Project Loan Data Analytic

January 2, 2019

1 Prosper Loan Data Analytic

In [1]:
```
#import libraries
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
% matplotlib inline
```

In [2]:
```
# Read and check the loan data
df_loan = pd.read_csv('prosperLoanData.csv')
df_loan.head(10)
```

Out[2]:
```
                ListingKey  ListingNumber            ListingCreationDate  \
0  1021339766868145413AB3B         193129  2007-08-26 19:09:29.263000000
1  10273602499503308B223C1        1209647  2014-02-27 08:28:07.900000000
2  0EE9337825851032864889A          81716  2007-01-05 15:00:47.090000000
3  0EF5356002482715299901A         658116  2012-10-22 11:02:35.010000000
4  0F023589499656230C5E3E2         909464  2013-09-14 18:38:39.097000000
5  0F05359734824199381F61D        1074836  2013-12-14 08:26:37.093000000
6  0F0A3576754255009D63151         750899  2013-04-12 09:52:56.147000000
7  0F1035772717087366F9EA7         768193  2013-05-05 06:49:27.493000000
8  0F043596202561788EA13D5        1023355  2013-12-02 10:43:39.117000000
9  0F043596202561788EA13D5        1023355  2013-12-02 10:43:39.117000000

  CreditGrade  Term LoanStatus           ClosedDate  BorrowerAPR  \
0           C    36  Completed  2009-08-14 00:00:00      0.16516
1         NaN    36    Current                  NaN      0.12016
2          HR    36  Completed  2009-12-17 00:00:00      0.28269
3         NaN    36    Current                  NaN      0.12528
4         NaN    36    Current                  NaN      0.24614
5         NaN    60    Current                  NaN      0.15425
6         NaN    36    Current                  NaN      0.31032
7         NaN    36    Current                  NaN      0.23939
8         NaN    36    Current                  NaN      0.07620
9         NaN    36    Current                  NaN      0.07620
```

```
   BorrowerRate  LenderYield    ...     LP_ServiceFees  LP_CollectionFees  \
0        0.1580       0.1380    ...            -133.18                0.0
1        0.0920       0.0820    ...               0.00                0.0
2        0.2750       0.2400    ...             -24.20                0.0
3        0.0974       0.0874    ...            -108.01                0.0
4        0.2085       0.1985    ...             -60.27                0.0
5        0.1314       0.1214    ...             -25.33                0.0
6        0.2712       0.2612    ...             -22.95                0.0
7        0.2019       0.1919    ...             -69.21                0.0
8        0.0629       0.0529    ...             -16.77                0.0
9        0.0629       0.0529    ...             -16.77                0.0

   LP_GrossPrincipalLoss  LP_NetPrincipalLoss LP_NonPrincipalRecoverypayments  \
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
5                    0.0                  0.0                             0.0
6                    0.0                  0.0                             0.0
7                    0.0                  0.0                             0.0
8                    0.0                  0.0                             0.0
9                    0.0                  0.0                             0.0

   PercentFunded  Recommendations InvestmentFromFriendsCount  \
0            1.0                0                          0
1            1.0                0                          0
2            1.0                0                          0
3            1.0                0                          0
4            1.0                0                          0
5            1.0                0                          0
6            1.0                0                          0
7            1.0                0                          0
8            1.0                0                          0
9            1.0                0                          0

  InvestmentFromFriendsAmount Investors
0                         0.0       258
1                         0.0         1
2                         0.0        41
3                         0.0       158
4                         0.0        20
5                         0.0         1
6                         0.0         1
7                         0.0         1
8                         0.0         1
9                         0.0         1
```

```
[10 rows x 81 columns]
```

In [3]:
```
# Understand the dataset
df_loan.info()
```

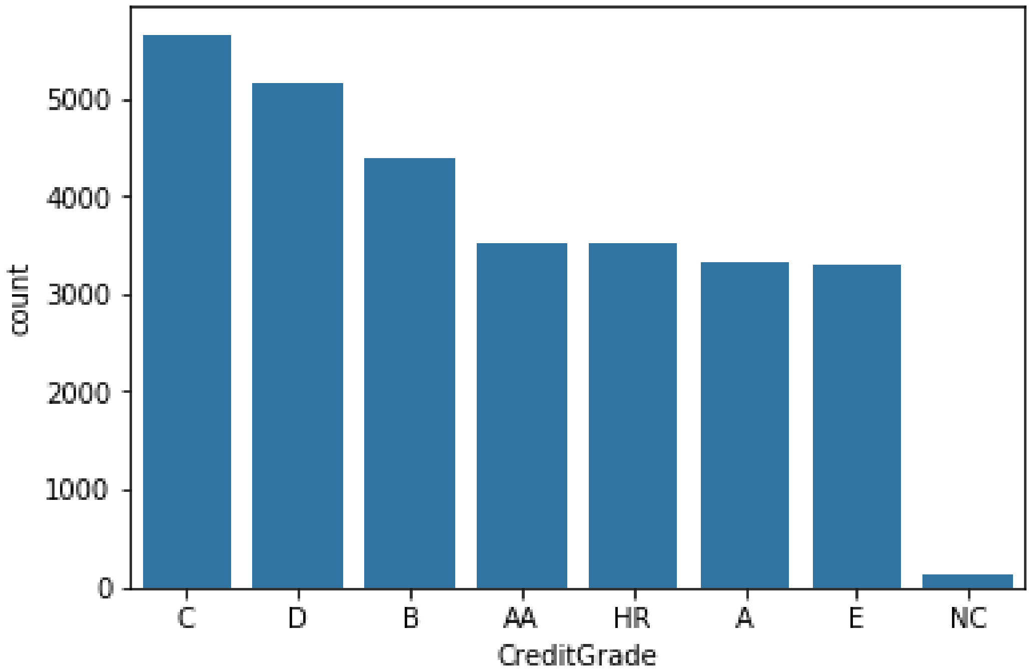
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
ListingKey                             113937 non-null object
ListingNumber                          113937 non-null int64
ListingCreationDate                    113937 non-null object
CreditGrade                            28953 non-null object
Term                                   113937 non-null int64
LoanStatus                             113937 non-null object
ClosedDate                             55089 non-null object
BorrowerAPR                            113912 non-null float64
BorrowerRate                           113937 non-null float64
LenderYield                            113937 non-null float64
EstimatedEffectiveYield                84853 non-null float64
EstimatedLoss                          84853 non-null float64
EstimatedReturn                        84853 non-null float64
ProsperRating (numeric)                84853 non-null float64
ProsperRating (Alpha)                  84853 non-null object
ProsperScore                           84853 non-null float64
ListingCategory (numeric)              113937 non-null int64
BorrowerState                          108422 non-null object
Occupation                             110349 non-null object
EmploymentStatus                       111682 non-null object
EmploymentStatusDuration               106312 non-null float64
IsBorrowerHomeowner                    113937 non-null bool
CurrentlyInGroup                       113937 non-null bool
GroupKey                               13341 non-null object
DateCreditPulled                       113937 non-null object
CreditScoreRangeLower                  113346 non-null float64
CreditScoreRangeUpper                  113346 non-null float64
FirstRecordedCreditLine                113240 non-null object
CurrentCreditLines                     106333 non-null float64
OpenCreditLines                        106333 non-null float64
TotalCreditLinespast7years             113240 non-null float64
OpenRevolvingAccounts                  113937 non-null int64
OpenRevolvingMonthlyPayment            113937 non-null float64
InquiriesLast6Months                   113240 non-null float64
TotalInquiries                         112778 non-null float64
CurrentDelinquencies                   113240 non-null float64
AmountDelinquent                       106315 non-null float64
DelinquenciesLast7Years                112947 non-null float64
PublicRecordsLast10Years               113240 non-null float64
PublicRecordsLast12Months              106333 non-null float64
```

```
RevolvingCreditBalance                  106333 non-null float64
BankcardUtilization                     106333 non-null float64
AvailableBankcardCredit                 106393 non-null float64
TotalTrades                             106393 non-null float64
TradesNeverDelinquent (percentage)      106393 non-null float64
TradesOpenedLast6Months                 106393 non-null float64
DebtToIncomeRatio                       105383 non-null float64
IncomeRange                             113937 non-null object
IncomeVerifiable                        113937 non-null bool
StatedMonthlyIncome                     113937 non-null float64
LoanKey                                 113937 non-null object
TotalProsperLoans                       22085 non-null float64
TotalProsperPaymentsBilled              22085 non-null float64
OnTimeProsperPayments                   22085 non-null float64
ProsperPaymentsLessThanOneMonthLate     22085 non-null float64
ProsperPaymentsOneMonthPlusLate         22085 non-null float64
ProsperPrincipalBorrowed                22085 non-null float64
ProsperPrincipalOutstanding             22085 non-null float64
ScorexChangeAtTimeOfListing             18928 non-null float64
LoanCurrentDaysDelinquent               113937 non-null int64
LoanFirstDefaultedCycleNumber           16952 non-null float64
LoanMonthsSinceOrigination              113937 non-null int64
LoanNumber                              113937 non-null int64
LoanOriginalAmount                      113937 non-null int64
LoanOriginationDate                     113937 non-null object
LoanOriginationQuarter                  113937 non-null object
MemberKey                               113937 non-null object
MonthlyLoanPayment                      113937 non-null float64
LP_CustomerPayments                     113937 non-null float64
LP_CustomerPrincipalPayments            113937 non-null float64
LP_InterestandFees                      113937 non-null float64
LP_ServiceFees                          113937 non-null float64
LP_CollectionFees                       113937 non-null float64
LP_GrossPrincipalLoss                   113937 non-null float64
LP_NetPrincipalLoss                     113937 non-null float64
LP_NonPrincipalRecoverypayments         113937 non-null float64
PercentFunded                           113937 non-null float64
Recommendations                         113937 non-null int64
InvestmentFromFriendsCount              113937 non-null int64
InvestmentFromFriendsAmount             113937 non-null float64
Investors                               113937 non-null int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

2 Exploratory Data Analysis

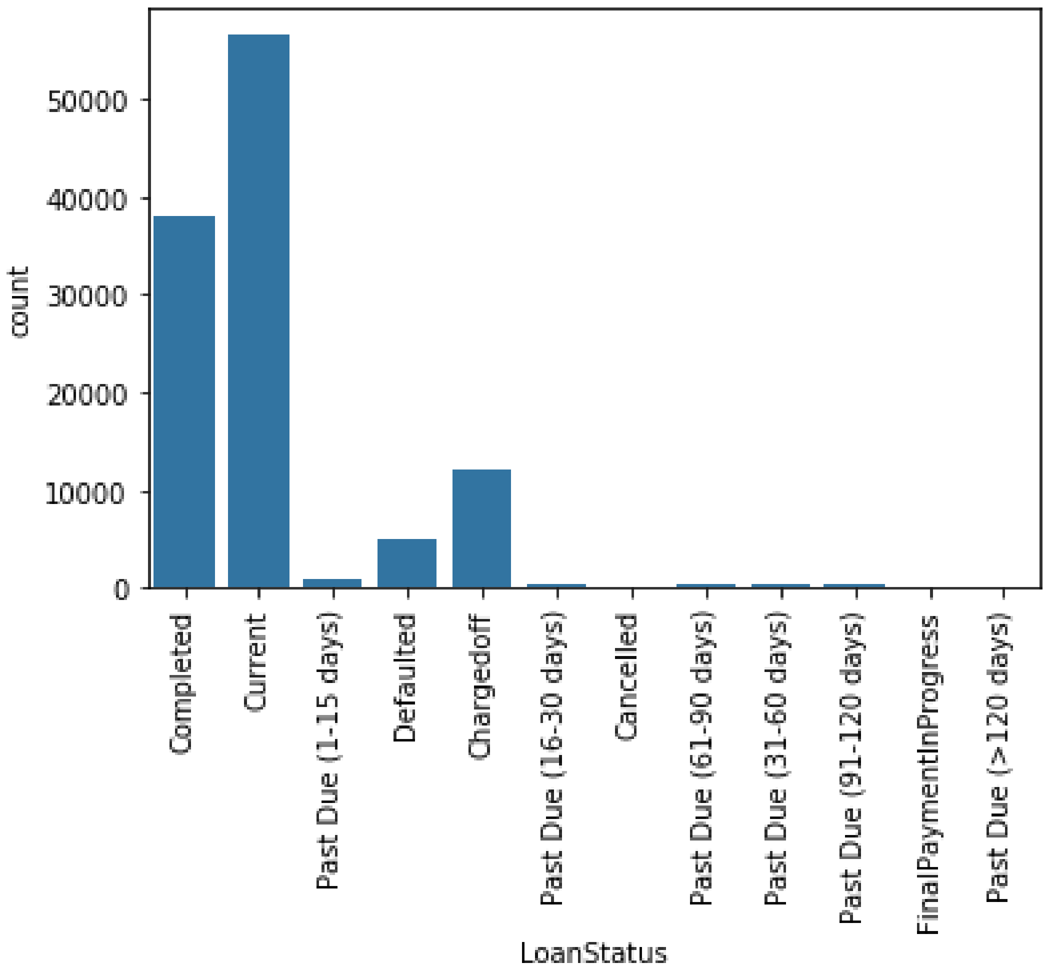
```
In [4]: # Check the univariate relationship of Credit Grade
        base_color = sb.color_palette()[0]
        credit_grade = df_loan['CreditGrade'].value_counts().index
        sb.countplot(data = df_loan, x = 'CreditGrade', color = base_color, order = credit_grac
```

```
Out[4]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1c723e10>
```



Here, we found out that the number of credit grade for C is the highest while D is the second

```
In [5]: # Check the univariate relationship of Loan Status
        base_color = sb.color_palette()[0]
        sb.countplot(data = df_loan, x = 'LoanStatus', color = base_color)
        plt.xticks(rotation = 90);
```



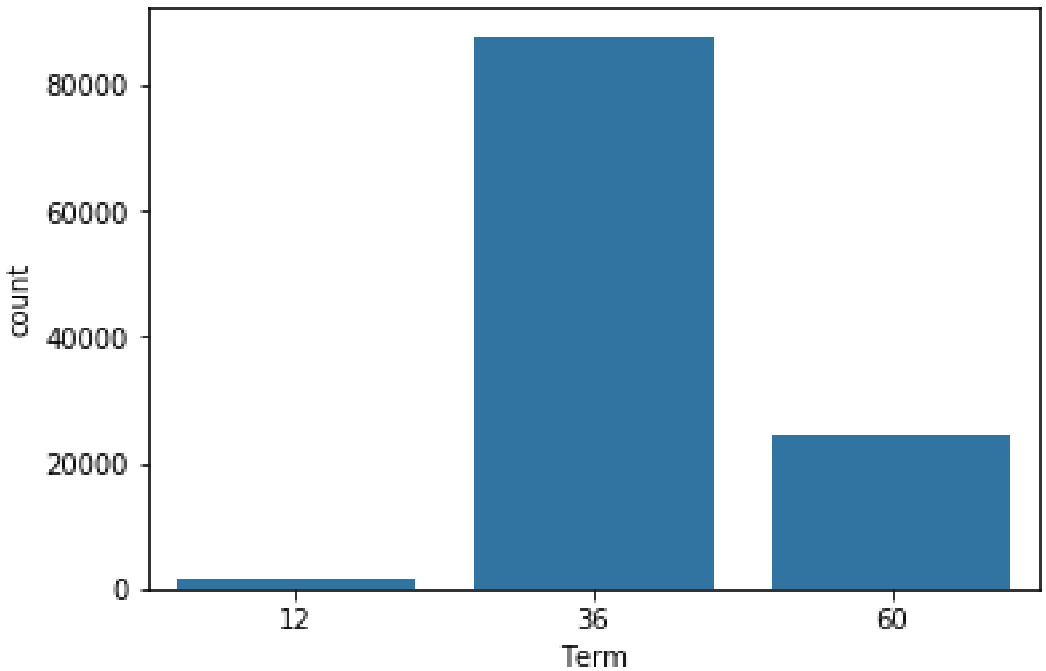
Here, we found out that the number of Loan that is still ongoing is the highest while Completed loan is the second highest

```
In [6]: # Check the unique values in Term column
        df_loan['Term'].unique()
```

```
Out[6]: array([36, 60, 12])
```

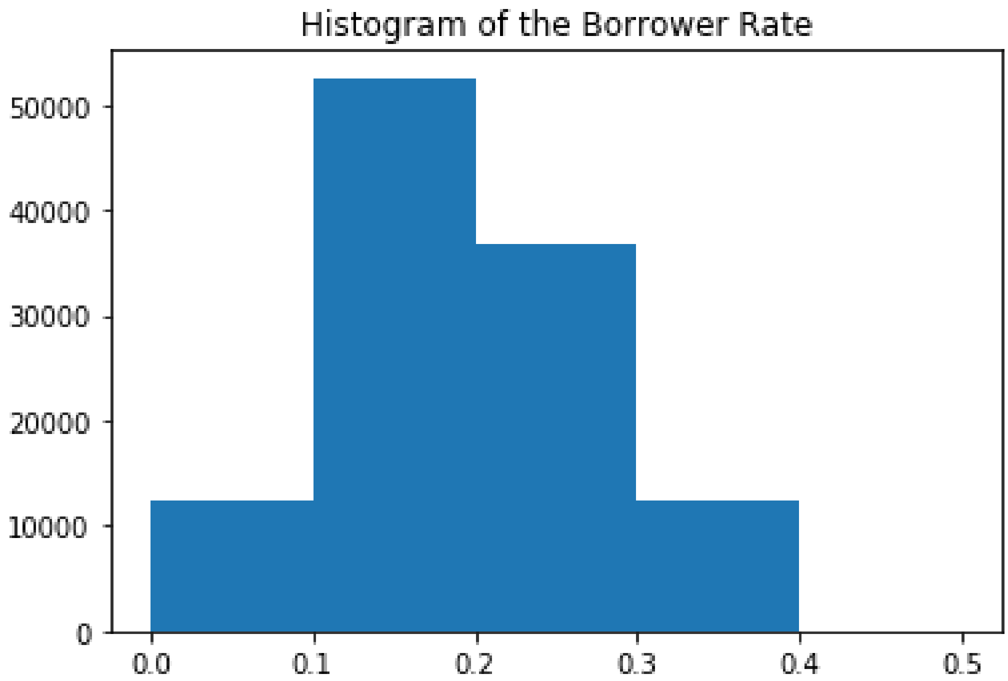
There are three different terms in the Loan Data which is 12, 36 and 60 months

```
In [7]: # Check the univariate relationship in Term
        base_color = sb.color_palette()[0]
        credit_grade = df_loan['Term'].value_counts().index
        plt.xlabel('Term (months)')
        sb.countplot(data = df_loan, x = 'Term', color = base_color);
```



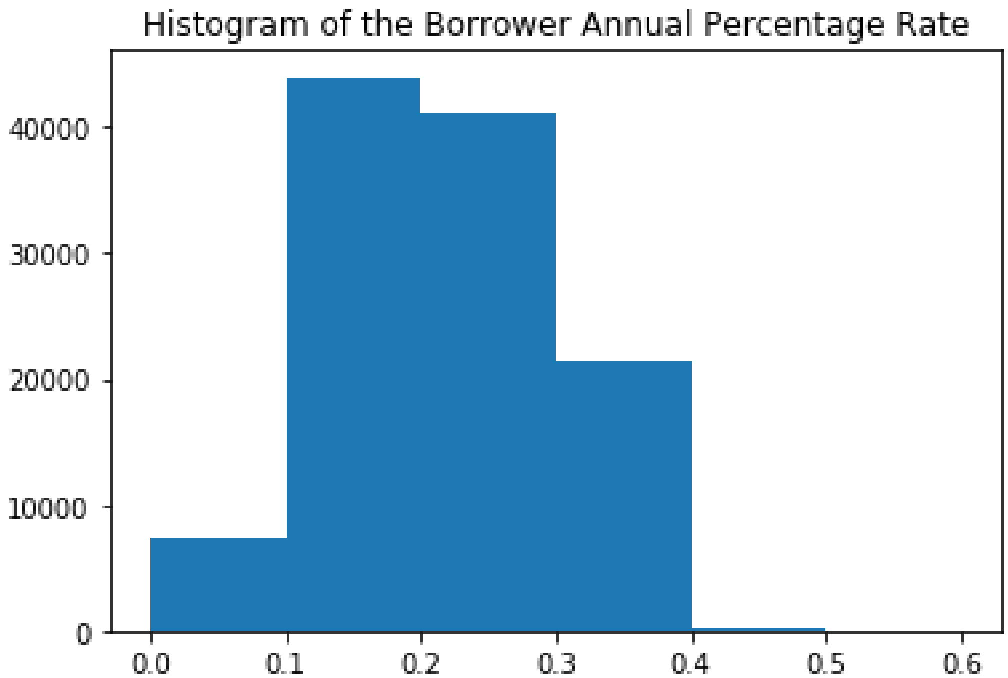
Here, we found out that 36 months is the highest number of loan

```
In [8]: # Check the univariate relationship of Borrower rate
        bin_edges = np.arange(0, df_loan['BorrowerRate'].max()+ 0.1, 0.1)
        plt.title('Histogram of the Borrower Rate')
        plt.xlabel('Borrower Rate')
        plt.hist(data = df_loan, x = 'BorrowerRate', bins = bin_edges);
```



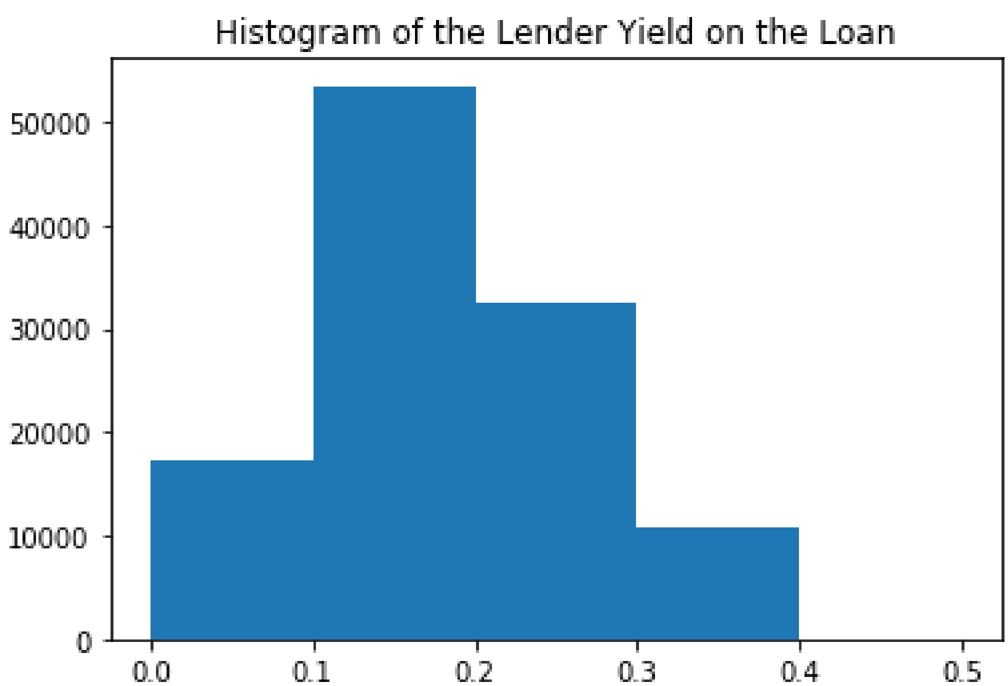
Here, we have found that between 0.1 and 0.2 is the highest borrower rate while the range between 0.0 and 0.1 and the range between 0.3 and 0.4 has the lowest borrower rate

In [9]:
```
# Check the univariate relationship in Borrower Annual Percentage Rate which is known
bin_edges = np.arange(0, df_loan['BorrowerAPR'].max()+ 0.1, 0.1)
plt.title('Histogram of the Borrower Annual Percentage Rate')
plt.hist(data = df_loan, x = 'BorrowerAPR', bins = bin_edges);
```



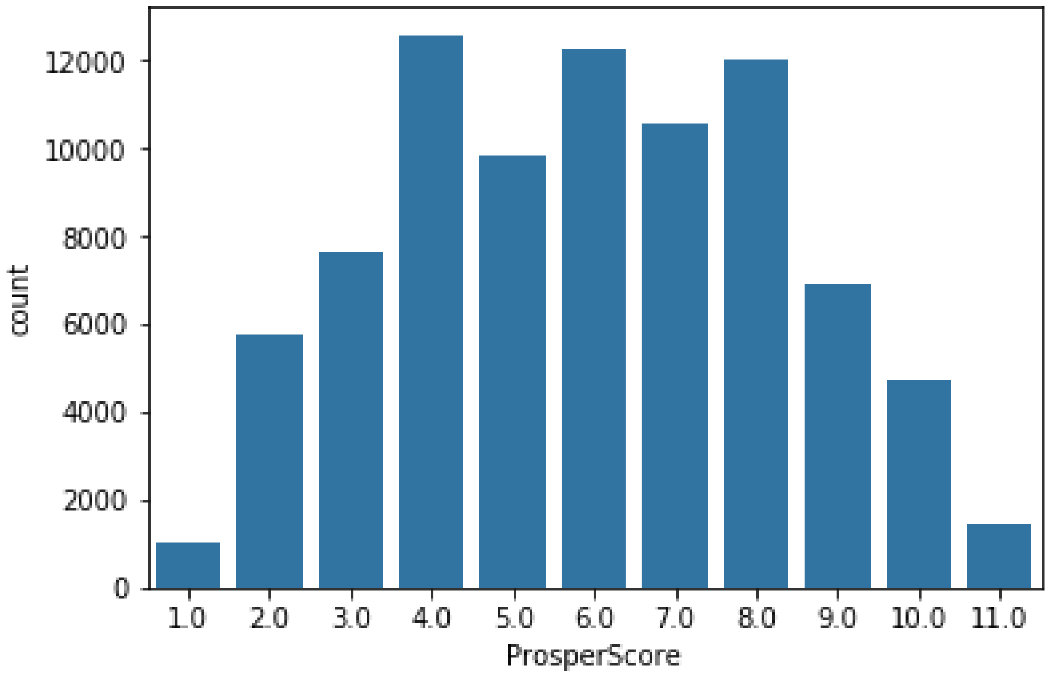
Here, we found out that the highest number of borrower annual percentage rate is between 0.1 and 0.2

In [10]:



Highest number of lender Yield is between 0.1 and 0.2

```
In [11]: # Check the univariate relationship in Prosper Score
         base_color = sb.color_palette()[0]
         credit_grade = df_loan['ProsperScore'].value_counts().index
         sb.countplot(data = df_loan, x = 'ProsperScore', color = base_color);
```

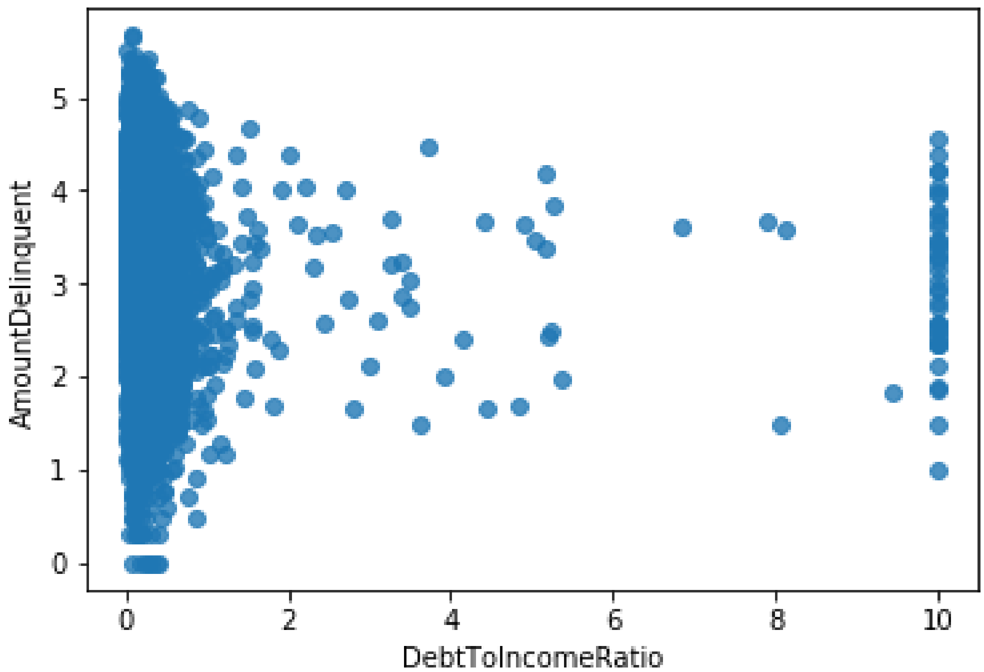


Bivariate Relationships 1. Scatterplot 2. Heatmaps 3. Violin Plots 4. Box Plot 5. Clustered Bar Charts 6. Line Plots

Multivariate Relationships

Scatterplot - to inspect relationship between two numeric variables

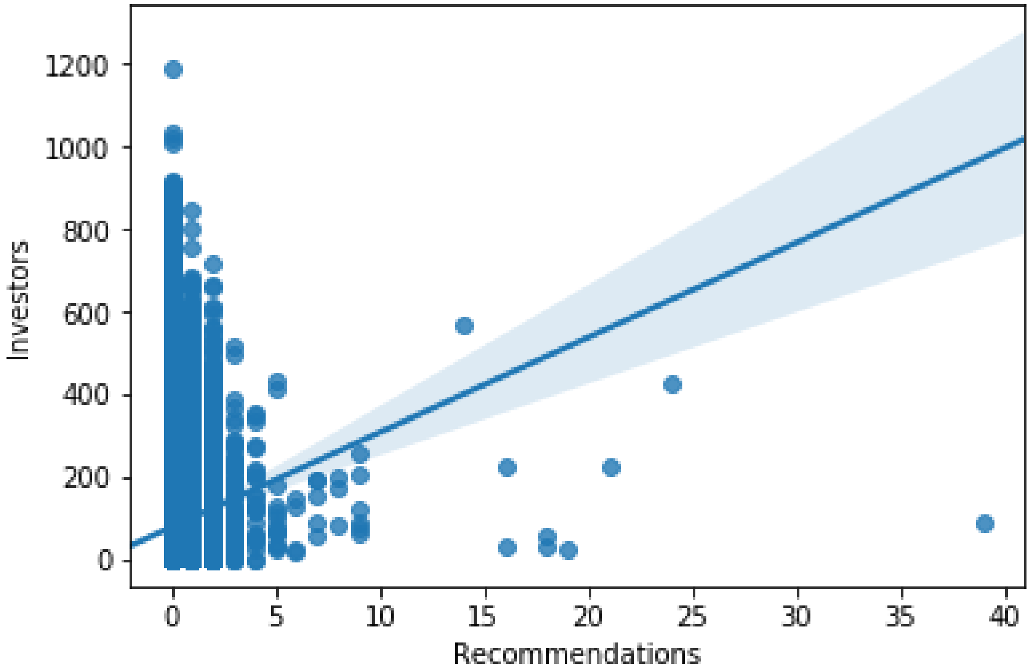
```
In [15]: # Check the Bivariate Relationship between DebtToIncomeRatio and AmountDeliquent
         def log_trans(x, inverse = False):
             if not inverse:
                 return np.log10(x)
             else:
                 return np.power(10, x)

         sb.regplot(df_loan['DebtToIncomeRatio'], df_loan['AmountDelinquent'].apply(log_trans)]

/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Here, we found out that there is no correlation between debt to income ratio and amount of deliquent

```
In [16]: # Check the Bivariate Relationship between Recommendations and Investors
         sb.regplot(data = df_loan, x = 'Recommendations', y = 'Investors');
```

```
/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

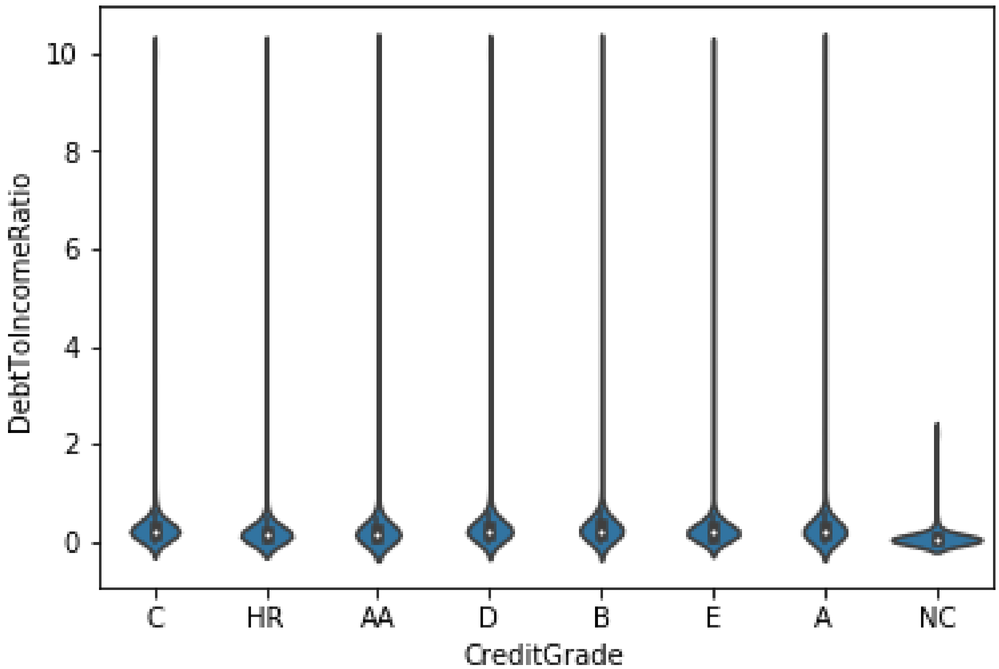


Here, we can see that the higher the number of recommendations, the higher the number of investor.

Box Plot - to show the relationship between a numeric variable and categorical variable

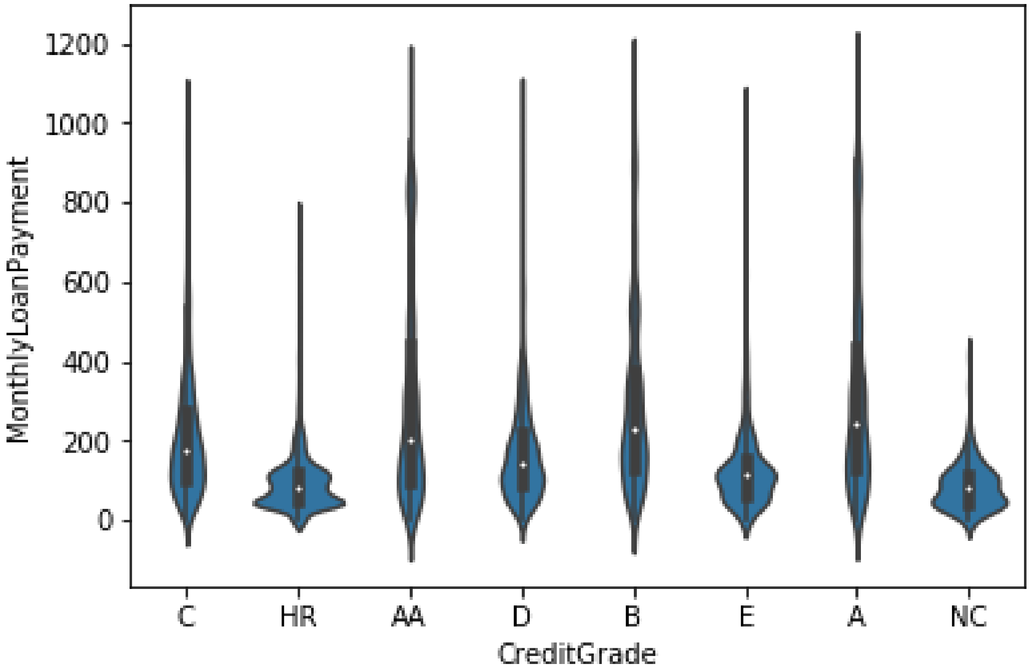
```
In [30]: # Check the Bivariate relationship between CreditGrade and DebtToIncomeRatio
         sb.violinplot(data = df_loan, x = 'CreditGrade', y = 'DebtToIncomeRatio', color = base
```

```
/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



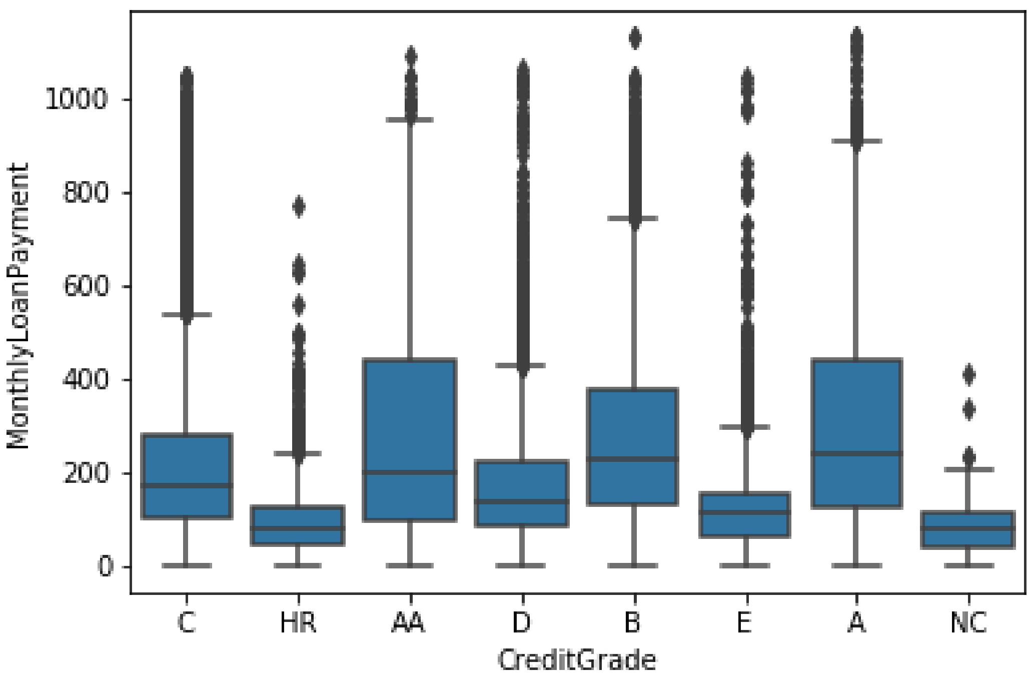
There is no insight found when we try to see if a better credit grade leads to lower debt to income ratio

```
In [35]: # Check the Bivariate Relationship between CreditGrade and MonthlyLoanPayment
         sb.violinplot(data = df_loan, x = 'CreditGrade', y = 'MonthlyLoanPayment', color = bas

/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tu
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



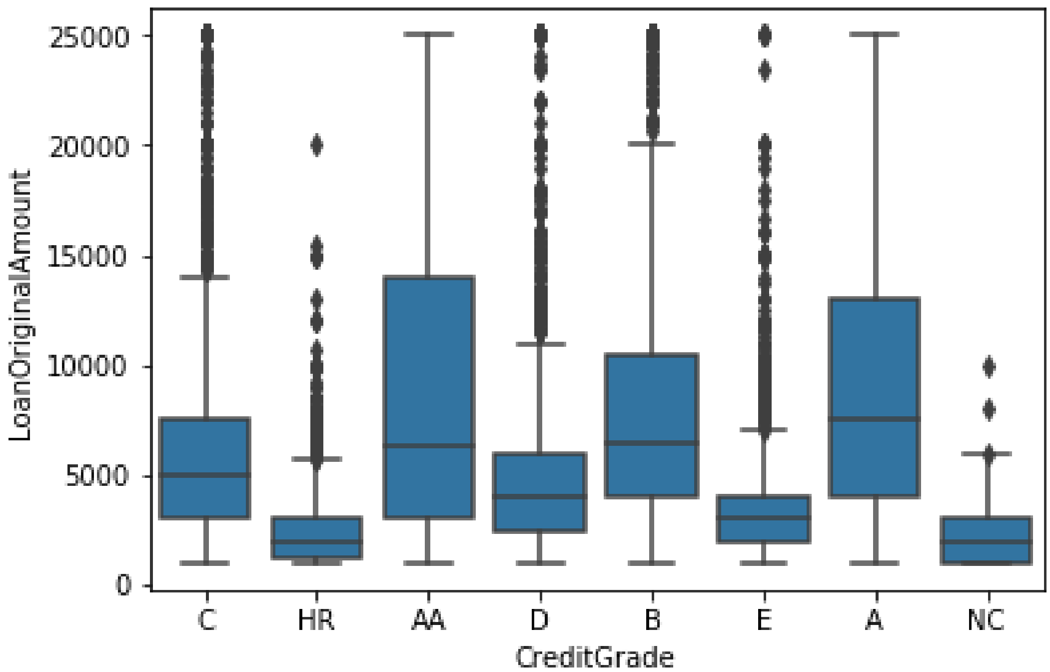
Here, we can discover that most credit grade falls between 0 to 200 monthly loan payment

```
In [36]: # Check the Bivariate Relationship between CreditGarde and MonthlyLoan Payment using b
         sb.boxplot(data = df_loan, x = 'CreditGrade', y = 'MonthlyLoanPayment', color = base_c
```



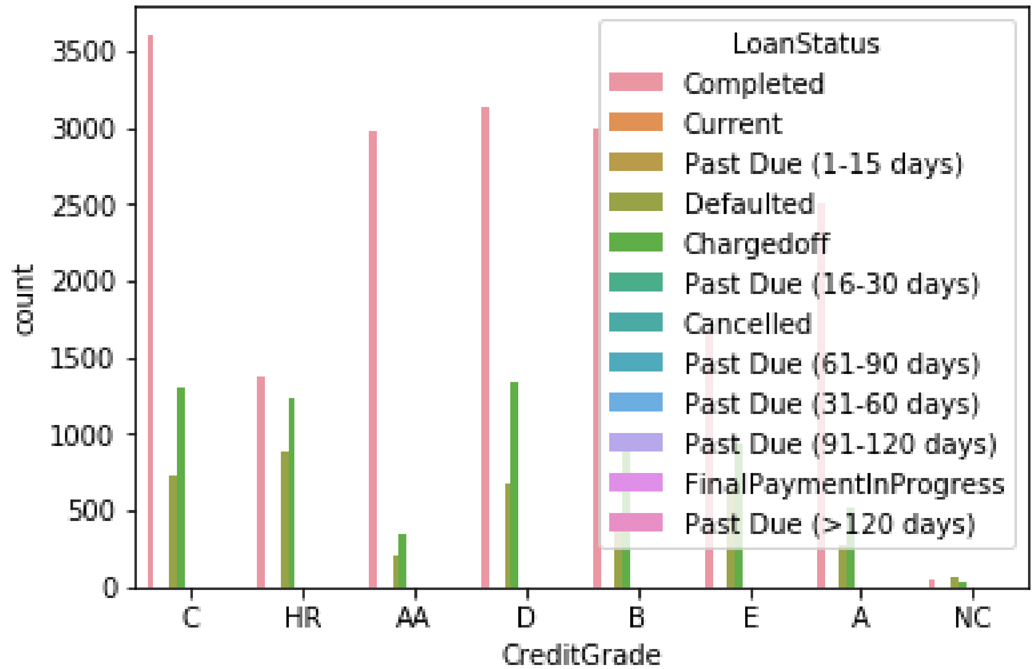
Here, we can discover clearly that High Credit Grade from Double A (AA) to C has among the highest monthly loan payment which is approximately USD 200.

In [37]:
```
# Checkt the Bivariate Relationship between CreditGrade and LoanOriginalAmount
sb.boxplot(data = df_loan, x = 'CreditGrade', y = 'LoanOriginalAmount', color = base_c
```



Here, we can understand why High Credit Grade from Double A (AA) to C has among the highest monthly loan payment. It is because they have the highest number of loan original amount

In [38]:
```
# Check the Bivariate Relatonship between CreditGrade and LoanStatus
sb.countplot(data = df_loan, x = 'CreditGrade', hue = 'LoanStatus');
```



Suprisingly, here we discover that Grade C has the highest number of completed loan not the conventional logic that the highest credit grade which is Double A (AA) that has the highest number of completed loan.

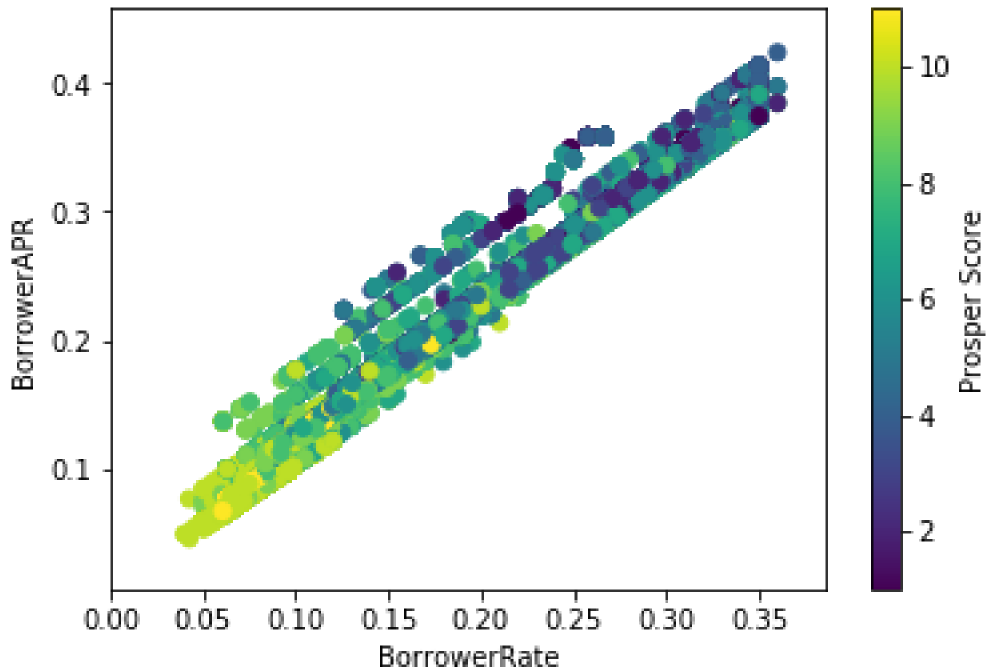
Multivariate Technique

Encoding via Size

In [51]:
```
# Understand the statistics in BorrowerAPR column
df_loan['BorrowerAPR'].describe()
```

Out[51]:
```
count    113912.000000
mean          0.218828
std           0.080364
min           0.006530
25%           0.156290
50%           0.209760
75%           0.283810
max           0.512290
Name: BorrowerAPR, dtype: float64
```

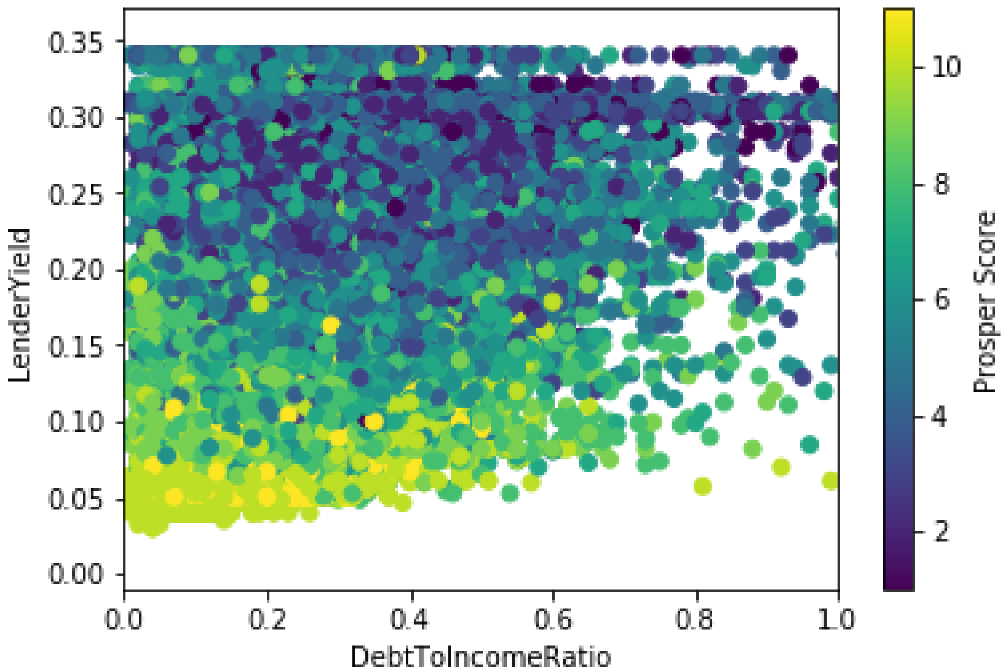
In [55]:
```
# Check the Multivariate Relationship between BorrowerRate and BorrowerAPR
plt.scatter(data = df_loan, x = 'BorrowerRate', y = 'BorrowerAPR',
                c = 'ProsperScore')
plt.colorbar(label = 'Prosper Score')
plt.xlim(df_loan['BorrowerRate'].min())
plt.ylim( df_loan['BorrowerAPR'].min())
plt.xlabel('BorrowerRate')
plt.ylabel('BorrowerAPR');
```



Here, we can see that both BorrowerRate and BorrowerAPR has a directly proportional relationship in terms of prospper score. With a low BorrowerRate and low BorrowerAPR has a high ProsperScore while high borrowerRate and low BorrowerAPR has a low ProsperScore

This suggest that the custom risk score which is called ProsperScore has been used in order to determine the rate borrower can give

In [63]:
```
#Check the Multivariate Relationship between DebtToIncomeRatio and Lender Yield
plt.scatter(data = df_loan, x = 'DebtToIncomeRatio', y = 'LenderYield',
                c = 'ProsperScore')
plt.colorbar(label = 'Prosper Score')
plt.xlim(df_loan['DebtToIncomeRatio'].min(),1.0)
plt.ylim( df_loan['LenderYield'].min())
plt.xlabel('DebtToIncomeRatio')
plt.ylabel('LenderYield');
```



Here, we can see that a low Debt to Income Ratio has a low LenderYield because it has a high ProsperScore(deemed less risky). We can also see that a high DebtToIncomeRatio has a high LenderYield because it has a low ProsperScore(deemed more risky)

```
In [ ]:
```