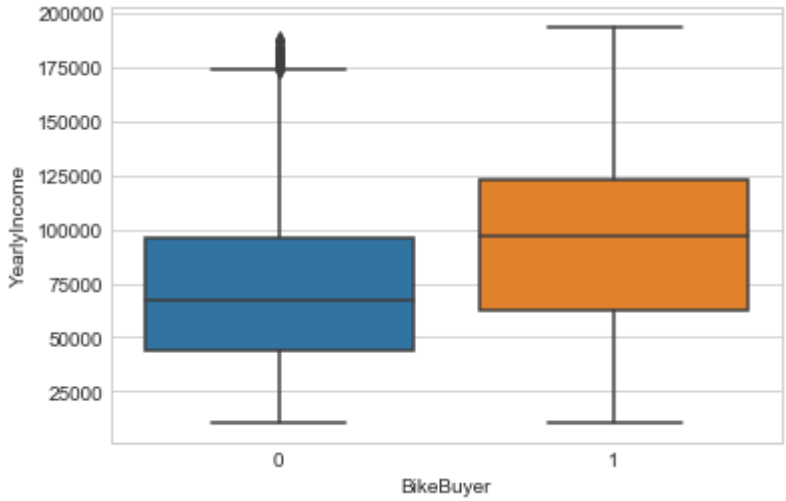
In [19]:

```
for col in num_col:
    sns.set_style("whitegrid")
    sns.boxplot('BikeBuyer', col, data=train)
    plt.xlabel('BikeBuyer')
    plt.ylabel(col)
    plt.show()
```

executed in 1.21s, finished 17:03:52 2019-07-10





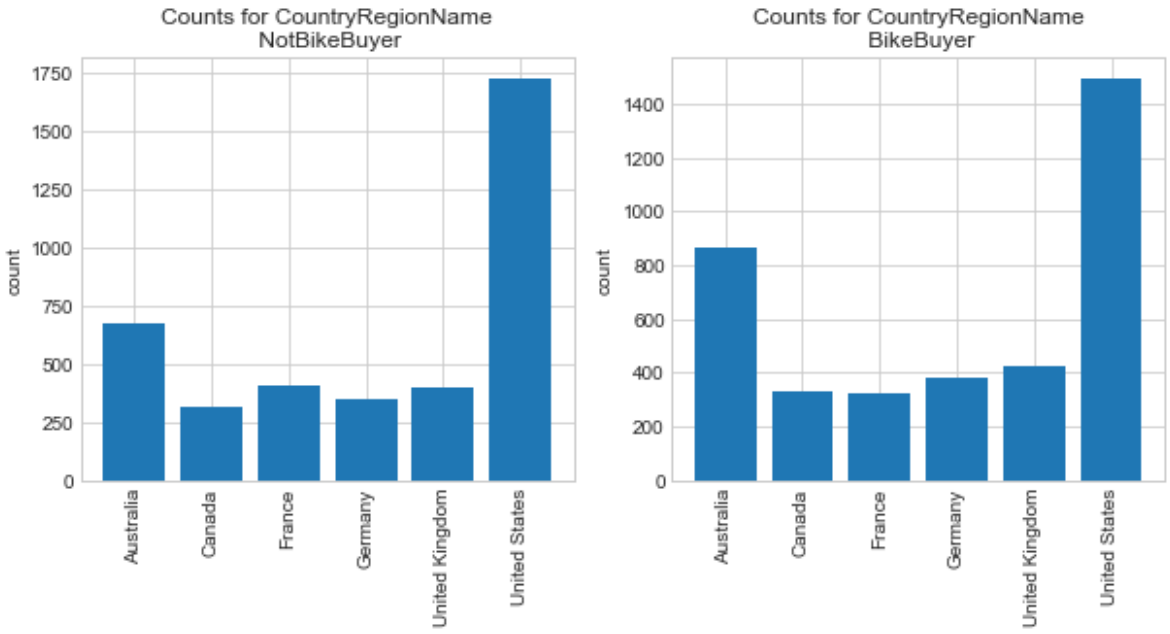


In [20]:

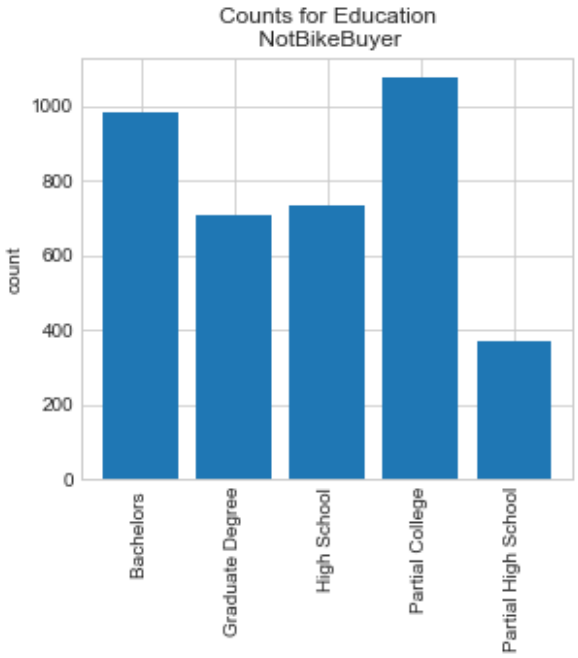
```
train['dummy'] = np.ones(shape = train.shape[0])
cat_col.remove('BikeBuyer')
cat_col.remove('StateProvinceName')
cat_col.remove('City')
for col in cat_col:
    print(col)
    counts = train[['dummy', 'BikeBuyer', col]].groupby(['BikeBuyer', col], as_index
    temp = counts[counts['BikeBuyer'] == 0][[col, 'dummy']]
    _ = plt.figure(figsize = (10,4))
    plt.subplot(1, 2, 1)
    temp = counts[counts['BikeBuyer'] == 0][[col, 'dummy']]
    plt.bar(temp[col], temp.dummy)
    plt.xticks(rotation=90)
    plt.title('Counts for ' + col + '\n NotBikeBuyer')
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    temp = counts[counts['BikeBuyer'] == 1][[col, 'dummy']]
    plt.bar(temp[col], temp.dummy)
    plt.xticks(rotation=90)
    plt.title('Counts for ' + col + '\n BikeBuyer')
    plt.ylabel('count')
    plt.show()

del train['dummy']
```

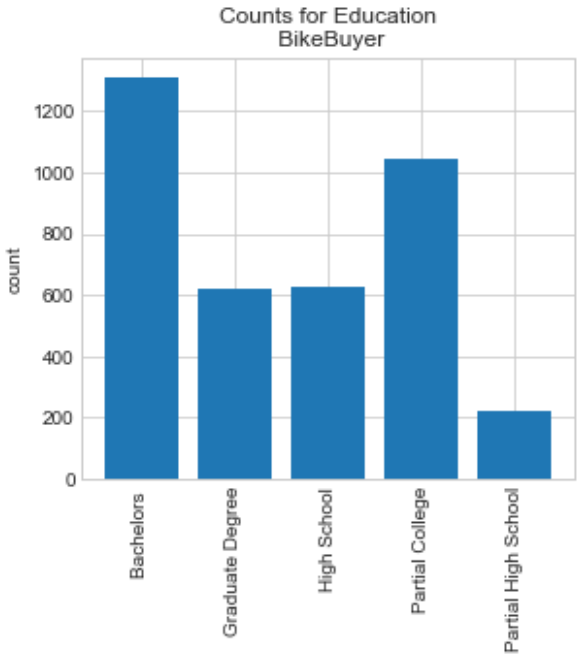
executed in 6.23s, finished 17:03:58 2019-07-10

CountryRegionName

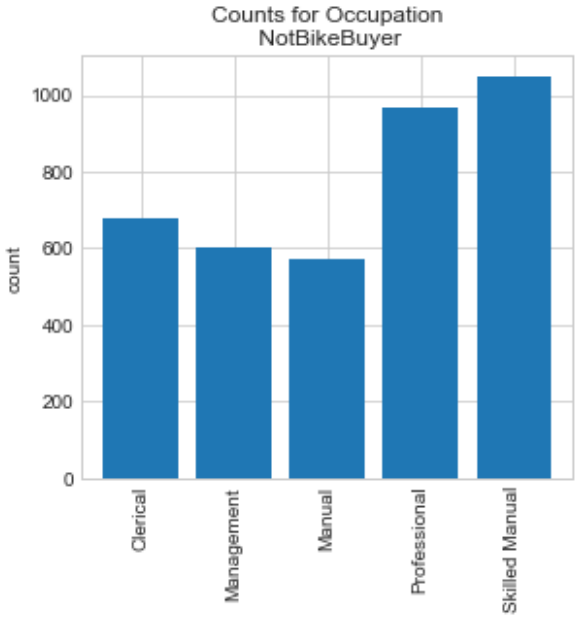


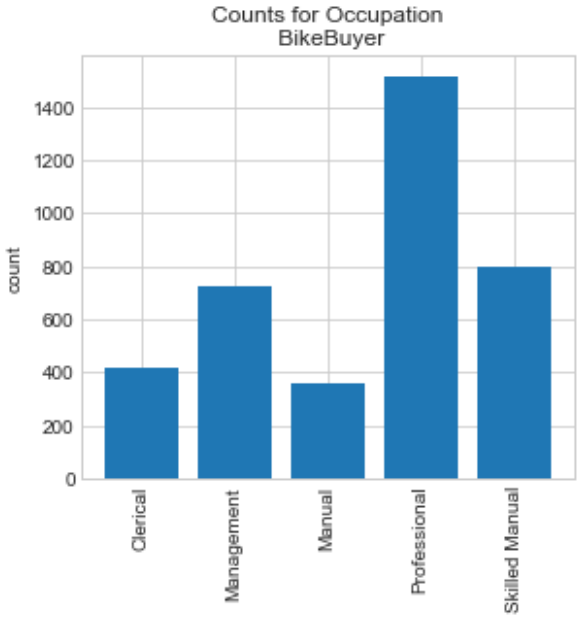
Education



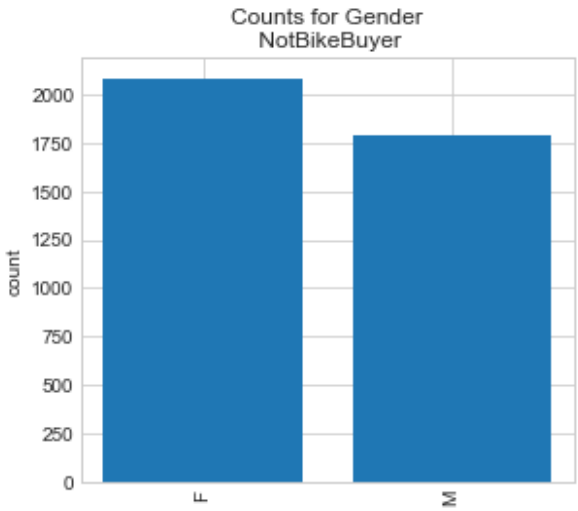


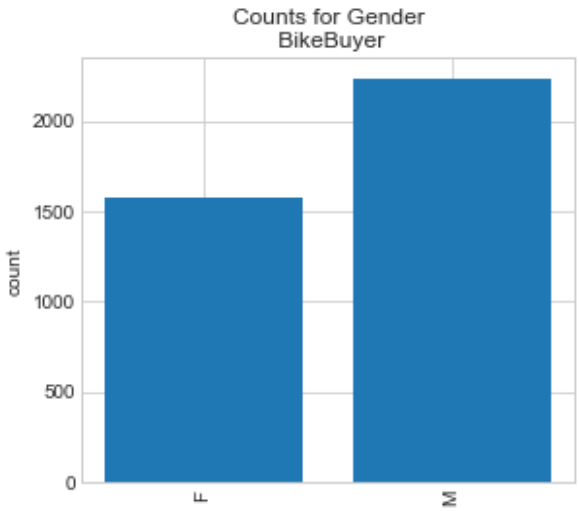
`Occupation`



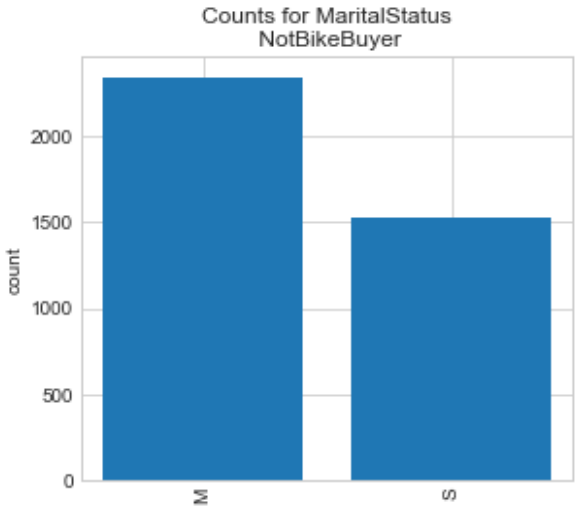


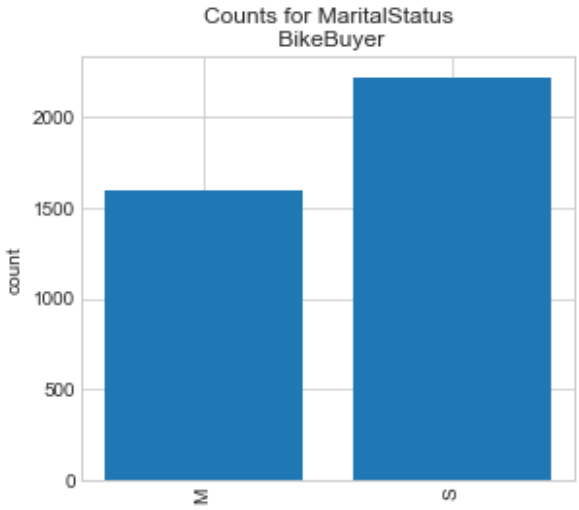
`Gender`



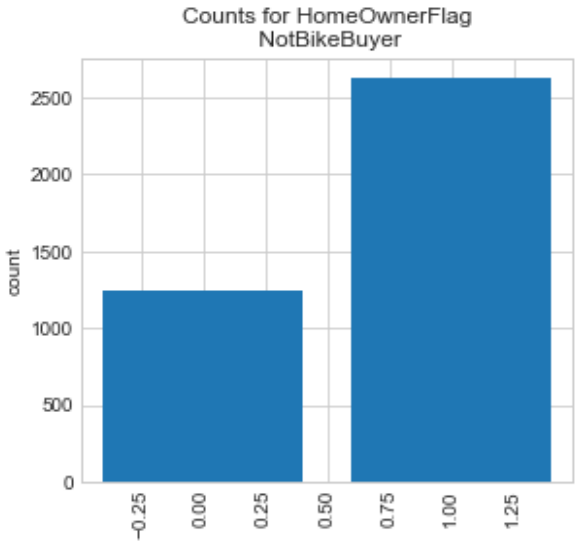


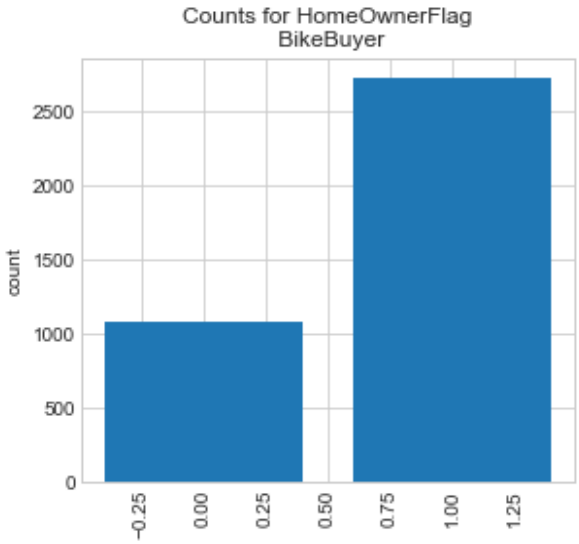
`MaritalStatus`



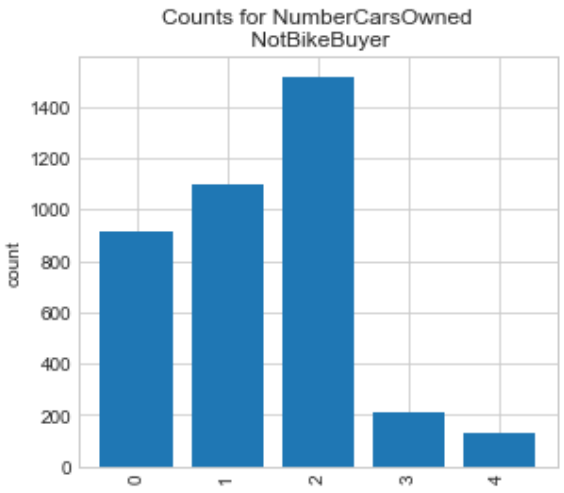


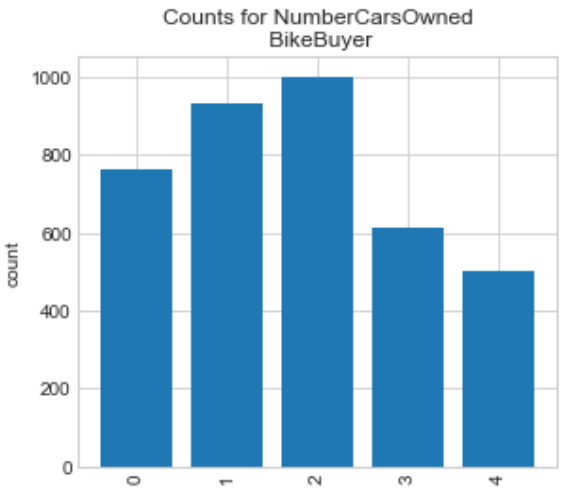
HomeOwnerFlag



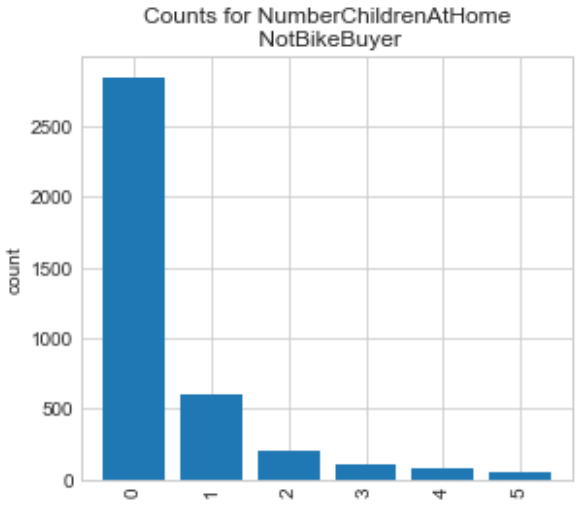


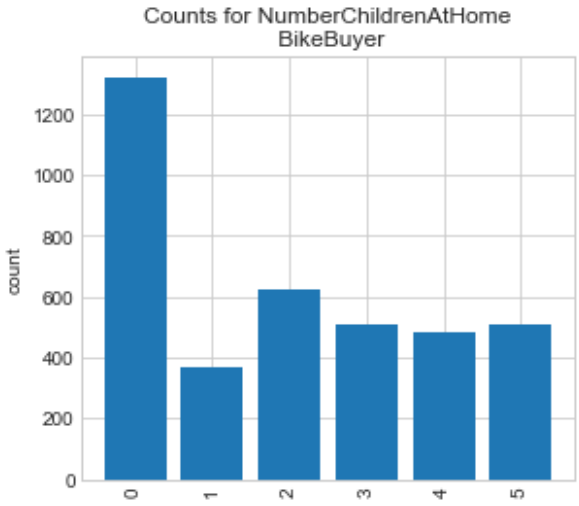
NumberCarsOwned



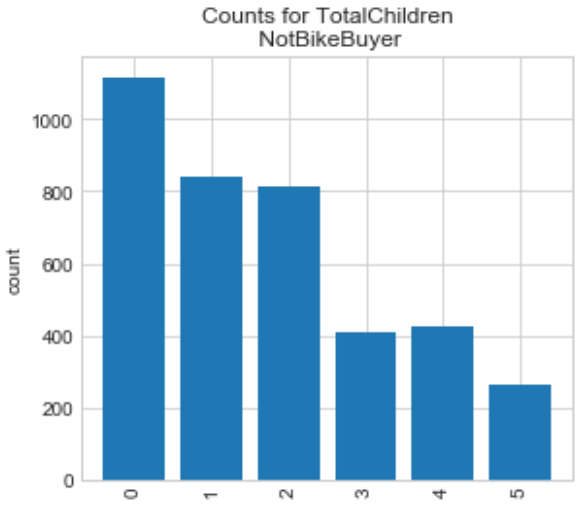


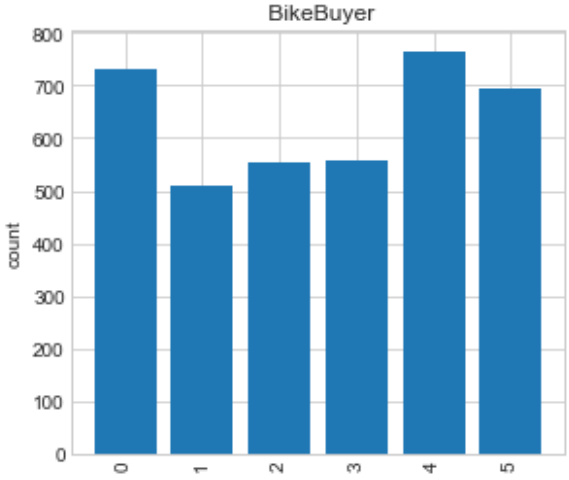
`NumberChildrenAtHome`





`TotalChildren`



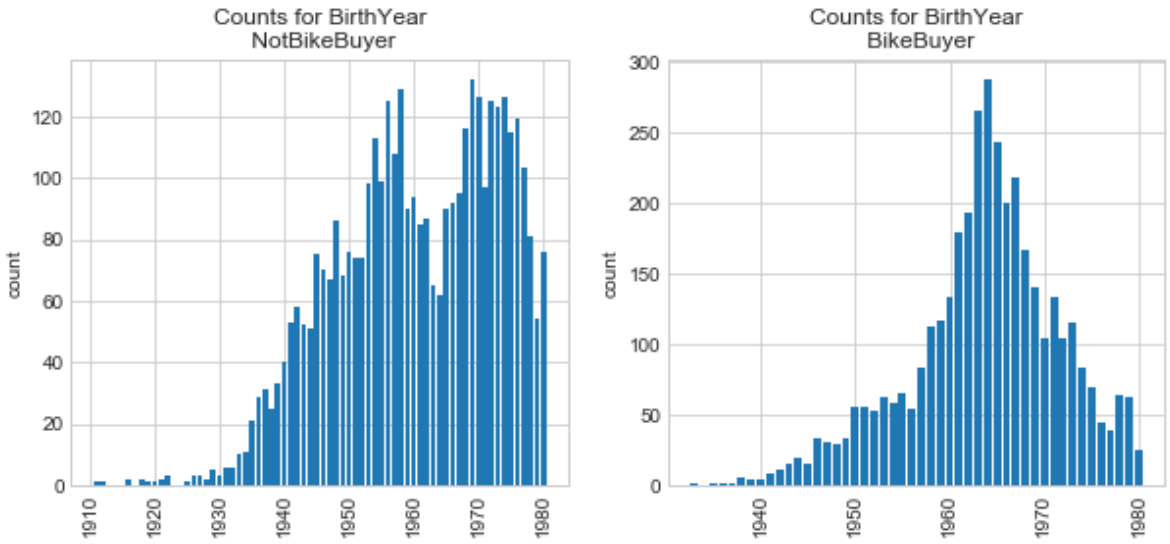


In [21]:

```
train['dummy'] = np.ones(shape = train.shape[0])

for col in ['BirthYear']:
    print(col)
    counts = train[['dummy', 'BikeBuyer', col]].groupby(['BikeBuyer', col], as_index
    temp = counts[counts['BikeBuyer'] == 0][[col, 'dummy']]
    _ = plt.figure(figsize = (10,4))
    plt.subplot(1, 2, 1)
    temp = counts[counts['BikeBuyer'] == 0][[col, 'dummy']]
    plt.bar(temp[col], temp.dummy)
    plt.xticks(rotation=90)
    plt.title('Counts for ' + col + '\n NotBikeBuyer')
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    temp = counts[counts['BikeBuyer'] == 1][[col, 'dummy']]
    plt.bar(temp[col], temp.dummy)
    plt.xticks(rotation=90)
    plt.title('Counts for ' + col + '\n BikeBuyer')
    plt.ylabel('count')
    plt.show()

del train['dummy']
```

executed in 1.39s, finished 17:04:00 2019-07-10

```
BirthYear
```



3 Modelling

In [22]:

```
from catboost import CatBoostClassifier,cv,Pool
```

executed in 3.15s, finished 17:04:04 2019-07-10

In [23]:

```
X=train.drop(['BikeBuyer'],axis=1)
Y=train['BikeBuyer']

Y=Y.ravel()
```

executed in 22ms, finished 17:04:04 2019-07-10

In [24]:

```
split_test_size=0.2

from sklearn.model_selection import train_test_split
Xtrain, Xtest, Ytrain, Ytest= train_test_split(X,Y, test_size=split_test_size, random
```

executed in 10.6s, finished 17:04:15 2019-07-10

In [25]:

```
X.shape
```

executed in 22ms, finished 17:04:16 2019-07-10

Out[25]:

```
(7685, 14)
```

In [28]:

```
X.columns
```

executed in 20ms, finished 17:05:30 2019-07-10

Out[28]:

```
Index(['City', 'StateProvinceName', 'CountryRegionName', 'Education',
       'Occupation', 'Gender', 'MaritalStatus', 'HomeOwnerFlag',
       'NumberCarsOwned', 'NumberChildrenAtHome', 'TotalChildren',
       'YearlyIncome', 'AveMonthSpend', 'BirthYear'],
      dtype='object')
```

In [29]:

```
cb_cat=CatBoostClassifier(iterations=1000,depth=5,eval_metric='Accuracy', cat_feature
                      random_seed=10,learning_rate=.1,verbose=False)

cb_cat.fit(Xtrain,Ytrain,use_best_model=True,eval_set=(Xtest,Ytest),early_stopping_rc
```

executed in 25.5s, finished 17:05:56 2019-07-10

```
0:      learn: 0.7587833        test: 0.7573195 best: 0.7573195 (0)     tota
l: 338ms        remaining: 5m 37s
50:     learn: 0.7802537        test: 0.7807417 best: 0.7807417 (41)    tota
l: 7.47s        remaining: 2m 19s
100:    learn: 0.7849707        test: 0.7794405 best: 0.7813923 (54)    tota
l: 12s  remaining: 1m 46s
150:    learn: 0.7896877        test: 0.7859466 best: 0.7865973 (132)   tota
l: 19.6s        remaining: 1m 49s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 0.7865972674
bestIteration = 132

Shrink model to first 133 iterations.
```

Out[29]:

```
<catboost.core.CatBoostClassifier at 0x14cc2ca3518>
```

In [30]:

```
cb_best=CatBoostClassifier(iterations=133,depth=5,eval_metric='Accuracy', cat_feature
                      random_seed=10,learning_rate=.1,verbose=False)

cb_best.fit(X,Y,verbose=1000)
```

executed in 18.7s, finished 17:06:14 2019-07-10

```
0:      learn: 0.7541965        total: 503ms    remaining: 1m 6s
132:    learn: 0.7915420        total: 17.9s    remaining: 0us
```

Out[30]:

```
<catboost.core.CatBoostClassifier at 0x14cc2ca3198>
```

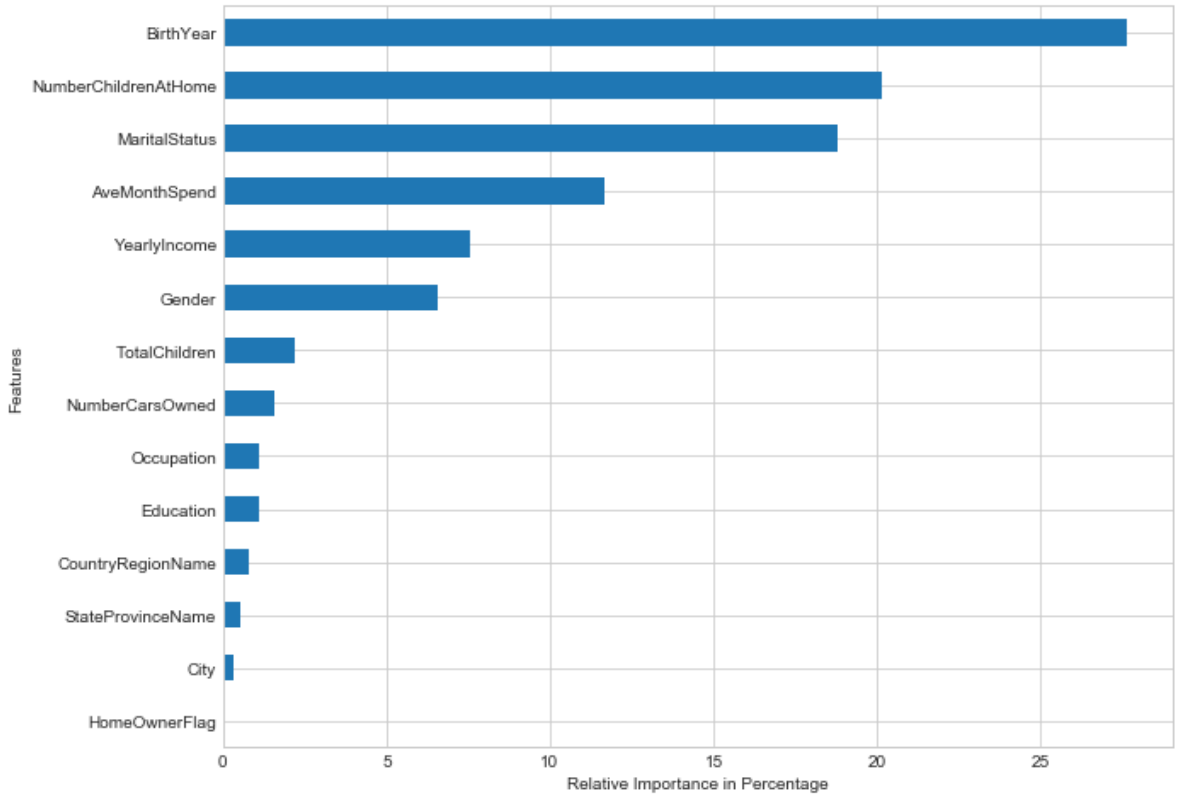
In [31]:

```
b=list(cb_best.feature_importances_[:])
pd.DataFrame(index=X.columns,data=b).sort_values(0).plot.barh(figsize=(10,8),legend=F
plt.ylabel('Features')
plt.xlabel('Relative Importance in Percentage')
```

executed in 875ms, finished 17:06:15 2019-07-10

Out[31]:

```
Text(0.5,0,'Relative Importance in Percentage')
```



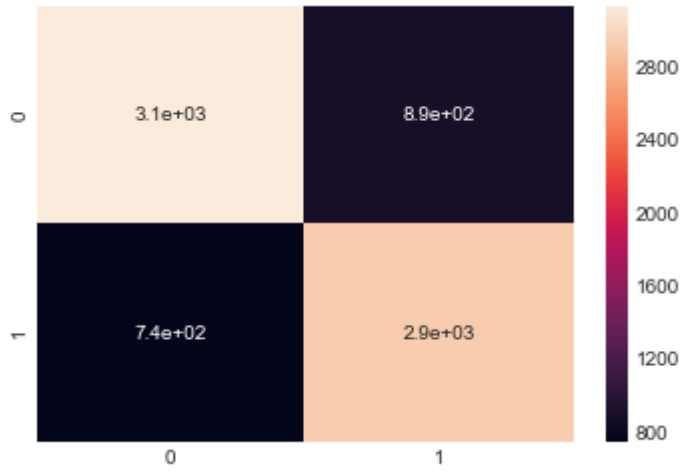
In [32]:

```
from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(cb_best.predict(X),Y),annot=True)
```

executed in 693ms, finished 17:06:16 2019-07-10

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14cc2ca3390>
```



In [33]:

```
1  print(classification_report(cb_best.predict(X),Y))
```

executed in 222ms, finished 17:06:16 2019-07-10

```
              precision    recall  f1-score   support

         0.0       0.81      0.78      0.79      4015
         1.0       0.77      0.80      0.78      3670

   micro avg       0.79      0.79      0.79      7685
   macro avg       0.79      0.79      0.79      7685
weighted avg       0.79      0.79      0.79      7685
```

In [35]:

```
1  mask=(cb_best.feature_importances_[:]>1)
2  (cb_best.feature_importances_[mask]).sum()
```

executed in 20ms, finished 17:06:17 2019-07-10

Out[35]:

```
98.36101204014514
```

In [36]:

```
1  c=Xtrain.columns[mask]
```

executed in 151ms, finished 17:06:17 2019-07-10

In [37]:

```
1  Xtrain_m=Xtrain[c]
2  Xtest_m=Xtest[c]
```

executed in 144ms, finished 17:06:17 2019-07-10

In [38]:

```
1  Xtrain_m.columns
```

executed in 18ms, finished 17:06:17 2019-07-10

Out[38]:

```
Index(['Education', 'Occupation', 'Gender', 'MaritalStatus', 'NumberCarsOwne
d',
       'NumberChildrenAtHome', 'TotalChildren', 'YearlyIncome',
       'AveMonthSpend', 'BirthYear'],
      dtype='object')
```

In [39]:

```
cb=CatBoostClassifier(iterations=1000,depth=5,eval_metric='Accuracy',cat_features=[0,
                      random_seed=10,learning_rate=.1,verbose=False)

cb.fit(Xtrain_m,Ytrain,use_best_model=True,eval_set=(Xtest_m,Ytest),early_stopping_rc
```

executed in 7.96s, finished 17:06:25 2019-07-10

```
0:      learn: 0.7483735        test: 0.7488614 best: 0.7488614 (0)     tota
l: 100ms        remaining: 1m 39s
50:     learn: 0.7716331        test: 0.7755368 best: 0.7794405 (35)    tota
l: 4.47s        remaining: 1m 23s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 0.7794404684
bestIteration = 35

Shrink model to first 36 iterations.
```

Out[39]:

```
<catboost.core.CatBoostClassifier at 0x14cc2d9a080>
```

In [40]:

```
cb_cat=CatBoostClassifier(iterations=1000,depth=5,eval_metric='Accuracy', cat_feature
                      random_seed=10,learning_rate=.1,verbose=False)

cb_cat.fit(Xtrain,Ytrain,use_best_model=True,eval_set=(Xtest,Ytest),early_stopping_rc
```

executed in 28.3s, finished 17:06:53 2019-07-10

```
0:      learn: 0.7275537        test: 0.7247885 best: 0.7247885 (0)     tota
l: 157ms        remaining: 2m 37s
50:     learn: 0.7719584        test: 0.7722837 best: 0.7729343 (42)    tota
l: 5.81s        remaining: 1m 48s
100:    learn: 0.7799284        test: 0.7800911 best: 0.7800911 (89)    tota
l: 11.1s        remaining: 1m 38s
150:    learn: 0.7887118        test: 0.7820429 best: 0.7826936 (146)   tota
l: 17.1s        remaining: 1m 35s
200:    learn: 0.7937541        test: 0.7891997 best: 0.7898504 (188)   tota
l: 23s  remaining: 1m 31s
Stopped by overfitting detector  (50 iterations wait)

bestTest = 0.7898503578
bestIteration = 188

Shrink model to first 189 iterations.
```

Out[40]:

```
<catboost.core.CatBoostClassifier at 0x14cc2d9acf8>
```

In [41]:

```
cb_cat=CatBoostClassifier(iterations=189,depth=5,eval_metric='Accuracy', cat_features
                    random_seed=10,learning_rate=.1,verbose=False)

cb_cat.fit(X,Y,verbose=50)
```

executed in 27.4s, finished 17:07:21 2019-07-10

```
0:      learn: 0.7453481        total: 184ms    remaining: 34.7s
50:     learn: 0.7776187        total: 6.71s    remaining: 18.1s
100:    learn: 0.7847755        total: 12.6s    remaining: 11s
150:    learn: 0.7903709        total: 20.9s    remaining: 5.25s
188:    learn: 0.7931034        total: 26.6s    remaining: 0us
```

Out[41]:

```
<catboost.core.CatBoostClassifier at 0x14cc2d9a048>
```

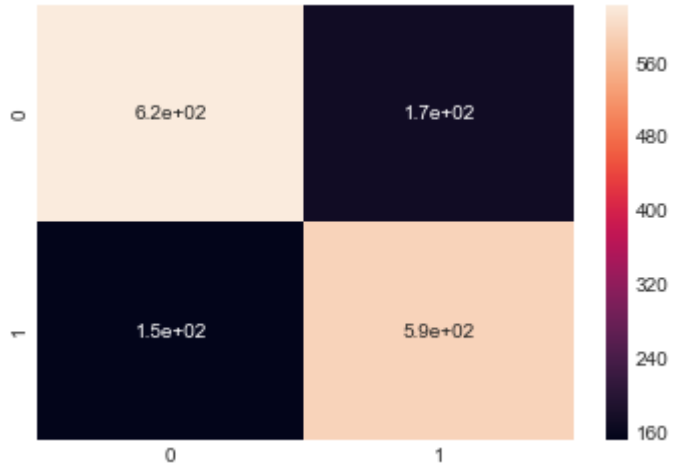
In [42]:

```
from sklearn.metrics import classification_report,confusion_matrix
sns.heatmap(confusion_matrix(cb_cat.predict(Xtest),Ytest),annot=True)
```

executed in 904ms, finished 17:07:22 2019-07-10

Out[42]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14cc2d9aeb8>
```



In [43]:

```
print(classification_report(cb_cat.predict(X),Y))
```

executed in 279ms, finished 17:07:22 2019-07-10

```
              precision    recall  f1-score   support

         0.0       0.81      0.78      0.79      3993
         1.0       0.77      0.80      0.78      3692

   micro avg       0.79      0.79      0.79      7685
   macro avg       0.79      0.79      0.79      7685
weighted avg       0.79      0.79      0.79      7685
```

In [ ]:

```
1
```