# upstart_data_challenge

January 17, 2018

In [16]:
```
"""
Applicant: Anthony Spalvieri-Kruse
Date: 01/17/2018
Email: ask417@nyu.edu
"""

# This tells matplotlib not to try opening a new window for each plot.
%matplotlib inline

# General libraries.
import re
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sb

# SK-learn model selection
from sklearn.cross_validation import train_test_split

# SK-learn libraries for learning.
from sklearn.linear_model import LogisticRegression

import statsmodels.api as sm
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

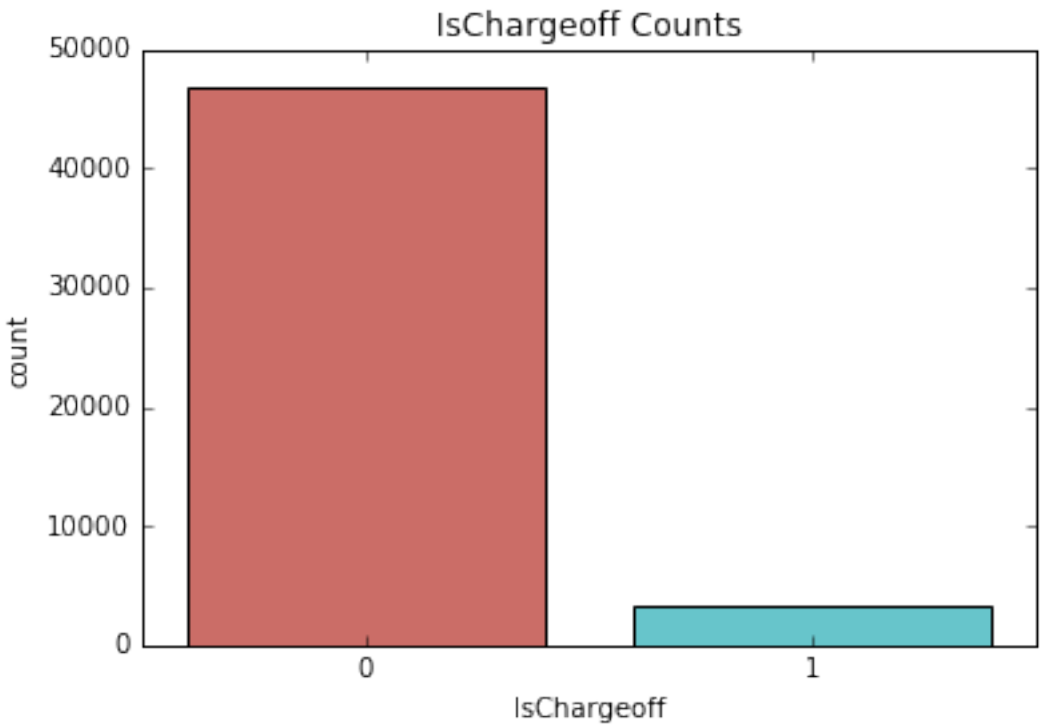
In [3]:
```
%%bash
ls
```

```
loan_timing.csv
problem_statement.pdf
upstart_data_challenge.ipynb
```

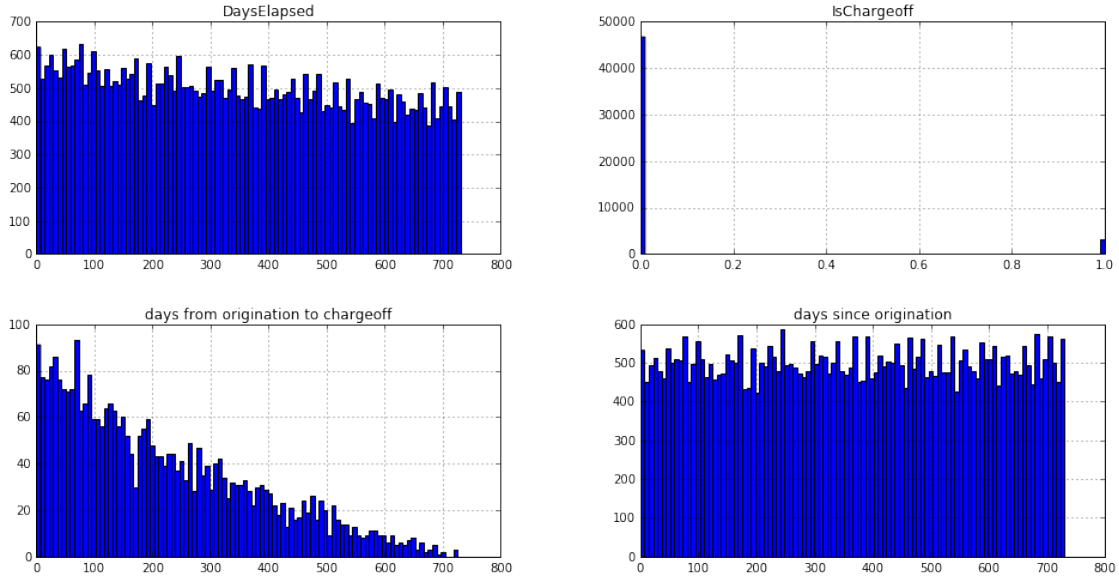
In [4]:
```
df = pd.read_csv("loan_timing.csv")

df["IsChargeoff"] = np.where(df["days from origination to chargeoff"].isnull(), 0, 1)

df["DaysElapsed"] = np.where(df["days from origination to chargeoff"].isnull(),
                             df["days since origination"].astype(int, errors="ignore"),
                             df["days from origination to chargeoff"].astype(int, errors="ignor
```

In [5]:
```
#Sanity check, chargeoff days elapsed should never be greater than days since origination
df[df["days since origination"] < df["days from origination to chargeoff"]]
```

```
Out[5]: Empty DataFrame
        Columns: [days since origination, days from origination to chargeoff, IsChargeoff, DaysElapsed]
        Index: []

In [6]: #Checking for missing values
        df.isnull().sum()

Out[6]: days since origination                   0
        days from origination to chargeoff    46695
        IsChargeoff                                0
        DaysElapsed                                0
        dtype: int64

In [7]: df.describe()

Out[7]:        days since origination  days from origination to chargeoff  \
        count             50000.000000                         3305.000000
        mean                365.844760                          214.852345
        std                 210.965114                          165.233874
        min                   0.000000                            0.000000
        25%                 183.000000                           76.000000
        50%                 366.000000                          180.000000
        75%                 549.000000                          323.000000
        max                 730.000000                          725.000000

                IsChargeoff   DaysElapsed
        count  50000.000000  50000.000000
        mean       0.066100    348.701960
        std        0.248459    211.373155
        min        0.000000      0.000000
        25%        0.000000    163.000000
        50%        0.000000    341.000000
        75%        0.000000    530.000000
        max        1.000000    730.000000

In [8]: ax = sb.countplot(x="IsChargeoff",data=df, palette='hls')
        ax.set_title("IsChargeoff Counts")
        plt.show()
```

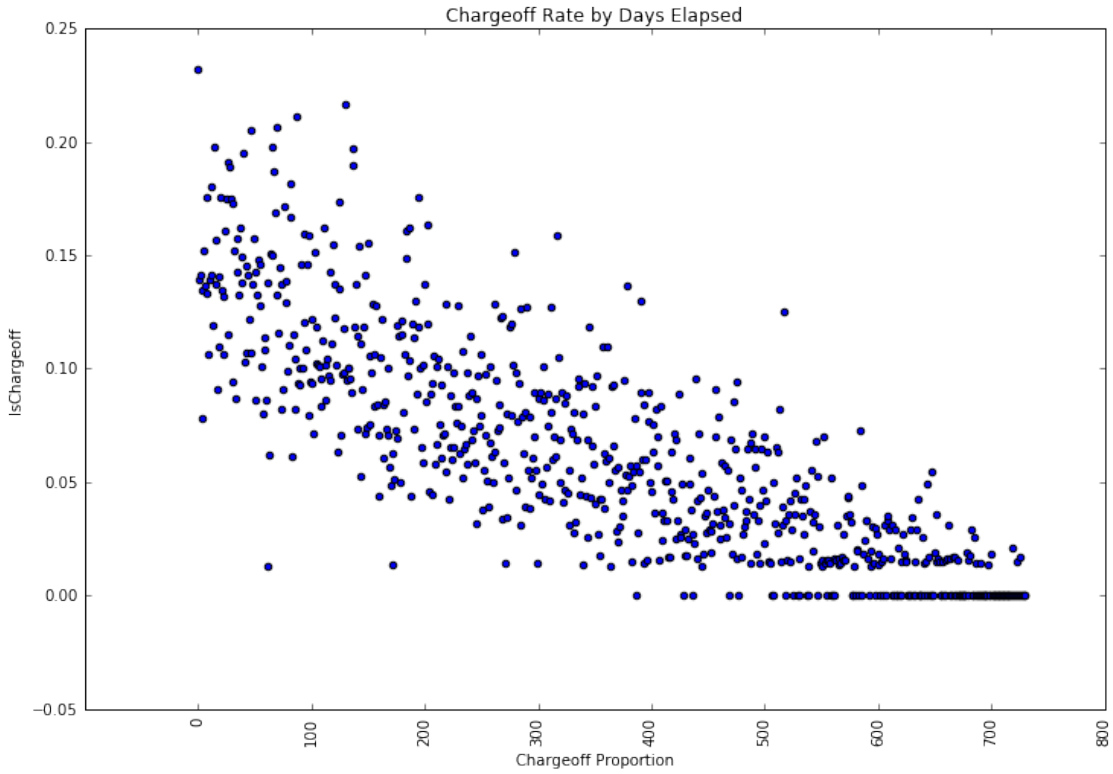



```
In [9]: df.hist(figsize=(16,8), bins=100)

Out[9]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x11090f090>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x110d87090>],
               [<matplotlib.axes._subplots.AxesSubplot object at 0x110dfde90>,
                <matplotlib.axes._subplots.AxesSubplot object at 0x110e6e4d0>]],
              dtype=object)
```



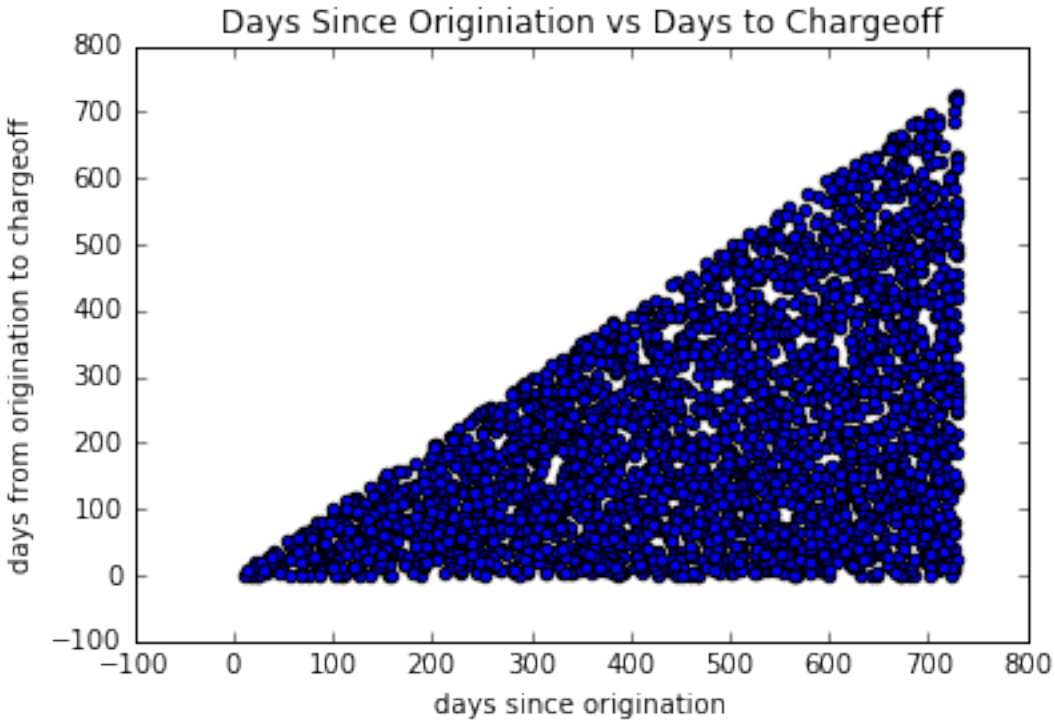
```
In [10]: plot = df.groupby("DaysElapsed")["IsChargeoff"].mean()\
                                                        .reset_index()\
                                                        .plot.scatter(x="DaysElapsed",
                                                                      y="IsChargeoff",
                                                                      xticks=[0,800],
                                                                      figsize=(12,8),
                                                                      title="Chargeoff Rate by Days Ela
         plot.set_xlabel("Chargeoff Proportion")
         plot.set_xticks([i*100 for i in range(9)])
         plot.set_xticklabels([i*100 for i in range(9)], rotation=90)
```

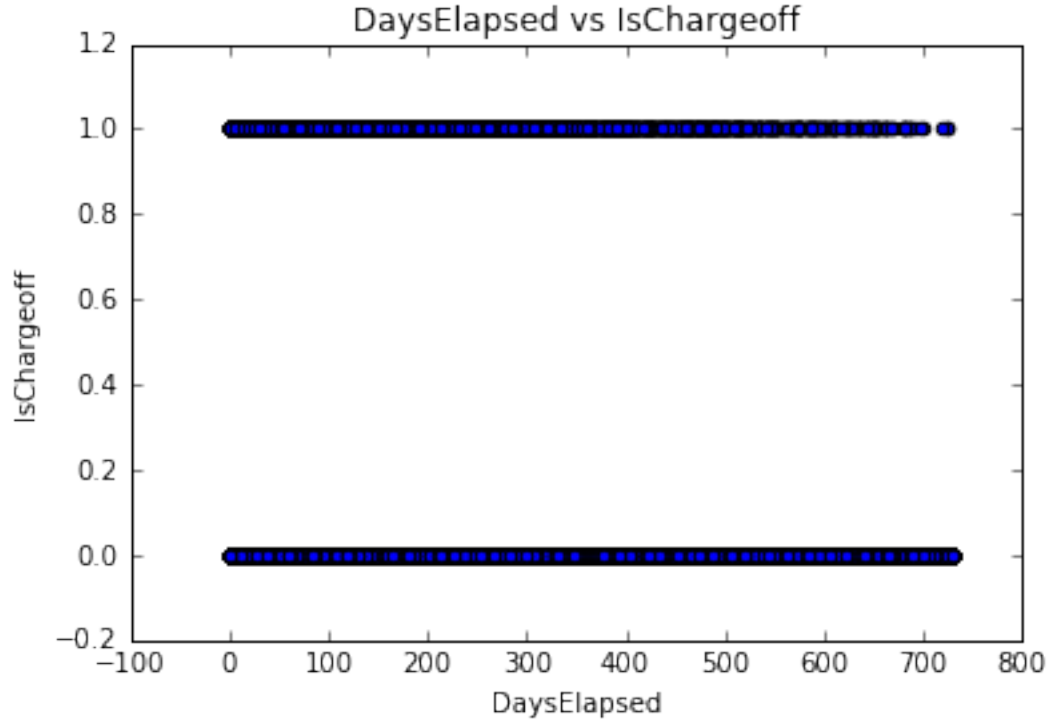
```
Out[10]: [<matplotlib.text.Text at 0x1118b2ad0>,
          <matplotlib.text.Text at 0x11173f350>,
          <matplotlib.text.Text at 0x104eea250>,
          <matplotlib.text.Text at 0x104eea750>,
          <matplotlib.text.Text at 0x104eeac50>,
          <matplotlib.text.Text at 0x104ee5610>,
          <matplotlib.text.Text at 0x104ee1690>,
          <matplotlib.text.Text at 0x104ef12d0>,
          <matplotlib.text.Text at 0x104ef17d0>]
```




```
In [11]: df.plot.scatter(x='days since origination', y="days from origination to chargeoff",
                         title="Days Since Originiation vs Days to Chargeoff")
         df.plot.scatter(x="DaysElapsed", y="IsChargeoff",
                         title="DaysElapsed vs IsChargeoff")
```

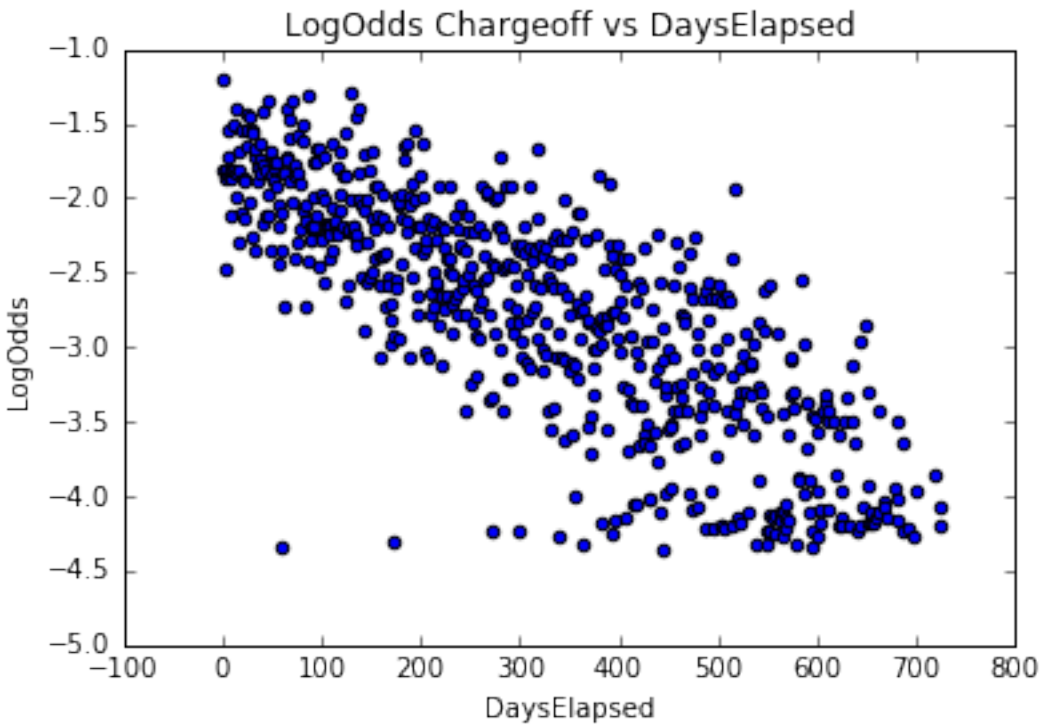
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x112170d10>



In [35]:
```
"""
Observation: The longer someone goes without defaulting on their loan, the less likely they ar

    We have a fairly uniform distribution of days since origination, but the distribution of c
    exponential exponential distribution, plateauing to 0 as we reach the 3 year mark.

    Could model this using logistic regression, where our dependent variable is IsDefault, and
    days elapsed since origination for IsDefault=0, else days_from_orig_to_chargeoff

    Assumptions:

        1) Will assume here that the sample is random and drawn from the same population
        2) Binary dependent variable -- this is the case here
        2) Independent Observations -- reasonable assumption, will assume two loan datapoints
        3) No multicollinearity -- only one predictor, so not an issue
        4) Linearity of predictor variables and log odds.
            i. Will do a crude test of this below by plotting observed log odds alongside Days
        5) Large sample size -- given that we only have a single predictor our sample size is
"""

# Plot clearly shows a strong linear relationship between the log odds of a
# chargeoff and our one predictor variable
probs = df.groupby("DaysElapsed")["IsChargeoff"].mean().reset_index()
probs = probs[probs["IsChargeoff"]!=0]
probs["Odds"] = probs.IsChargeoff/(1-probs.IsChargeoff)
probs["LogOdds"] = np.log(probs["Odds"])
ax = probs.plot.scatter(x="DaysElapsed", y="LogOdds")

ax.set_title("LogOdds Chargeoff vs DaysElapsed")
```

Out[35]:
```
<matplotlib.text.Text at 0x11d10f090>
```




In [65]:
```
"""
Because i'm using this model to predict the chargeoff probability as a function of DaysElapsed
omitting a test/train split and performing a regression on all of the data instead.
"""
lr = LogisticRegression()
X = np.reshape(df.DaysElapsed.values,(-1,1))
Y = np.reshape(df.IsChargeoff.values,(-1,1)).ravel()
lr.fit(X, Y)

X_not_yet_charged_off = np.reshape(df["DaysElapsed"][df.IsChargeoff == 0].values,(-1,1))
predictions = lr.predict_proba(X_not_yet_charged_off)
chargeoff_probs = predictions[:,1]
print "Intercept: ", lr.intercept_
print "Coefficient: ", lr.coef_[0]
print "Expectation future chargeoffs: ", sum(chargeoff_probs)
print "With each extra day elapsed since origination, our odds \
of a chargeoff change by a factor of", round(np.exp(result.params)[0],3)
```

```
Intercept:  [-1.61603736]
Coefficient:  [-0.00365308]
Expectation future chargeoffs:  2997.7507665416315
With each extra day elapsed since origination, our odds of a chargeoff change by a factor of 0.991
```

In [ ]:
```
"""
Conclusion: We've trained a logistic regression model to obtain the probability of chargeoff as
    of DaysElapsed since origination. If we take the probability of a chargeoff given DaysElaps
    random variable X_i, assuming our loans are independent, we know that
```

```
        E[{X_1,X_2,...,X_N}] = E[X_1] + E[X_2] + ... + E[X_N]

    And we know that for each E[X_i], the expectation of a chargeoff is just the probability of
    the given value of X.  So in this way, we can sum the probability predictions for each valu
    obtain our estimate.  We already see that of the 50000 loans in our sample, 3305 have alrea
    Summing the probabilities for the remaining data points, we get an expectation of 2997.75 c
    my model predicts that by the end of the 3 years, we'll have 2997.75 + 3305 ~= 6303 chargeo
    100*(6303/50000) = 12.606% of our loans.
"""
```