Project Overview

Build a predictive model that identifies the factors most likely to result in a consumer disputing a financial institution's response to their complaint.

Background

- In July 2011, the Consumer Financial Protection Bureau (CFPB) was formed in large part due to the factors leading to the Great Recession (Dec 2007 – June 2009)
- Consumers can file a complaint against a financial institution; financial institutions have 15 days to respond to the complaint

Hypothesis

Consumers are more likely to dispute the financial institution's response to their complaint if:

- They are located in CA, TX, or FL
- Complaint focuses on mortgage
- Bank is BofA or Wells Fargo
- Complaint is submitted via web or referral

Data Overview/Cleansing

File obtained from data.gov

- csv file
- 679,879 rows x 18 columns
- All categorical data
- dtypes = object except for one column which was int64 ('complaint id')

- Identify data (columns) needed for analysis; removed all others
- Removed null values
- Selected top ~15 values: Product, Sub-Product, Issue, State (otherwise data frame would have been 4,000+ columns due to dummy variables)
- Changed 'Consumer disputed?' to binary
- Created dummy variables for all factors except target
- RESULT = dataframe with 99 columns x 275K rows

Analysis and Results

- Split data into train and test (80%, 20%)
- Target variable > 'Consumer disputed?'
- 98 columns of dummy variables for factors
- Used Logistic Regression CV

Based on the coefficients for each factor, this model predicts that a consumer is more likely to dispute the financial institution's response if (factors in order):

- Consumer lives in: CA, TX, CO, AZ

- Complaint is submitted via web, fax, or email
- Financial institution is: BofA, JP Morgan, Ocwen, Wells Fargo
- Consumer complaint topic: Mortgage, Home Equity Loan or Credit Line
- Specific issues: Loan servicing/payments/escrow, application/mortgage broker
- Complaint submitted in: October, September, or November

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
df = pd.read_csv('Consumer_Complaints1.csv')
```

```
/Users/krys/anaconda2/lib/python2.7/site-packages/IPython/core/interact
iveshell.py:2717: DtypeWarning: Columns (5,11) have mixed types. Specif
y dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [3]:
```
df.head(3)
```

Out[3]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company
0	07/29/2013	Consumer Loan	Vehicle loan	Managing the loan or lease	NaN	NaN	NaN	Wells Fargo & Company
1	07/29/2013	Bank account or service	Checking account	Using a debit or ATM card	NaN	NaN	NaN	Wells Fargo & Company
2	07/29/2013	Bank account or service	Checking account	Account opening, closing, or management	NaN	NaN	NaN	Santander Bank US

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 679879 entries, 0 to 679878
Data columns (total 18 columns):
Date received                   679879 non-null object
Product                         679879 non-null object
Sub-product                     478762 non-null object
Issue                           679878 non-null object
Sub-issue                       274342 non-null object
Consumer complaint narrative    118315 non-null object
Company public response         150445 non-null object
Company                         679879 non-null object
State                           674511 non-null object
ZIP code                        674498 non-null object
Tags                            96164 non-null object
Consumer consent provided?      215359 non-null object
Submitted via                   679878 non-null object
Date sent to company            679879 non-null object
Company response to consumer    679879 non-null object
Timely response?                679879 non-null object
Consumer disputed?              639285 non-null object
Complaint ID                    679879 non-null int64
dtypes: int64(1), object(17)
memory usage: 93.4+ MB
```

In [5]:
```
# Remove these columns: Sub-issue (4), Consumer Complaint Narrative (5),

#Company Public Response (6), Zip Code (8), Tags (9), Consumer consent p
rovided? (10),
#Date sent to company (12), Complaint ID (17)
```

In [6]:
```
df.drop(df.columns[[4,5,6]], axis=1, inplace=True)
```

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 679879 entries, 0 to 679878
Data columns (total 15 columns):
Date received                   679879 non-null object
Product                         679879 non-null object
Sub-product                     478762 non-null object
Issue                           679878 non-null object
Company                         679879 non-null object
State                           674511 non-null object
ZIP code                        674498 non-null object
Tags                            96164 non-null object
Consumer consent provided?      215359 non-null object
Submitted via                   679878 non-null object
Date sent to company            679879 non-null object
Company response to consumer    679879 non-null object
Timely response?                679879 non-null object
Consumer disputed?              639285 non-null object
Complaint ID                    679879 non-null int64
dtypes: int64(1), object(14)
memory usage: 77.8+ MB
```

```
In [8]: df.drop(df.columns[[6,7,8,10,12,14]], axis=1, inplace=True)
```

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 679879 entries, 0 to 679878
Data columns (total 9 columns):
Date received                   679879 non-null object
Product                         679879 non-null object
Sub-product                     478762 non-null object
Issue                           679878 non-null object
Company                         679879 non-null object
State                           674511 non-null object
Submitted via                   679878 non-null object
Company response to consumer    679879 non-null object
Consumer disputed?              639285 non-null object
dtypes: object(9)
memory usage: 46.7+ MB
```

```
In [10]: df['Product'].value_counts()
```

```
Out[10]: Mortgage                    212178
         Debt collection             126369
         Credit reporting            120998
         Credit card                  80119
         Bank account or service      77253
         Consumer Loan                27101
         Student loan                 22083
         Payday loan                   4893
         Money transfers               4792
         Prepaid card                  3242
         Other financial service        836
         Virtual currency                15
         Name: Product, dtype: int64
```

```
In [11]: threshold = 1000
         for col in df.columns:
             value_counts = df['Product'].value_counts()
             to_remove = value_counts[value_counts <= threshold].index
             df['Product'].replace(to_remove, inplace=True)
         print df['Product'].value_counts()
```

```
Mortgage                    212358
Debt collection             126586
Credit reporting            121152
Credit card                  80223
Bank account or service      77350
Consumer Loan                27143
Student loan                 22117
Payday loan                   4897
Money transfers               4802
Prepaid card                  3251
Name: Product, dtype: int64
```

In [12]:
```
df['Sub-product'].value_counts()
```

Out[12]:
```
Other mortgage                              82264
Conventional fixed mortgage                 65690
Checking account                            53724
Other (i.e. phone, health club, etc.)       37853
I do not know                               26429
Credit card                                 25464
Conventional adjustable mortgage (ARM)      23806
FHA mortgage                                22582
Non-federal student loan                    20938
Medical                                     17504
Vehicle loan                                15386
Other bank product/service                  15016
Payday loan                                 11721
Home equity loan or line of credit          10674
Installment loan                             7241
Savings account                              4750
VA mortgage                                  4608
Mortgage                                     4359
Federal student loan servicing               3716
Auto                                         3193
(CD) Certificate of deposit                  3191
International money transfer                 2508
Domestic (US) money transfer                 2299
Vehicle lease                                2221
Federal student loan                         2168
Reverse mortgage                             1892
Personal line of credit                      1717
General purpose card                         1489
Second mortgage                               662
Cashing a check without an account            572
Title loan                                    454
Payroll card                                  374
Mobile wallet                                 345
Gift or merchant card                         331
Government benefit payment card               313
Debt settlement                               238
Check cashing                                 212
ID prepaid card                               188
Other special purpose card                    163
Money order                                   118
Pawn loan                                      82
Credit repair                                  81
Traveler’s/Cashier’s checks                    72
Refund anticipation check                      60
Foreign currency exchange                      55
Transit card                                   33
Electronic Benefit Transfer / EBT card          6
Name: Sub-product, dtype: int64
```

In [13]:
```
threshold = 4500
for col in df.columns:
    value_counts = df['Sub-product'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['Sub-product'].replace(to_remove, inplace=True)
print df['Sub-product'].value_counts()
```

```
Other mortgage                              119633
Conventional fixed mortgage                  99733
Checking account                             80711
Other (i.e. phone, health club, etc.)        60339
I do not know                                41208
Credit card                                  39579
Conventional adjustable mortgage (ARM)       36262
FHA mortgage                                 34552
Non-federal student loan                     31949
Medical                                      28453
Vehicle loan                                 23804
Other bank product/service                   23387
Payday loan                                  18079
Home equity loan or line of credit           16298
Installment loan                             11539
VA mortgage                                   7212
Savings account                               7141
Name: Sub-product, dtype: int64
```

In [14]:
```
df['Issue'].value_counts()
```

```
Out[14]: Loan modification,collection,foreclosure    107093
         Incorrect information on credit report       88243
         Loan servicing, payments, escrow account     70979
         Cont'd attempts collect debt not owed        52502
         Account opening, closing, or management      33832
         Disclosure verification of debt              25173
         Communication tactics                        21621
         Deposits and withdrawals                     20618
         Application, originator, mortgage broker     15702
         Credit reporting company's investigation     14178
         Billing disputes                             13374
         Other                                        13262
         Managing the loan or lease                   12973
         Problems caused by my funds being low        10785
         Dealing with my lender or servicer           10546
         False statements or representation           10174
         Unable to get credit report/credit score      9870
         Improper contact or sharing of info           8938
         Problems when you are unable to pay           8281
         Settlement process and costs                  8182
         Taking/threatening an illegal action          7961
         Identity theft / Fraud / Embezzlement         7493
         Making/receiving payments, sending money      6580
         Closing/Cancelling account                    5730
         Using a debit or ATM card                     5438
         Can't repay my loan                           5418
         Credit decision / Underwriting                5188
         APR or interest rate                          5183
         Improper use of my credit report              4799
         Credit monitoring or identity protection      3908
                                                      ...
         Privacy                                        443
         Bankruptcy                                     426
         Payment to acct not credited                   390
         Applied for loan/did not receive money         321
         Arbitration                                    311
         Sale of account                                311
         Shopping for a line of credit                  287
         Charged bank acct wrong day or amt             260
         Wrong amount charged or received               244
         Cash advance                                   232
         Customer service/Customer relations            231
         Fees                                           205
         Overlimit fee                                  202
         Balance transfer fee                           196
         Adding money                                   185
         Cash advance fee                               184
         Incorrect/missing disclosures or info          181
         Convenience checks                             132
         Excessive fees                                  85
         Unexpected/Other fees                           84
         Lender repossessed or sold the vehicle          68
         Advertising, marketing or disclosures           66
         Overdraft, savings or rewards features          46
         Lost or stolen check                            37
         Lost or stolen money order                      36
         Disclosures                                     35
```

```
Incorrect exchange rate                          22
Lender sold the property                          7
Lender damaged or destroyed vehicle               6
Lender damaged or destroyed property              1
Name: Issue, dtype: int64
```

In [15]:

```
threshold = 10000
for col in df.columns:
    value_counts = df['Issue'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['Issue'].replace(to_remove, inplace=True)
print df['Issue'].value_counts()
```

```
Loan modification,collection,foreclosure    139974
Incorrect information on credit report      114181
Loan servicing, payments, escrow account     93172
Cont'd attempts collect debt not owed        68129
Account opening, closing, or management      44140
Disclosure verification of debt              32598
Communication tactics                        28170
Deposits and withdrawals                     27000
Application, originator, mortgage broker     20801
Credit reporting company's investigation     18529
Billing disputes                             17587
Other                                        17476
Managing the loan or lease                   17107
Problems caused by my funds being low        14067
Dealing with my lender or servicer           13682
False statements or representation           13266
Name: Issue, dtype: int64
```

In [16]:
```
df['State'].value_counts()
```

```
Out[16]: CA    99006
         FL    65497
         TX    51656
         NY    46939
         GA    31050
         NJ    27278
         IL    24570
         PA    24448
         VA    22013
         MD    21459
         OH    20940
         NC    19035
         MI    17168
         AZ    15281
         WA    14140
         MA    13284
         CO    11761
         TN    10436
         MO     8974
         SC     8606
         NV     8401
         OR     8076
         CT     7907
         MN     7855
         IN     7520
         WI     7319
         AL     7061
         LA     6821
         KY     4841
         OK     4532
               ...
         DE     3484
         NM     3391
         KS     3286
         NH     3262
         MS     3185
         AR     2915
         IA     2896
         ID     2341
         HI     2321
         ME     2248
         RI     2187
         NE     2101
         WV     1760
         PR     1603
         MT     1103
         VT     1101
         SD      926
         AK      790
         WY      673
         ND      580
         AE      279
         AP      184
         VI      170
         GU       90
         FM       41
         MH       28
```

```
MP       25
AS       21
AA       13
PW       11
Name: State, dtype: int64
```

In [17]:
```
threshold = 10000
for col in df.columns:
    value_counts = df['State'].value_counts()
    to_remove = value_counts[value_counts <= threshold].index
    df['State'].replace(to_remove, inplace=True)
print df['State'].value_counts()
```

```
CA    125638
FL     83251
TX     65341
NY     59612
GA     39427
NJ     34645
IL     31057
PA     30780
VA     27782
MD     27310
OH     26720
NC     24096
MI     21786
AZ     19431
WA     17951
MA     16837
CO     14964
TN     13251
Name: State, dtype: int64
```

In [18]:
```
#df1 = df.dropna(subset=[['Sub-product']])
```

In [19]:
```
df1 = df.dropna(subset=[['Consumer disputed?']]).copy()
```

In [20]:
```
df1.shape
```

Out[20]:
```
(639285, 9)
```

In [21]:
```
df1['Consumer disputed?'].value_counts()
```

Out[21]:
```
No     504944
Yes    134341
Name: Consumer disputed?, dtype: int64
```

In [22]:
```
df1['Consumer disputed?'] = df1['Consumer disputed?'].map({'No':0,
'Yes':1})
# df1['Consumer disputed?'].map({'No':0, 'Yes':1}).head()
df1['Consumer disputed?'].head()
```

Out[22]:
```
0    0
1    0
2    0
3    0
4    0
Name: Consumer disputed?, dtype: int64
```

In [23]:
```
df1['Consumer disputed?'].shape
```

Out[23]:
```
(639285,)
```

In [24]:
```
dummy_product = pd.get_dummies(df1[['Product']], prefix='Product')
print dummy_product.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 639285 entries, 0 to 679878
Data columns (total 10 columns):
Product_Bank account or service    639285 non-null float64
Product_Consumer Loan              639285 non-null float64
Product_Credit card                639285 non-null float64
Product_Credit reporting           639285 non-null float64
Product_Debt collection            639285 non-null float64
Product_Money transfers            639285 non-null float64
Product_Mortgage                   639285 non-null float64
Product_Payday loan                639285 non-null float64
Product_Prepaid card               639285 non-null float64
Product_Student loan               639285 non-null float64
dtypes: float64(10)
memory usage: 53.7 MB
None
```

In [25]:
```
dummy_product = pd.get_dummies(df1[['Product']], prefix='Product')
print dummy_product.head()
```

```
   Product_Bank account or service  Product_Consumer Loan  \
0                              0.0                    1.0   
1                              1.0                    0.0   
2                              1.0                    0.0   
3                              1.0                    0.0   
4                              0.0                    0.0   

   Product_Credit card  Product_Credit reporting  Product_Debt collecti
on  \
0                  0.0                       0.0                      
 0.0   
1                  0.0                       0.0                      
 0.0   
2                  0.0                       0.0                      
 0.0   
3                  0.0                       0.0                      
 0.0   
4                  0.0                       0.0                      
 0.0   

   Product_Money transfers  Product_Mortgage  Product_Payday loan  \
0                      0.0               0.0                  0.0   
1                      0.0               0.0                  0.0   
2                      0.0               0.0                  0.0   
3                      0.0               0.0                  0.0   
4                      0.0               1.0                  0.0   

   Product_Prepaid card  Product_Student loan  
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  
```

In [26]:
```
dummy_product.isnull().values.any()
```

Out[26]:
```
False
```

In [27]:
```
dummy_subproduct = pd.get_dummies(df1[['Sub-product']], prefix='Sub-pro
d')
print dummy_subproduct.head()
```

```
   Sub-prod_Checking account  Sub-prod_Conventional adjustable mortgage
(ARM)  \
0                        0.0
   0.0
1                        1.0
   0.0
2                        1.0
   0.0
3                        1.0
   0.0
4                        0.0
   0.0

   Sub-prod_Conventional fixed mortgage  Sub-prod_Credit card  \
0                                   0.0                   0.0
1                                   0.0                   0.0
2                                   0.0                   0.0
3                                   0.0                   0.0
4                                   1.0                   0.0

   Sub-prod_FHA mortgage  Sub-prod_Home equity loan or line of credit
 \
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


   Sub-prod_I do not know  Sub-prod_Installment loan  Sub-prod_Medical
\
0                     0.0                        0.0               0.0

1                     0.0                        0.0               0.0

2                     0.0                        0.0               0.0

3                     0.0                        0.0               0.0

4                     0.0                        0.0               0.0


   Sub-prod_Non-federal student loan  \
0                                0.0
1                                0.0
2                                0.0
3                                0.0
4                                0.0

   Sub-prod_Other (i.e. phone, health club, etc.)  \
0                                             0.0
1                                             0.0
2                                             0.0
```

```
3                                              0.0
4                                              0.0

   Sub-prod_Other bank product/service  Sub-prod_Other mortgage  \
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

   Sub-prod_Payday loan  Sub-prod_Savings account  Sub-prod_VA mortgage
  \
0                   0.0                       0.0                   0.0

1                   0.0                       0.0                   0.0

2                   0.0                       0.0                   0.0

3                   0.0                       0.0                   0.0

4                   0.0                       0.0                   0.0


   Sub-prod_Vehicle loan
0                    1.0
1                    0.0
2                    0.0
3                    0.0
4                    0.0
```

In [28]:
```
dummy_subproduct.isnull().values.any()
```

Out[28]:
```
False
```

In [29]:
```
dummy_issue = pd.get_dummies(df1[['Issue']], prefix='Issue')
print dummy_issue.head()
```

```
   Issue_Account opening, closing, or management  \
0                                            0.0
1                                            0.0
2                                            1.0
3                                            0.0
4                                            0.0

   Issue_Application, originator, mortgage broker  Issue_Billing disput
es  \
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

   Issue_Communication tactics  Issue_Cont'd attempts collect debt not
 owed  \
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

   Issue_Credit reporting company's investigation  \
0                                             0.0
1                                             0.0
2                                             0.0
3                                             0.0
4                                             0.0

   Issue_Dealing with my lender or servicer  Issue_Deposits and withdra
wals  \
0                                       0.0
0.0
1                                       0.0
0.0
2                                       0.0
0.0
3                                       0.0
1.0
4                                       0.0
0.0

   Issue_Disclosure verification of debt  \
0                                    0.0
1                                    0.0
2                                    0.0
```

```
3                                     0.0
4                                     0.0

   Issue_False statements or representation  \
0                                        0.0
1                                        0.0
2                                        0.0
3                                        0.0
4                                        0.0

   Issue_Incorrect information on credit report  \
0                                            0.0
1                                            0.0
2                                            0.0
3                                            0.0
4                                            0.0

   Issue_Loan modification,collection,foreclosure  \
0                                              0.0
1                                              0.0
2                                              0.0
3                                              0.0
4                                              0.0

   Issue_Loan servicing, payments, escrow account  \
0                                              0.0
1                                              0.0
2                                              0.0
3                                              0.0
4                                              1.0

   Issue_Managing the loan or lease  Issue_Other  \
0                               1.0          0.0
1                               1.0          0.0
2                               0.0          0.0
3                               0.0          0.0
4                               0.0          0.0

   Issue_Problems caused by my funds being low
0                                          0.0
1                                          0.0
2                                          0.0
3                                          0.0
4                                          0.0
```

In [30]:
```
dummy_comp_response = pd.get_dummies(df1[['Company response to
consumer']],
                                     prefix='Company_respone')
print dummy_comp_response.head()
```

```
   Company_respone_Closed  Company_respone_Closed with explanation  \
0                     0.0                                      1.0
1                     0.0                                      1.0
2                     1.0                                      0.0
3                     0.0                                      1.0
4                     0.0                                      1.0

   Company_respone_Closed with monetary relief  \
0                                          0.0
1                                          0.0
2                                          0.0
3                                          0.0
4                                          0.0

   Company_respone_Closed with non-monetary relief  \
0                                              0.0
1                                              0.0
2                                              0.0
3                                              0.0
4                                              0.0

   Company_respone_Closed with relief  Company_respone_Closed without r
elief  \
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

   Company_respone_Untimely response
0                                0.0
1                                0.0
2                                0.0
3                                0.0
4                                0.0
```

In [31]:
```
dummy_submittedvia = pd.get_dummies(df1[['Submitted via']], prefix='Subm
itVia')
print dummy_submittedvia.head()
```

```
   SubmitVia_Email  SubmitVia_Fax  SubmitVia_Phone  SubmitVia_Postal ma
il  \
0              0.0            0.0              1.0                    
 0.0
1              0.0            0.0              0.0                    
 0.0
2              0.0            1.0              0.0                    
 0.0
3              0.0            0.0              0.0                    
 0.0
4              0.0            0.0              0.0                    
 0.0

   SubmitVia_Referral  SubmitVia_Web
0                 0.0            0.0
1                 0.0            1.0
2                 0.0            0.0
3                 0.0            1.0
4                 0.0            1.0
```

In [32]:
```
dummy_state = pd.get_dummies(df1[['State']], prefix='State')
print dummy_state.head()
```

```
   State_AZ  State_CA  State_CO  State_FL  State_GA  State_IL  State_MA
  \
0       0.0       0.0       0.0       0.0       0.0       0.0       0.0

1       0.0       1.0       0.0       0.0       0.0       0.0       0.0

2       0.0       0.0       0.0       0.0       0.0       0.0       0.0

3       0.0       0.0       0.0       0.0       1.0       0.0       0.0

4       0.0       0.0       0.0       0.0       1.0       0.0       0.0


   State_MD  State_MI  State_NC  State_NJ  State_NY  State_OH  State_PA
  \
0       0.0       0.0       0.0       0.0       0.0       0.0       0.0

1       0.0       0.0       0.0       0.0       0.0       0.0       0.0

2       0.0       0.0       0.0       0.0       1.0       0.0       0.0

3       0.0       0.0       0.0       0.0       0.0       0.0       0.0

4       0.0       0.0       0.0       0.0       0.0       0.0       0.0


   State_TN  State_TX  State_VA  State_WA
0       0.0       0.0       1.0       0.0
1       0.0       0.0       0.0       0.0
2       0.0       0.0       0.0       0.0
3       0.0       0.0       0.0       0.0
4       0.0       0.0       0.0       0.0
```

In [33]:
```
df1['Company'].value_counts().shape
```

Out[33]:
```
(3573,)
```

In [34]:
```
df1['Company'].value_counts().head(12)
```

Out[34]:
```
Bank of America                             59933
Wells Fargo & Company                       46699
Equifax                                     39645
JPMorgan Chase & Co.                        37973
Experian                                    36715
TransUnion Intermediate Holdings, Inc.      31106
Citibank                                    30171
Ocwen                                       22417
Capital One                                 17571
Nationstar Mortgage                         14596
Synchrony Financial                         11039
U.S. Bancorp                                10803
Name: Company, dtype: int64
```

In [35]:
```
banks = ['Bank of America',
           'Wells Fargo & Company',
           'JPMorgan Chase & Co.',
           'Ocwen',
           'Citibank',
           'Nationstar Mortgage',
           'Ditech Financial LLC',
           'Navient Solutions, Inc.',
           'U.S. Bancorp',
           'PNC Bank N.A.',
           'Encore Capital Group',
           'Capital One']

df2 = df1[df1.Company.isin(banks)]
print df2['Company'].value_counts()
```

```
Bank of America            59933
Wells Fargo & Company      46699
JPMorgan Chase & Co.       37973
Citibank                   30171
Ocwen                      22417
Capital One                17571
Nationstar Mortgage        14596
U.S. Bancorp               10803
Ditech Financial LLC       10069
Navient Solutions, Inc.     9928
PNC Bank N.A.               7556
Encore Capital Group        6929
Name: Company, dtype: int64
```

In [36]:
```
df2.shape
```

Out[36]:
```
(274645, 9)
```

```
In [37]: dummy_company = pd.get_dummies(df2[['Company']], prefix='Company')
         print dummy_company.head()
```

```
   Company_Bank of America  Company_Capital One  Company_Citibank  \
0                      0.0                  0.0               0.0
1                      0.0                  0.0               0.0
3                      0.0                  0.0               0.0
5                      1.0                  0.0               0.0
8                      0.0                  0.0               1.0

   Company_Ditech Financial LLC  Company_Encore Capital Group  \
0                           0.0                           0.0
1                           0.0                           0.0
3                           0.0                           0.0
5                           0.0                           0.0
8                           0.0                           0.0

   Company_JPMorgan Chase & Co.  Company_Nationstar Mortgage  \
0                           0.0                          0.0
1                           0.0                          0.0
3                           0.0                          0.0
5                           0.0                          0.0
8                           0.0                          0.0

   Company_Navient Solutions, Inc.  Company_Ocwen  Company_PNC Bank N.
A.  \
0                              0.0            0.0                    0.
0
1                              0.0            0.0                    0.
0
3                              0.0            0.0                    0.
0
5                              0.0            0.0                    0.
0
8                              0.0            0.0                    0.
0

   Company_U.S. Bancorp  Company_Wells Fargo & Company
0                   0.0                            1.0
1                   0.0                            1.0
3                   0.0                            1.0
5                   0.0                            0.0
8                   0.0                            0.0
```

In [38]:
```
df2.head(3)
```

Out[38]:

	Date received	Product	Sub-product	Issue	Company	State	Submitted via	Company response to consumer
0	07/29/2013	Consumer Loan	Vehicle loan	Managing the loan or lease	Wells Fargo & Company	VA	Phone	Closed with explanation
1	07/29/2013	Bank account or service	Checking account	Managing the loan or lease	Wells Fargo & Company	CA	Web	Closed with explanation
3	07/29/2013	Bank account or service	Checking account	Deposits and withdrawals	Wells Fargo & Company	GA	Web	Closed with explanation

In [80]:
```
from datetime import datetime
from dateutil.parser import parse
df2copy = df2.copy()
df2copy['Date received'] = pd.to_datetime(df2copy['Date received'], form
at='%m/%d/%Y')

print df2copy['Date received'].head()
```

```
0   2013-07-29
1   2013-07-29
3   2013-07-29
5   2013-07-29
8   2013-07-29
Name: Date received, dtype: datetime64[ns]
```

In [81]:
```
df2copy['Date received'] = df2copy['Date received'].dt.month
```

In [82]:
```
df2copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 274645 entries, 0 to 679867
Data columns (total 9 columns):
Date received                   274645 non-null int64
Product                         274645 non-null object
Sub-product                     274645 non-null object
Issue                           274645 non-null object
Company                         274645 non-null object
State                           274645 non-null object
Submitted via                   274645 non-null object
Company response to consumer    274645 non-null object
Consumer disputed?              274645 non-null int64
dtypes: int64(2), object(7)
memory usage: 21.0+ MB
```

```
In [42]: df2copy.head()
```

Out[42]:

	Date received	Product	Sub-product	Issue	Company	State	Submitted via	Company response to consumer	Cc di
0	7	Consumer Loan	Vehicle loan	Managing the loan or lease	Wells Fargo & Company	VA	Phone	Closed with explanation	0
1	7	Bank account or service	Checking account	Managing the loan or lease	Wells Fargo & Company	CA	Web	Closed with explanation	0
3	7	Bank account or service	Checking account	Deposits and withdrawals	Wells Fargo & Company	GA	Web	Closed with explanation	0
5	7	Bank account or service	Checking account	Deposits and withdrawals	Bank of America	TX	Web	Closed with explanation	0
8	7	Credit card	I do not know	Cont'd attempts collect debt not owed	Citibank	OH	Referral	Closed with explanation	1

```
In [43]: dummy_daterecd = pd.get_dummies(df2copy['Date received'], prefix='Date_R
         ecd')
         print dummy_daterecd.head()
```

```
   Date_Recd_1  Date_Recd_2  Date_Recd_3  Date_Recd_4  Date_Recd_5  \
0          0.0          0.0          0.0          0.0          0.0   
1          0.0          0.0          0.0          0.0          0.0   
3          0.0          0.0          0.0          0.0          0.0   
5          0.0          0.0          0.0          0.0          0.0   
8          0.0          0.0          0.0          0.0          0.0   

   Date_Recd_6  Date_Recd_7  Date_Recd_8  Date_Recd_9  Date_Recd_10  \
0          0.0          1.0          0.0          0.0           0.0   
1          0.0          1.0          0.0          0.0           0.0   
3          0.0          1.0          0.0          0.0           0.0   
5          0.0          1.0          0.0          0.0           0.0   
8          0.0          1.0          0.0          0.0           0.0   

   Date_Recd_11  Date_Recd_12  
0           0.0           0.0  
1           0.0           0.0  
3           0.0           0.0  
5           0.0           0.0  
8           0.0           0.0  
```

```
In [44]: df1['Consumer disputed?'].shape

Out[44]: (639285,)
```

```
In [45]: dummy_product.shape

Out[45]: (639285, 10)
```

```
In [46]: dummy_subproduct.shape

Out[46]: (639285, 17)
```

In [47]:
```
data1 = pd.concat([dummy_product, dummy_subproduct], axis=1, join='inne
r')
print data1.head()
```

```
   Product_Bank account or service  Product_Consumer Loan  \
0                              0.0                    1.0
1                              1.0                    0.0
2                              1.0                    0.0
3                              1.0                    0.0
4                              0.0                    0.0

   Product_Credit card  Product_Credit reporting  Product_Debt collecti
on  \
0                  0.0                       0.0
 0.0
1                  0.0                       0.0
 0.0
2                  0.0                       0.0
 0.0
3                  0.0                       0.0
 0.0
4                  0.0                       0.0
 0.0

   Product_Money transfers  Product_Mortgage  Product_Payday loan  \
0                      0.0               0.0                  0.0
1                      0.0               0.0                  0.0
2                      0.0               0.0                  0.0
3                      0.0               0.0                  0.0
4                      0.0               1.0                  0.0

   Product_Prepaid card  Product_Student loan          ...              \
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          ...

   Sub-prod_Installment loan  Sub-prod_Medical  \
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

   Sub-prod_Non-federal student loan  \
0                                0.0
1                                0.0
2                                0.0
3                                0.0
4                                0.0

   Sub-prod_Other (i.e. phone, health club, etc.)  \
0                                             0.0
1                                             0.0
2                                             0.0
3                                             0.0
4                                             0.0

   Sub-prod_Other bank product/service  Sub-prod_Other mortgage  \
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

   Sub-prod_Payday loan  Sub-prod_Savings account  Sub-prod_VA mortgage
  \
0                   0.0                       0.0                   0.0

1                   0.0                       0.0                   0.0

2                   0.0                       0.0                   0.0

3                   0.0                       0.0                   0.0

4                   0.0                       0.0                   0.0


   Sub-prod_Vehicle loan
0                    1.0
1                    0.0
2                    0.0
3                    0.0
4                    0.0

[5 rows x 27 columns]
```

In [48]:
```
data = pd.concat([dummy_product,
                  dummy_subproduct,
                  dummy_issue,
                  dummy_comp_response,
                  dummy_submittedvia,
                 dummy_state,
                 dummy_company,
                 dummy_daterecd,
                  df1[['Consumer disputed?']]],
                 axis=1, join='inner')

print data.head()
```

```
   Product_Bank account or service  Product_Consumer Loan  \
0                              0.0                    1.0
1                              1.0                    0.0
3                              1.0                    0.0
5                              1.0                    0.0
8                              0.0                    0.0

   Product_Credit card  Product_Credit reporting  Product_Debt collecti
on  \
0                  0.0                       0.0
 0.0
1                  0.0                       0.0
 0.0
3                  0.0                       0.0
 0.0
5                  0.0                       0.0
 0.0
8                  1.0                       0.0
 0.0

   Product_Money transfers  Product_Mortgage  Product_Payday loan  \
0                      0.0               0.0                  0.0
1                      0.0               0.0                  0.0
3                      0.0               0.0                  0.0
5                      0.0               0.0                  0.0
8                      0.0               0.0                  0.0

   Product_Prepaid card  Product_Student loan         ...          \
0                   0.0                   0.0         ...
1                   0.0                   0.0         ...
3                   0.0                   0.0         ...
5                   0.0                   0.0         ...
8                   0.0                   0.0         ...

   Date_Recd_4  Date_Recd_5  Date_Recd_6  Date_Recd_7  Date_Recd_8  \
0          0.0          0.0          0.0          1.0          0.0
1          0.0          0.0          0.0          1.0          0.0
3          0.0          0.0          0.0          1.0          0.0
5          0.0          0.0          0.0          1.0          0.0
8          0.0          0.0          0.0          1.0          0.0

   Date_Recd_9  Date_Recd_10  Date_Recd_11  Date_Recd_12  Consumer disp
uted?
0          0.0           0.0           0.0           0.0
    0
1          0.0           0.0           0.0           0.0
    0
3          0.0           0.0           0.0           0.0
    0
5          0.0           0.0           0.0           0.0
    0
8          0.0           0.0           0.0           0.0
    1

[5 rows x 99 columns]
```

In [49]:
```
data.shape
```
Out[49]:
```
(274645, 99)
```

In [50]:
```
data.isnull().values.sum()
```
Out[50]:
```
0
```

In [51]:
```
from sklearn.cross_validation import train_test_split

train, test = train_test_split(data, train_size=.80, test_size=.20)
```

In [52]:
```
train.shape
```
Out[52]:
```
(219716, 99)
```

In [53]:
```
train.head(3)
```
Out[53]:

	Product_Bank account or service	Product_Consumer Loan	Product_Credit card	Product_Credit reporting	Product_Del collection
119255	0.0	0.0	0.0	0.0	0.0
423145	0.0	0.0	0.0	0.0	0.0
574464	0.0	0.0	1.0	0.0	0.0

3 rows × 99 columns

In [54]:
```
from sklearn import linear_model
from sklearn.linear_model import LogisticRegressionCV
```

In [55]:
```
log_model = linear_model.LogisticRegressionCV()
```

In [56]:
```
log_model.fit(train.iloc[:,:-1], train.iloc[:,-1])
```
Out[56]:
```
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, verbos
e=0)
```

In [57]:
```
log_model.decision_function(train.iloc[:,:-1])
```
Out[57]:
```
array([-1.17655326, -1.02948142, -1.31679575, ..., -1.32022911,
       -1.04772884, -1.13939025])
```

In [58]:
```
log_model.predict(train.iloc[:,:-1])
```
Out[58]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [59]:
```
log_model.predict_log_proba(train.iloc[:,:-1])
```

Out[59]:
```
array([[-0.26875891, -1.44531217],
       [-0.30541797, -1.33489939],
       [-0.23743505, -1.55423081],
       ...,
       [-0.23671039, -1.5569395 ],
       [-0.30064772, -1.34837655],
       [-0.27764241, -1.41703266]])
```

In [60]:
```
log_model.predict_proba(train.iloc[:,:-1])
```

Out[60]:
```
array([[ 0.7643275 ,  0.2356725 ],
       [ 0.73681535,  0.26318465],
       [ 0.78864811,  0.21135189],
       ...,
       [ 0.78921982,  0.21078018],
       [ 0.74033854,  0.25966146],
       [ 0.75756767,  0.24243233]])
```

In [83]:
```
log_model.score(train.iloc[:,:-1], train.iloc[:,-1])
```

Out[83]:
```
0.77530084290629719
```

In [84]:
```
print(log_model.intercept_)
print(log_model.coef_)
```

```
[-1.30371132]
[[ -1.10784590e-02   2.78722046e-04   2.04892223e-03   2.81166792e-06
   -7.87324025e-03   3.83278910e-04   1.96968861e-02  -2.22135714e-04
   -5.61713447e-04  -2.67330624e-03  -9.05347954e-03   8.61544476e-03
    1.93226542e-02  -3.47025704e-03   4.56763493e-03   4.16132711e-03
   -6.54738044e-04   1.21114849e-03   3.66046259e-04  -3.65303711e-03
   -3.49980960e-03  -1.53056579e-03  -1.73104721e-02  -1.11878085e-03
    6.83470219e-04   1.43015736e-03  -6.49769320e-05  -3.26307450e-03
    6.18659502e-03   4.54995271e-03  -5.11601290e-03  -3.69592186e-03
    1.00188533e-04   1.84417473e-03  -3.37586298e-03   4.24306720e-04
    2.00597129e-03  -4.41661118e-05  -3.80752320e-03   1.51402293e-02
    1.29839966e-03  -6.92272249e-03  -5.32276756e-03   7.43015114e-04
    6.05240204e-02  -4.30834191e-02  -2.66889141e-02  -5.17091525e-03
    1.36779793e-02   0.00000000e+00  -1.99222906e-04   4.85433388e-04
   -9.97207166e-03  -5.09844517e-03  -3.74856456e-02   5.22717183e-02
   -9.83828509e-04   1.21656474e-02   3.44925518e-04  -4.36518939e-03
   -8.96650737e-03   4.69103739e-03   8.17938302e-04   2.59938597e-03
   -1.06624111e-03  -1.29909629e-03  -2.06968049e-03  -8.73662716e-04
    1.26689080e-03  -5.77244792e-03   4.03370402e-04   4.23550698e-03
    8.66344822e-04  -1.99262749e-03   6.47421411e-03  -1.12054747e-02
   -7.81283541e-03  -1.92566191e-03  -3.19437388e-03   4.48266054e-03
    1.09737027e-03  -2.52507214e-03   8.00736626e-03  -3.67412454e-03
    3.02334508e-03   7.25435267e-03  -6.84273994e-03  -1.39682098e-03
    1.31865650e-03  -3.75651762e-03  -4.85222387e-03   4.05343965e-04
    2.37748390e-04  -9.36140864e-04   4.10874928e-03   1.06486512e-02
    3.97737649e-03  -2.91031621e-03]]
```

```
In [85]: log_model.score(test.iloc[:,:-1], test.iloc[:,-1])

Out[85]: 0.77398095723570426
```

```
In [67]: test.describe()
```

Out[67]:

	Product_Bank account or service	Product_Consumer Loan	Product_Credit card	Product_Credit reporting	Product_Debt collection
count	54929.000000	54929.000000	54929.000000	54929.000000	54929.000000
mean	0.159388	0.022083	0.178612	0.002822	0.067341
std	0.366040	0.146955	0.383031	0.053046	0.250615
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 99 columns

```
In [86]: print(log_model.intercept_)
         print(log_model.coef_)
```

```
[-1.30371132]
[[ -1.10784590e-02   2.78722046e-04   2.04892223e-03   2.81166792e-06
   -7.87324025e-03   3.83278910e-04   1.96968861e-02  -2.22135714e-04
   -5.61713447e-04  -2.67330624e-03  -9.05347954e-03   8.61544476e-03
    1.93226542e-02  -3.47025704e-03   4.56763493e-03   4.16132711e-03
   -6.54738044e-04   1.21114849e-03   3.66046259e-04  -3.65303711e-03
   -3.49980960e-03  -1.53056579e-03  -1.73104721e-02  -1.11878085e-03
    6.83470219e-04   1.43015736e-03  -6.49769320e-05  -3.26307450e-03
    6.18659502e-03   4.54995271e-03  -5.11601290e-03  -3.69592186e-03
    1.00188533e-04   1.84417473e-03  -3.37586298e-03   4.24306720e-04
    2.00597129e-03  -4.41661118e-05  -3.80752320e-03   1.51402293e-02
    1.29839966e-03  -6.92272249e-03  -5.32276756e-03   7.43015114e-04
    6.05240204e-02  -4.30834191e-02  -2.66889141e-02  -5.17091525e-03
    1.36779793e-02   0.00000000e+00  -1.99222906e-04   4.85433388e-04
   -9.97207166e-03  -5.09844517e-03  -3.74856456e-02   5.22717183e-02
   -9.83828509e-04   1.21656474e-02   3.44925518e-04  -4.36518939e-03
   -8.96650737e-03   4.69103739e-03   8.17938302e-04   2.59938597e-03
   -1.06624111e-03  -1.29909629e-03  -2.06968049e-03  -8.73662716e-04
    1.26689080e-03  -5.77244792e-03   4.03370402e-04   4.23550698e-03
    8.66344822e-04  -1.99262749e-03   6.47421411e-03  -1.12054747e-02
   -7.81283541e-03  -1.92566191e-03  -3.19437388e-03   4.48266054e-03
    1.09737027e-03  -2.52507214e-03   8.00736626e-03  -3.67412454e-03
    3.02334508e-03   7.25435267e-03  -6.84273994e-03  -1.39682098e-03
    1.31865650e-03  -3.75651762e-03  -4.85222387e-03   4.05343965e-04
    2.37748390e-04  -9.36140864e-04   4.10874928e-03   1.06486512e-02
    3.97737649e-03  -2.91031621e-03]]
```

```
In [71]: print(log_model.Cs_)
```

```
[  1.00000000e-04   7.74263683e-04   5.99484250e-03   4.64158883e-02
   3.59381366e-01   2.78255940e+00   2.15443469e+01   1.66810054e+02
   1.29154967e+03   1.00000000e+04]
```

In [72]:
```
print(log_model.coefs_paths_)
```

```
{1: array([[[ -8.47263607e-03,   4.29488741e-04,  -1.81079223e-03, ...,
          -7.16735501e-04,  -3.14290348e-03,  -1.28552490e+00],
        [ -1.80229018e-02,   3.88680673e-03,   1.26414413e-02, ...,
          -6.03323639e-03,  -2.01781326e-02,  -1.43938192e+00],
        [ -8.86861981e-03,   2.01316027e-02,   6.52944004e-02, ...,
          -1.47099340e-02,  -6.70437698e-02,  -1.57145173e+00],
        ...,
        [  3.65581062e-02,   9.85543647e-02,   1.17349736e-01, ...,
          -1.92312005e-02,  -1.07181984e-01,  -1.70416374e+00],
        [  3.65545780e-02,   9.85555047e-02,   1.17347627e-01, ...,
          -1.92313757e-02,  -1.07183542e-01,  -1.70417699e+00],
        [  3.65526480e-02,   9.85586206e-02,   1.17349288e-01, ...,
          -1.92295238e-02,  -1.07184399e-01,  -1.70416647e+00]],

       [[ -9.86690592e-03,  -3.19869274e-04,   1.95499493e-03, ...,
           4.62589176e-03,  -2.18109549e-03,  -1.28460271e+00],
        [ -2.60686852e-02,  -1.49515974e-03,   3.00101193e-02, ...,
           2.64005421e-02,  -1.47683879e-02,  -1.43602804e+00],
        [ -3.47898696e-02,  -9.92056179e-03,   1.04175715e-01, ...,
           8.02386213e-02,  -5.25150514e-02,  -1.56839472e+00],
        ...,
        [  4.82807983e-02,  -1.66120373e-02,   2.47615365e-01, ...,
           1.15019449e-01,  -8.31855855e-02,  -1.67532412e+00],
        [  4.82759882e-02,  -1.66135411e-02,   2.47613533e-01, ...,
           1.15019716e-01,  -8.31890364e-02,  -1.67534614e+00],
        [  4.82773041e-02,  -1.66138107e-02,   2.47616609e-01, ...,
           1.15021213e-01,  -8.31893371e-02,  -1.67533553e+00]],

       [[ -7.22812278e-03,   4.05957170e-04,   2.32860981e-03, ...,
           4.32383957e-03,  -5.32926542e-04,  -1.27994738e+00],
        [ -1.56244107e-02,   3.14740781e-03,   3.04108449e-02, ...,
           2.55029573e-02,  -4.78851813e-03,  -1.42122820e+00],
        [ -1.99288296e-03,   1.21258875e-02,   1.05015981e-01, ...,
           7.99314183e-02,  -2.22624151e-02,  -1.55467259e+00],
        ...,
        [  7.76494139e-02,   3.54834549e-02,   1.93662839e-01, ...,
           1.16987191e-01,  -4.01838288e-02,  -1.66666464e+00],
        [  7.75186784e-02,   3.54791254e-02,   1.93623820e-01, ...,
           1.17003092e-01,  -4.02077569e-02,  -1.66668284e+00],
        [  7.75719495e-02,   3.54806959e-02,   1.93644888e-01, ...,
           1.17014060e-01,  -4.02132552e-02,  -1.66664381e+00]]])}
```

```
In [73]: print(log_model.scores_)
```

```
{1: array([[ 0.77396101,  0.77396101,  0.77396101,  0.77396101,  0.7739
6101,
         0.77396101,  0.77396101,  0.77396101,  0.77396101,  0.7739610
1],
       [ 0.77399093,  0.77399093,  0.77399093,  0.77399093,  0.7739909
3,
         0.77399093,  0.77399093,  0.77399093,  0.77399093,  0.7739909
3],
       [ 0.77399093,  0.77399093,  0.77399093,  0.77399093,  0.7739909
3,
         0.77399093,  0.77399093,  0.77399093,  0.77399093,  0.7739909
3]])}
```

```
In [74]: print(log_model.C_)
```

```
[ 0.0001]
```

```
In [87]: print(log_model.n_iter_)
```

```
[[[ 9 11 23 35 59 48  0 36  0  0]
  [ 9 11 22 36 64 65 23  0  0  0]
  [ 9 11 20 51 58 48 12  1  2  0]]]
```

```
In [ ]:
```