Part_I_exploration_template

November 6, 2022

1 Part I - (Prosper Loan Data Dataset Exploration)

1.1 by (Ahmed Abdelhady Saleh)

1.2 Introduction

Introduce the dataset

This data set contains 113,937 loans with 81 variables on each loan, including loan amount, borrower rate (or interest rate), current loan status, borrower income, and many others. I will analyze this data and find out which factors affect the loans and create a complete analysis of all available items.

> **Rubric Tip**: Your code should not generate any errors, and should use functions, loops where possible to reduce repetitive code. Prefer to use functions to reuse code statements.

> **Rubric Tip**: Document your approach and findings in markdown cells. Use comments and docstrings in code cells to document the code functionality.

> **Rubric Tip**: Markup cells should have headers and text that organize your thoughts, findings, and what you plan on investigating next.

1.3 Preliminary Wrangling

In [49]:
```
# import all packages and set plots to be embedded inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

%matplotlib inline
```

Load in your dataset and describe its properties through the questions below. Try and motivate your exploration goals through this section.

In [50]:
```
#load the dataset
df=pd.read_csv('prosperLoanData.csv')
df.head()
```

Out[50]:
```
                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

  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

   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

   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

   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

  InvestmentFromFriendsAmount Investors
0                         0.0       258
1                         0.0         1
2                         0.0        41
3                         0.0       158
4                         0.0        20

[5 rows x 81 columns]
```

In [51]:
```
df.shape
```

Out[51]:
```
(113937, 81)
```

In [52]:
```
df.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
```

In [53]:
```
df.describe()
```

Out[53]:
```
       ListingNumber           Term    BorrowerAPR   BorrowerRate  \
count   1.139370e+05  113937.000000  113912.000000  113937.000000
mean    6.278857e+05      40.830248       0.218828       0.192764
std     3.280762e+05      10.436212       0.080364       0.074818
min     4.000000e+00      12.000000       0.006530       0.000000
25%     4.009190e+05      36.000000       0.156290       0.134000
```

50%	6.005540e+05	36.000000	0.209760	0.184000
75%	8.926340e+05	36.000000	0.283810	0.250000
max	1.255725e+06	60.000000	0.512290	0.497500

	LenderYield	EstimatedEffectiveYield	EstimatedLoss	EstimatedReturn \
count	113937.000000	84853.000000	84853.000000	84853.000000
mean	0.182701	0.168661	0.080306	0.096068
std	0.074516	0.068467	0.046764	0.030403
min	-0.010000	-0.182700	0.004900	-0.182700
25%	0.124200	0.115670	0.042400	0.074080
50%	0.173000	0.161500	0.072400	0.091700
75%	0.240000	0.224300	0.112000	0.116600
max	0.492500	0.319900	0.366000	0.283700

	ProsperRating (numeric)	ProsperScore	...	LP_ServiceFees \
count	84853.000000	84853.000000	...	113937.000000
mean	4.072243	5.950067	...	-54.725641
std	1.673227	2.376501	...	60.675425
min	1.000000	1.000000	...	-664.870000
25%	3.000000	4.000000	...	-73.180000
50%	4.000000	6.000000	...	-34.440000
75%	5.000000	8.000000	...	-13.920000
max	7.000000	11.000000	...	32.060000

	LP_CollectionFees	LP_GrossPrincipalLoss	LP_NetPrincipalLoss \
count	113937.000000	113937.000000	113937.000000
mean	-14.242698	700.446342	681.420499
std	109.232758	2388.513831	2357.167068
min	-9274.750000	-94.200000	-954.550000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	0.000000	25000.000000	25000.000000

	LP_NonPrincipalRecoverypayments	PercentFunded	Recommendations \
count	113937.000000	113937.000000	113937.000000
mean	25.142686	0.998584	0.048027
std	275.657937	0.017919	0.332353
min	0.000000	0.700000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	0.000000
max	21117.900000	1.012500	39.000000

	InvestmentFromFriendsCount	InvestmentFromFriendsAmount	Investors
count	113937.000000	113937.000000	113937.000000
mean	0.023460	16.550751	80.475228
std	0.232412	294.545422	103.239020

```
min                   0.000000                  0.000000     1.000000
25%                   0.000000                  0.000000     2.000000
50%                   0.000000                  0.000000    44.000000
75%                   0.000000                  0.000000   115.000000
max                  33.000000              25000.000000  1189.000000

[8 rows x 61 columns]
```

In [54]:
```
# drop coulmns with empty values or not usefu info from the dataset
df.drop(['ListingKey', 'ListingNumber', 'ListingCreationDate', 'CreditGrade', 'ClosedDa
```

In [55]:
```
df.shape
```

Out[55]:
```
(113937, 61)
```

In [56]:
```
df.head()
```

Out[56]:
```
   Term LoanStatus  BorrowerAPR  BorrowerRate  LenderYield  \
0    36  Completed      0.16516        0.1580       0.1380
1    36    Current      0.12016        0.0920       0.0820
2    36  Completed      0.28269        0.2750       0.2400
3    36    Current      0.12528        0.0974       0.0874
4    36    Current      0.24614        0.2085       0.1985

   EstimatedEffectiveYield  EstimatedLoss  EstimatedReturn  \
0                      NaN            NaN              NaN
1                  0.07960         0.0249          0.05470
2                      NaN            NaN              NaN
3                  0.08490         0.0249          0.06000
4                  0.18316         0.0925          0.09066

   ProsperRating (numeric) ProsperRating (Alpha)    ...      LP_ServiceFees  \
0                      NaN                   NaN    ...             -133.18
1                      6.0                     A    ...                0.00
2                      NaN                   NaN    ...              -24.20
3                      6.0                     A    ...             -108.01
4                      3.0                     D    ...              -60.27

   LP_CollectionFees LP_GrossPrincipalLoss LP_NetPrincipalLoss  \
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

  LP_NonPrincipalRecoverypayments  PercentFunded  Recommendations  \
0                             0.0            1.0                0
1                             0.0            1.0                0
2                             0.0            1.0                0
```

```
3                             0.0            1.0                 0
4                             0.0            1.0                 0

   InvestmentFromFriendsCount  InvestmentFromFriendsAmount  Investors
0                           0                          0.0        258
1                           0                          0.0          1
2                           0                          0.0         41
3                           0                          0.0        158
4                           0                          0.0         20

[5 rows x 61 columns]
```

```
In [57]: sum(df.duplicated())
```

```
Out[57]: 0
```

```
In [58]: df.isnull().sum()
```

```
Out[58]: Term                              0
         LoanStatus                        0
         BorrowerAPR                      25
         BorrowerRate                      0
         LenderYield                       0
         EstimatedEffectiveYield       29084
         EstimatedLoss                 29084
         EstimatedReturn               29084
         ProsperRating (numeric)       29084
         ProsperRating (Alpha)         29084
         ProsperScore                  29084
         ListingCategory (numeric)         0
         BorrowerState                  5515
         Occupation                     3588
         EmploymentStatus               2255
         EmploymentStatusDuration       7625
         IsBorrowerHomeowner               0
         CreditScoreRangeLower           591
         CreditScoreRangeUpper           591
         CurrentCreditLines             7604
         OpenCreditLines                7604
         TotalCreditLinespast7years      697
         OpenRevolvingAccounts             0
         OpenRevolvingMonthlyPayment       0
         InquiriesLast6Months            697
         TotalInquiries                 1159
         CurrentDelinquencies            697
         AmountDelinquent               7622
         DelinquenciesLast7Years          990
         PublicRecordsLast10Years         697
                                       ...
```

```
RevolvingCreditBalance                 7604
BankcardUtilization                    7604
AvailableBankcardCredit                7544
TotalTrades                            7544
TradesNeverDelinquent (percentage)     7544
TradesOpenedLast6Months                7544
DebtToIncomeRatio                      8554
IncomeRange                               0
IncomeVerifiable                          0
StatedMonthlyIncome                       0
LoanMonthsSinceOrigination                0
LoanNumber                                0
LoanOriginalAmount                        0
LoanOriginationDate                       0
LoanOriginationQuarter                    0
MemberKey                                 0
MonthlyLoanPayment                        0
LP_CustomerPayments                       0
LP_CustomerPrincipalPayments              0
LP_InterestandFees                        0
LP_ServiceFees                            0
LP_CollectionFees                         0
LP_GrossPrincipalLoss                     0
LP_NetPrincipalLoss                       0
LP_NonPrincipalRecoverypayments           0
PercentFunded                             0
Recommendations                           0
InvestmentFromFriendsCount                0
InvestmentFromFriendsAmount               0
Investors                                 0
Length: 61, dtype: int64
```

In [59]:
```
#remove loans without ProsperScores
df2 = df[df['ProsperScore'].isnull()==False]
```

In [60]:
```
df2.isnull().sum()
```

Out[60]:
```
Term                                    0
LoanStatus                              0
BorrowerAPR                             0
BorrowerRate                            0
LenderYield                             0
EstimatedEffectiveYield                 0
EstimatedLoss                           0
EstimatedReturn                         0
ProsperRating (numeric)                 0
ProsperRating (Alpha)                   0
ProsperScore                            0
```

```
ListingCategory (numeric)              0
BorrowerState                          0
Occupation                          1333
EmploymentStatus                       0
EmploymentStatusDuration              19
IsBorrowerHomeowner                    0
CreditScoreRangeLower                  0
CreditScoreRangeUpper                  0
CurrentCreditLines                     0
OpenCreditLines                        0
TotalCreditLinespast7years             0
OpenRevolvingAccounts                  0
OpenRevolvingMonthlyPayment            0
InquiriesLast6Months                   0
TotalInquiries                         0
CurrentDelinquencies                   0
AmountDelinquent                       0
DelinquenciesLast7Years                0
PublicRecordsLast10Years               0
                                     ...
RevolvingCreditBalance                 0
BankcardUtilization                    0
AvailableBankcardCredit                0
TotalTrades                            0
TradesNeverDelinquent (percentage)     0
TradesOpenedLast6Months                0
DebtToIncomeRatio                   7296
IncomeRange                            0
IncomeVerifiable                       0
StatedMonthlyIncome                    0
LoanMonthsSinceOrigination             0
LoanNumber                             0
LoanOriginalAmount                     0
LoanOriginationDate                    0
LoanOriginationQuarter                 0
MemberKey                              0
MonthlyLoanPayment                     0
LP_CustomerPayments                    0
LP_CustomerPrincipalPayments           0
LP_InterestandFees                     0
LP_ServiceFees                         0
LP_CollectionFees                      0
LP_GrossPrincipalLoss                  0
LP_NetPrincipalLoss                    0
LP_NonPrincipalRecoverypayments        0
PercentFunded                          0
Recommendations                        0
InvestmentFromFriendsCount             0
```

```
InvestmentFromFriendsAmount            0
Investors                              0
Length: 61, dtype: int64
```

1.3.1 What is the structure of your dataset?

Loan dataset of data from 81 columns and 113,937 people.

Each row is someone who has loan with 81 attributes , i will go far with this great data.

1.3.2 What is/are the main feature(s) of interest in your dataset?

Maybe the Borrower APR and the interest rate beside to factors like score, occupation and income where these factors can change each other.

1.3.3 What features in the dataset do you think will help support your investigation into your feature(s) of interest?

The Prosper Rating and Prosperscore have impact on Borrower's APR due to high rating reflects the nature of the person who will borrow and Creditscore can also have an impact on orrower's APR.

1.4 Univariate Exploration

In this section, investigate distributions of individual variables. If you see unusual points or outliers, take a deeper look to clean things up and prepare yourself to look at relationships between variables.

Rubric Tip: The project (Parts I alone) should have at least 15 visualizations distributed over univariate, bivariate, and multivariate plots to explore many relationships in the data set. Use reasoning to justify the flow of the exploration.

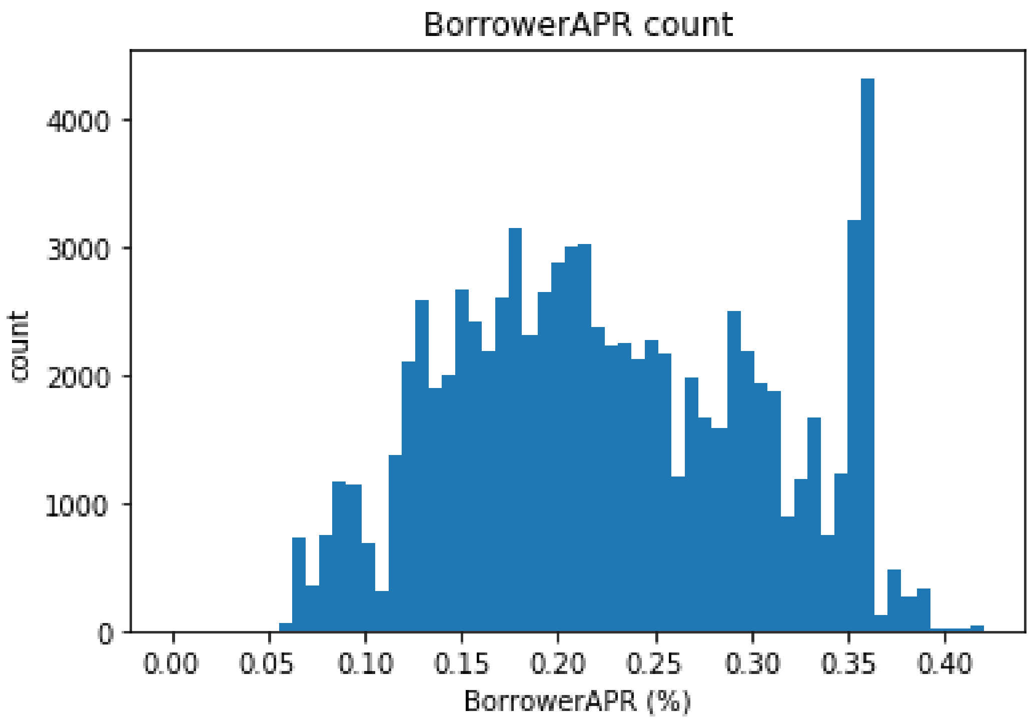
Rubric Tip: Use the "Question-Visualization-Observations" framework throughout the exploration. This framework involves **asking a question from the data, creating a visualization to find answers, and then recording observations after each visualisation.**

In [61]:
```
base_color = sb.color_palette()[0];
```

In [62]:
```
#see what is BorrowerAPR most count
df2.BorrowerAPR.value_counts().head()
```

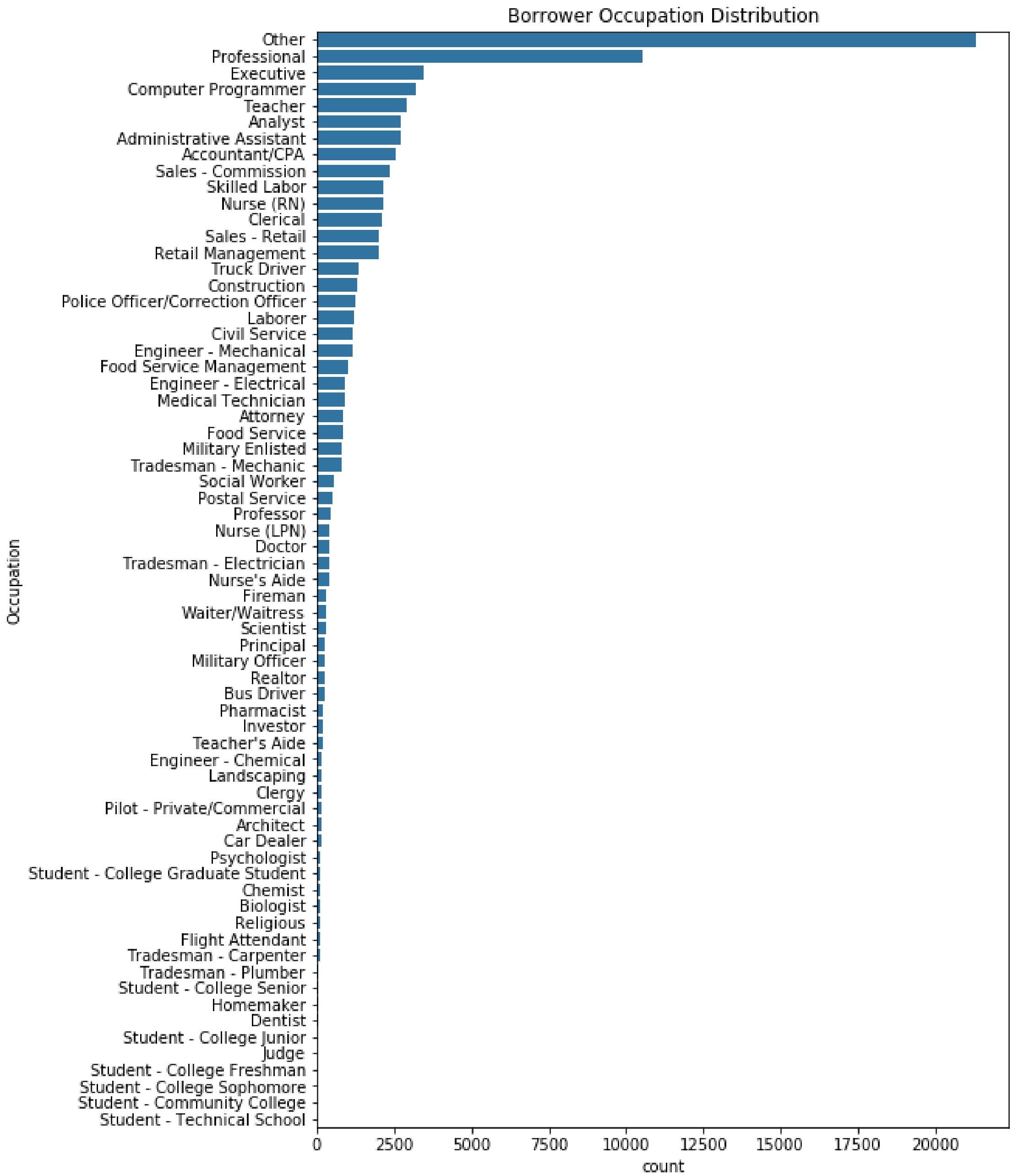
Out[62]:
```
0.35797    3672
0.35643    1644
0.30532     902
0.29510     747
0.35356     721
Name: BorrowerAPR, dtype: int64
```

```
In [63]: #counts for all BorrowerAPR values
         bins = np.arange(0, df2['BorrowerAPR'].max(), 0.007)
         plt.hist(data = df2, x = 'BorrowerAPR', bins = bins)
         plt.title('BorrowerAPR count')
         plt.xlabel('BorrowerAPR (%)')
         plt.ylabel('count')
         plt.xticks(np.arange(0, df2['BorrowerAPR'].max(), 0.05));
```



Distribution is considered normal except in a specific period in which the distribution is interesting, which is the period when the value of BorrowerAPR between0.35797% and 0.35643%.

```
In [64]: #top occupations of borrowers
         order = df2.Occupation.value_counts().index
         plt.figure(figsize=[8, 13])
         sb.countplot(data=df2,y='Occupation',color=base_color, order=order);
         plt.title('Borrower Occupation Distribution');
```



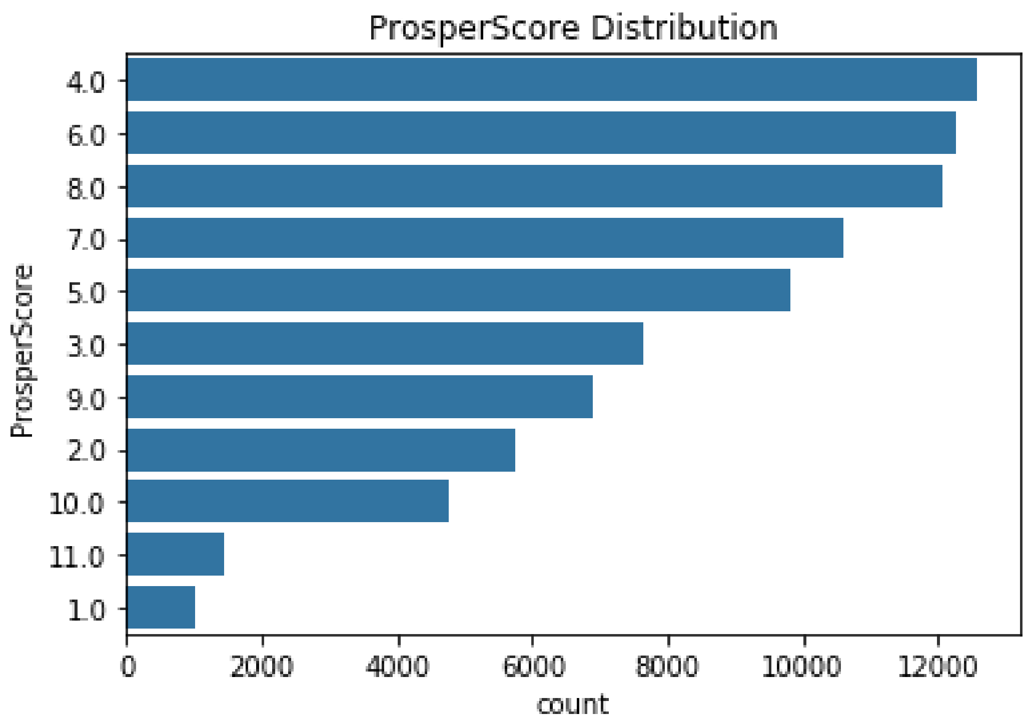
We note that most loan applicants said that the professions are others & professional due to they often do not want to share this information and may have said that they are professionals to enhance their chances of obtaining a loan

```
In [65]: df2.ProsperScore.value_counts()

Out[65]: 4.0       12595
         6.0       12278
         8.0       12053
```

```
7.0     10597
5.0      9813
3.0      7642
9.0      6911
2.0      5766
10.0     4750
11.0     1456
1.0       992
Name: ProsperScore, dtype: int64
```

In [66]:
```
#Prosper Score Distribution
def fn(df2,c):
 order = df2[c].value_counts().index
 sb.countplot(data=df2, y=df2[c], color=base_color, order=order);
 plt.ylabel(c);
 plt.title(c+ ' Distribution');
print(fn(df2,'ProsperScore'))
```

```
None
```



the less prosper score for borrowers, the less the loan they got.

In [67]:
```
#see what the loan original amount and what it's like.
df2['LoanOriginalAmount'].value_counts()
```

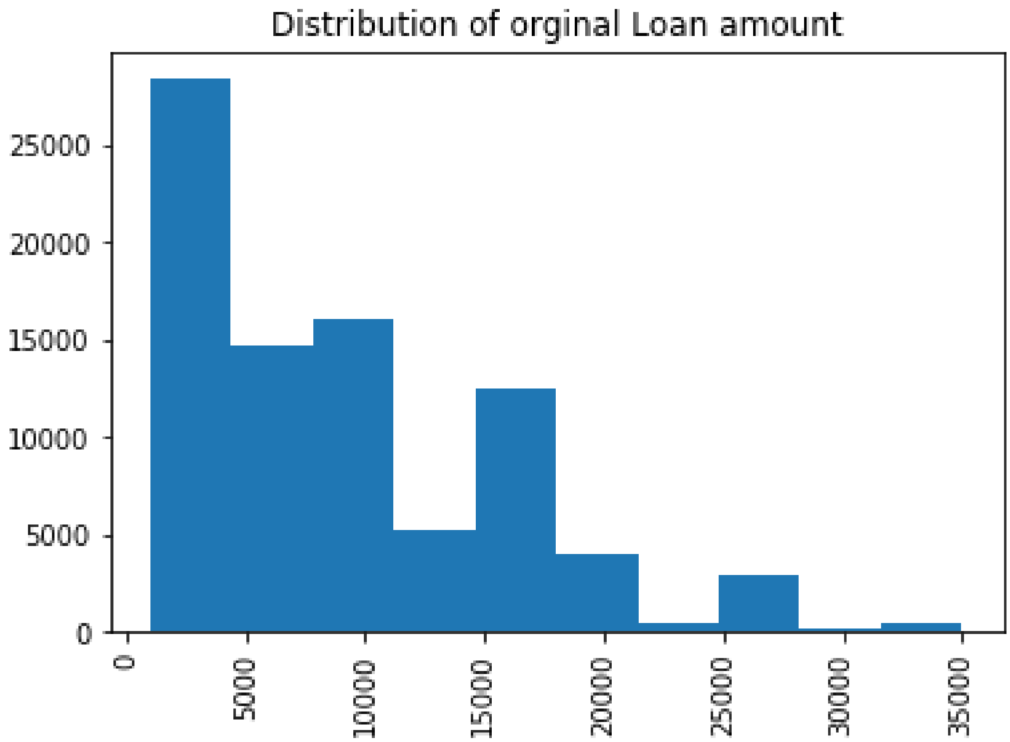
```
Out[67]: 4000      13233
         15000     11460
         10000      9816
         2000       4591
         5000       4224
         3000       3451
         20000      2928
         25000      2788
         7000       2292
         7500       2177
         6000       2043
         8000       1899
         2500       1848
         3500       1674
         12000      1571
         13000      1388
         9000       1312
         6500       1000
         4500        973
         8500        865
         1000        761
         11000       649
         5500        641
         14000       528
         11500       524
         9500        493
         1500        475
         35000       430
         18000       396
         12500       346
                   ...
         3369          1
         5546          1
         11821         1
         34679         1
         16160         1
         13792         1
         10914         1
         7845          1
         12003         1
         3879          1
         12131         1
         14304         1
         4133          1
         18542         1
         10466         1
         6564          1
         6692          1
```

```
6948         1
24300        1
11170        1
5029         1
7268         1
9443         1
5349         1
7524         1
19950        1
1575         1
5733         1
11938        1
8196         1
Name: LoanOriginalAmount, Length: 1934, dtype: int64
```

In [68]:
```
df2['LoanOriginalAmount'].describe()
```

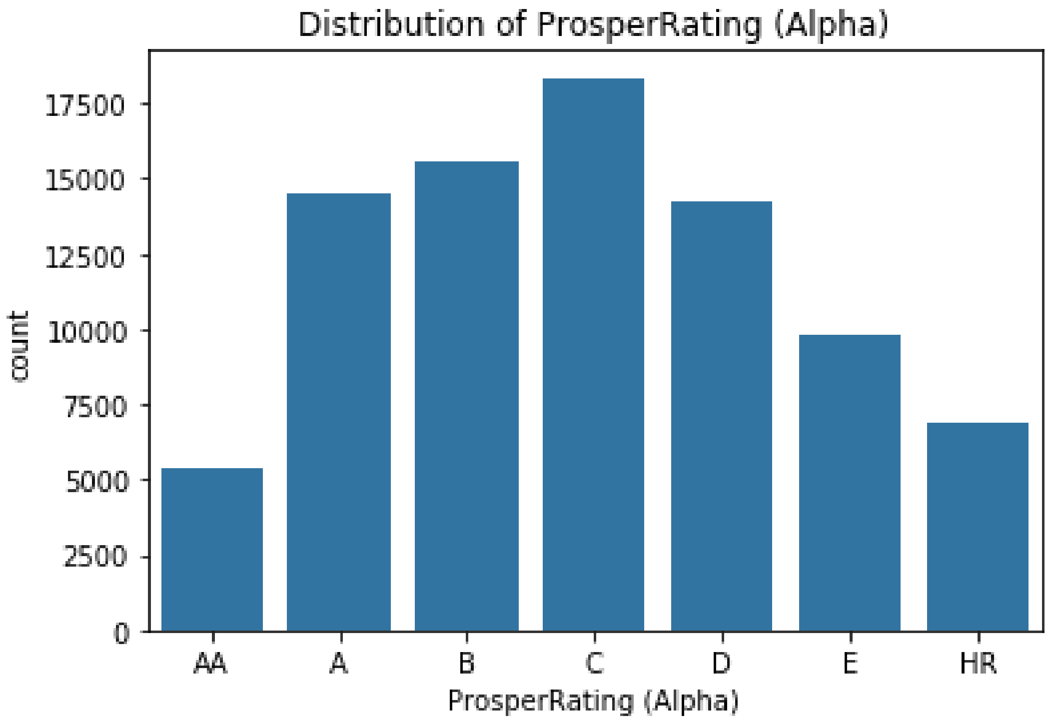
Out[68]:
```
count    84853.000000
mean      9083.440515
std       6287.860058
min       1000.000000
25%       4000.000000
50%       7500.000000
75%      13500.000000
max      35000.000000
Name: LoanOriginalAmount, dtype: float64
```

In [69]:
```
# Distribution of orginal Loan amount
plt.hist(data=df2,x='LoanOriginalAmount',color=base_color);
plt.title('Distribution of orginal Loan amount')
plt.xticks(rotation=90);
```

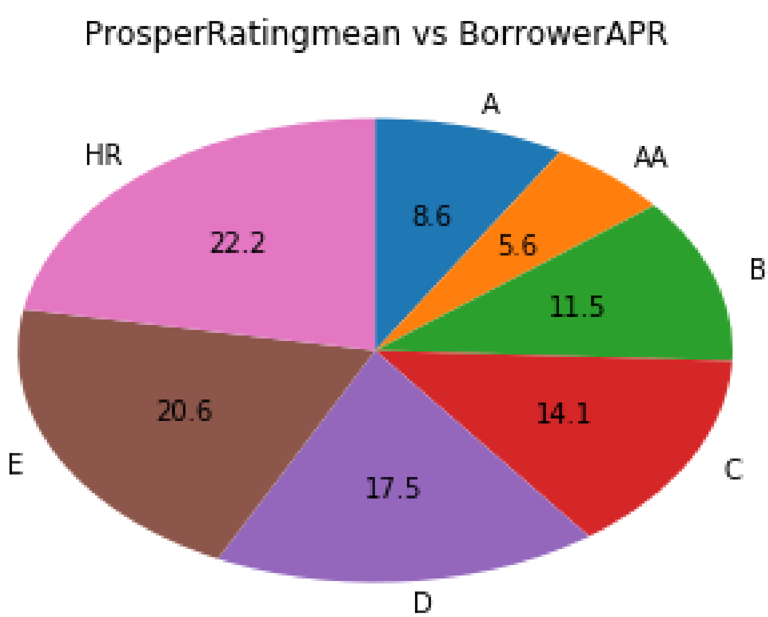


The distribution is normal , other than some points, like 10000 and 15000.

```
In [70]: #Distribution of ProsperRating_mean
         df2['ProsperRating (Alpha)'].value_counts()

Out[70]: C      18345
         B      15581
         A      14551
         D      14274
         E       9795
         HR      6935
         AA      5372
         Name: ProsperRating (Alpha), dtype: int64

In [89]: # Distribution of Prosper rating
         plt.title('Distribution of ProsperRating (Alpha)')
         sb.countplot(data = df2, x = 'ProsperRating (Alpha)', color = base_color)

Out[89]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb175eca6a0>
```




```
In [73]: ProsperRatmean  = df2.groupby('ProsperRating (Alpha)').BorrowerAPR.mean()

In [74]: plt.pie(ProsperRatmean, labels = ProsperRatmean.index, startangle = 90,
             counterclock = False, autopct='%1.1f');
         plt.title(' ProsperRatingmean vs BorrowerAPR ');
```

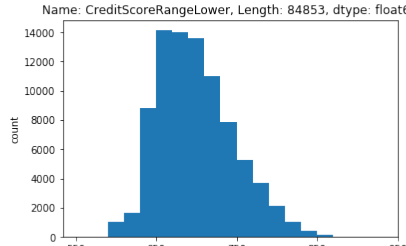


The HR appears to be the highest rated with a score of 22.2 and the lowest in the aa rating at 5.6

```
In [75]: # Histogram for Credit Score ranges for CreditScoreRangeLower
         def fun(df2,c):
          bins = np.arange(550, df2[c].max(), 20)
          plt.hist(data = df2, x = df2[c], bins = bins)
          plt.xticks(np.arange(550, 1000, 100))
          plt.title(df2[c])
          plt.xlabel(df2[c])
          plt.ylabel('count')
         print(fun(df2,'CreditScoreRangeLower'))
```

```
None
```

```
1         680.0
3         800.0
4         680.0
5         740.0
6         680.0
7         700.0
8         820.0
9         820.0
10        640.0
12        680.0
13        740.0
14        740.0
15        700.0
16        640.0
18        740.0
19        680.0
20        660.0
22        700.0
23        680.0
24        660.0
25        680.0
26        660.0
27        700.0
28        720.0
29        740.0
30        740.0
31        680.0
32        660.0
33        760.0
34        640.0
          ...
113894    800.0
113895    660.0
113898    720.0
113899    760.0
113900    680.0
113901    720.0
113903    700.0
113905    740.0
113907    640.0
113908    700.0
113909    800.0
113910    640.0
113911    660.0
113912    800.0
113913    780.0
113916    660.0
113917    660.0
113919    680.0
113920    740.0
113924    640.0
113925    680.0
113928    740.0
113929    660.0
113930    680.0
113931    800.0
113932    700.0
113933    700.0
113934    700.0
113935    680.0
113936    680.0
Name: CreditScoreRangeLower, Length: 84853, dtype: float64
```

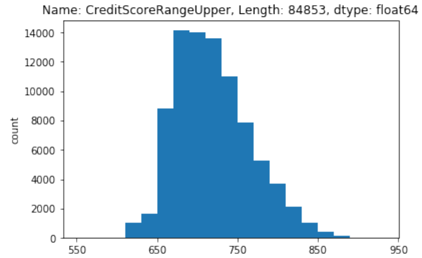



```
1         680.0
3         800.0
4         680.0
5         740.0
6         680.0
7         700.0
8         820.0
9         820.0
10        640.0
12        680.0
13        740.0
14        740.0
15        700.0
16        640.0
18        740.0
19        680.0
20        660.0
22        700.0
23        680.0
24        660.0
25        680.0
26        660.0
27        700.0
28        720.0
29        740.0
30        740.0
31        680.0
32        660.0
33        760.0
34        640.0
          ...
113894    800.0
113895    660.0
113898    720.0
113899    760.0
113900    680.0
113901    720.0
113903    700.0
113905    740.0
113907    640.0
113908    700.0
113909    800.0
113910    640.0
113911    660.0
113912    800.0
113913    780.0
113916    660.0
113917    660.0
113919    680.0
113920    740.0
113924    640.0
113925    680.0
113928    740.0
113929    660.0
113930    680.0
113931    800.0
113932    700.0
113933    700.0
113934    700.0
113935    680.0
113936    680.0
Name: CreditScoreRangeLower, Length: 84853, dtype: float64
```

In [76]:
```
# Histogram for Credit Score ranges for CreditScoreRangeUpper
print(fun(df2,'CreditScoreRangeUpper'))
```

```
None
```

```
1         699.0
3         819.0
4         699.0
5         759.0
6         699.0
7         719.0
8         839.0
9         839.0
10        659.0
12        699.0
13        759.0
14        759.0
15        719.0
16        659.0
18        759.0
19        699.0
20        679.0
22        719.0
23        699.0
24        679.0
25        699.0
26        679.0
27        719.0
28        739.0
29        759.0
30        759.0
31        699.0
32        679.0
33        779.0
34        659.0
          ...
113894    819.0
113895    679.0
113898    739.0
113899    779.0
113900    699.0
113901    739.0
113903    719.0
113905    759.0
113907    659.0
113908    719.0
113909    819.0
113910    659.0
113911    679.0
113912    819.0
113913    799.0
113916    679.0
113917    679.0
113919    699.0
113920    759.0
113924    659.0
113925    699.0
113928    759.0
113929    679.0
113930    699.0
113931    819.0
113932    719.0
113933    719.0
113934    719.0
113935    699.0
113936    699.0
Name: CreditScoreRangeUpper, Length: 84853, dtype: float64
```




```
1         699.0
3         819.0
4         699.0
5         759.0
6         699.0
7         719.0
8         839.0
9         839.0
10        659.0
12        699.0
13        759.0
14        759.0
15        719.0
16        659.0
18        759.0
19        699.0
20        679.0
22        719.0
23        699.0
24        679.0
25        699.0
26        679.0
27        719.0
28        739.0
29        759.0
30        759.0
31        699.0
32        679.0
33        779.0
34        659.0
          ...
113894    819.0
113895    679.0
113898    739.0
113899    779.0
113900    699.0
113901    739.0
113903    719.0
113905    759.0
113907    659.0
113908    719.0
113909    819.0
113910    659.0
113911    679.0
113912    819.0
113913    799.0
113916    679.0
113917    679.0
113919    699.0
113920    759.0
113924    659.0
113925    699.0
113928    759.0
113929    679.0
113930    699.0
113931    819.0
113932    719.0
113933    719.0
113934    719.0
113935    699.0
113936    699.0
Name: CreditScoreRangeUpper, Length: 84853, dtype: float64
```

Both Histograms are very similar

```
In [77]: #see Investors feature and hist
         df['Investors'].value_counts()
```

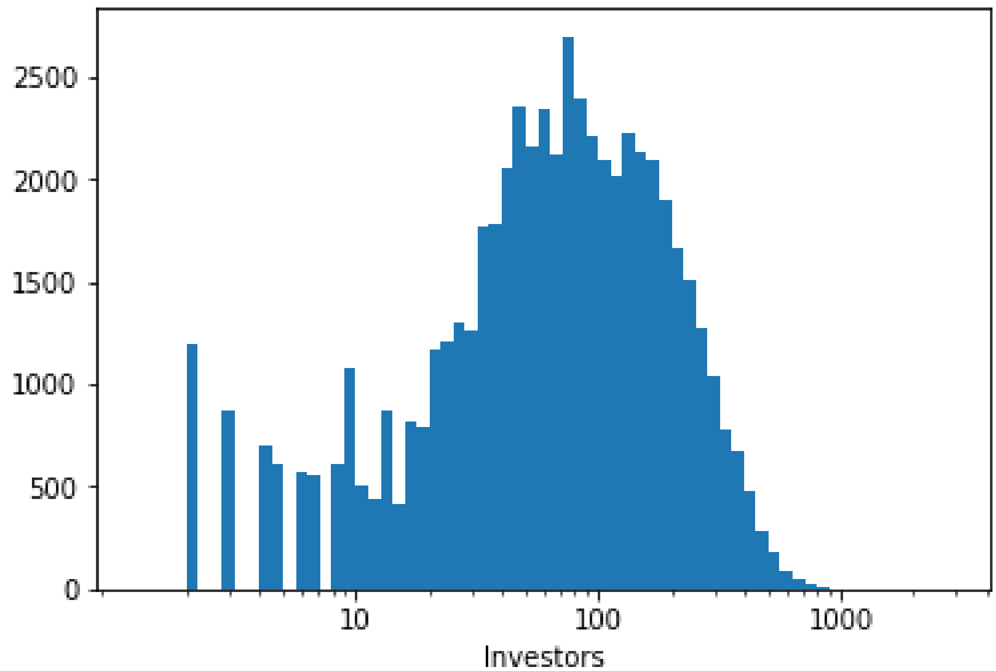
```
Out[77]: 1       27814
         2        1386
         3         991
         4         827
         5         753
         8         753
         10        728
         6         721
         9         721
         11        717
         7         701
         34        701
         13        700
         33        696
         27        683
         37        681
         25        674
         29        671
         26        668
         31        665
         21        664
         14        661
         35        661
         24        661
         17        657
         39        652
         19        652
         38        650
         30        650
         23        649
                 ...
         1035        1
         779         1
         735         1
         863         1
         645         1
         695         1
         856         1
         630         1
         838         1
         821         1
```

```
693       1
647       1
711       1
692       1
800       1
1011      1
840       1
819       1
755       1
691       1
627       1
818       1
754       1
690       1
609       1
881       1
801       1
752       1
715       1
831       1
Name: Investors, Length: 751, dtype: int64
```

```
In [78]: df2['Investors'].describe()
```

```
Out[78]: count    84853.000000
         mean        68.264669
         std         95.195831
         min          1.000000
         25%          1.000000
         50%         32.000000
         75%         97.000000
         max       1189.000000
         Name: Investors, dtype: float64
```

```
In [79]: bins = 10 ** np.arange(0.1, 3.5,0.05)
         plt.hist(df2['Investors'], bins = bins);
         plt.xscale('log')
         plt.xticks([1e1, 1e2, 1e3], ['10', '100', '1000'])
         plt.xlabel('Investors')
         plt.show()
```



the distribution has values are significantly skewed, almost a natural distribution with some skewed from the right

Rubric Tip: Visualizations should depict the data appropriately so that the plots are easily interpretable. You should choose an appropriate plot type, data encodings, and formatting as needed. The formatting may include setting/adding the title, labels, legend, and comments. Also, do not overplot or incorrectly plot ordinal data.

1.4.1 Discuss the distribution(s) of your variable(s) of interest. Were there any unusual points? Did you need to perform any transformations?

Distribution is normal , especially related to upper and lower credits. And there is some discrimination in the owner's job and hisBorrowerAPR and Looking at the BorrowerAPR count, two BorrowerAPR counts were higher than the rest of the values. Owing to the large number of counts falling to these two values, these two values can be used for resonable purposes. The two BorrowerAPR values are also left unchanged.

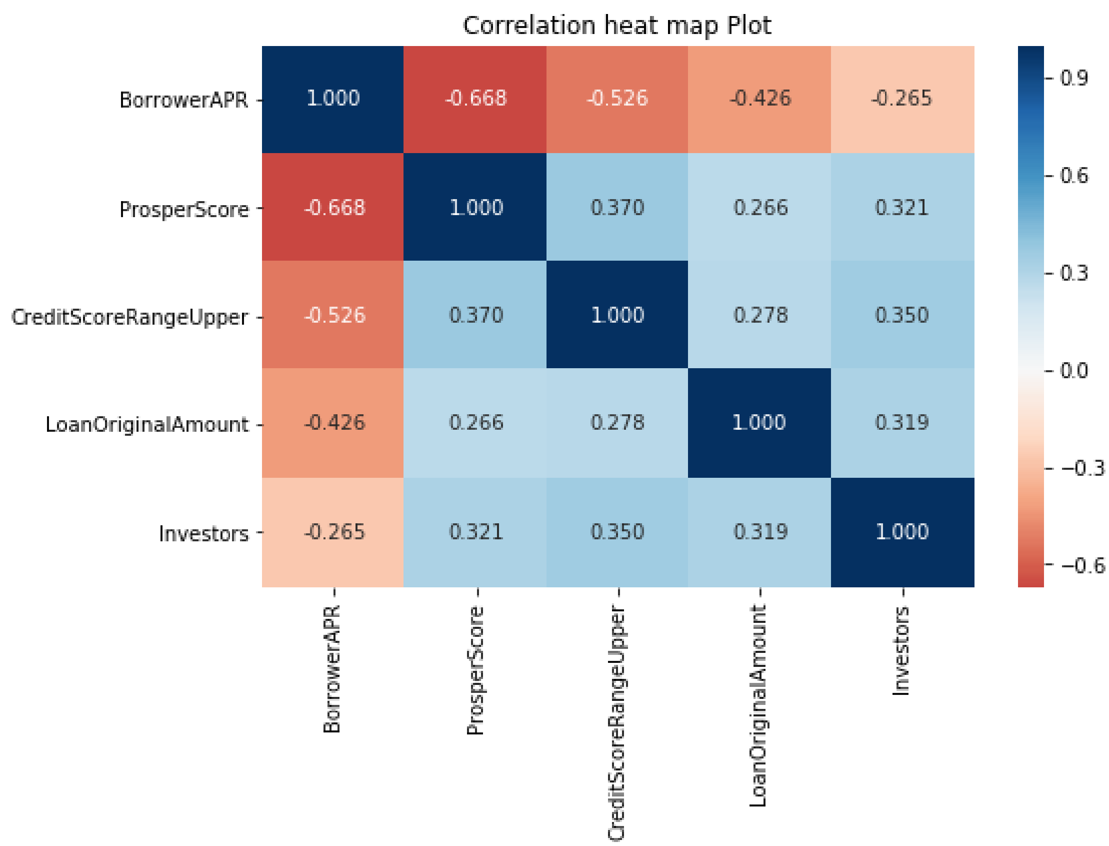
1.4.2 Of the features you investigated, were there any unusual distributions? Did you perform any operations on the data to tidy, adjust, or change the form of the data? If so, why did you do this?

many features have perverted distributions and a long tail, and have been examined on a logarithmic scale revealing the hidden distribution in some points.

1.5 Bivariate Exploration

In this section, investigate relationships between pairs of variables in your data. Make sure the variables that you cover here have been introduced in some fashion in the previous section (univariate exploration).

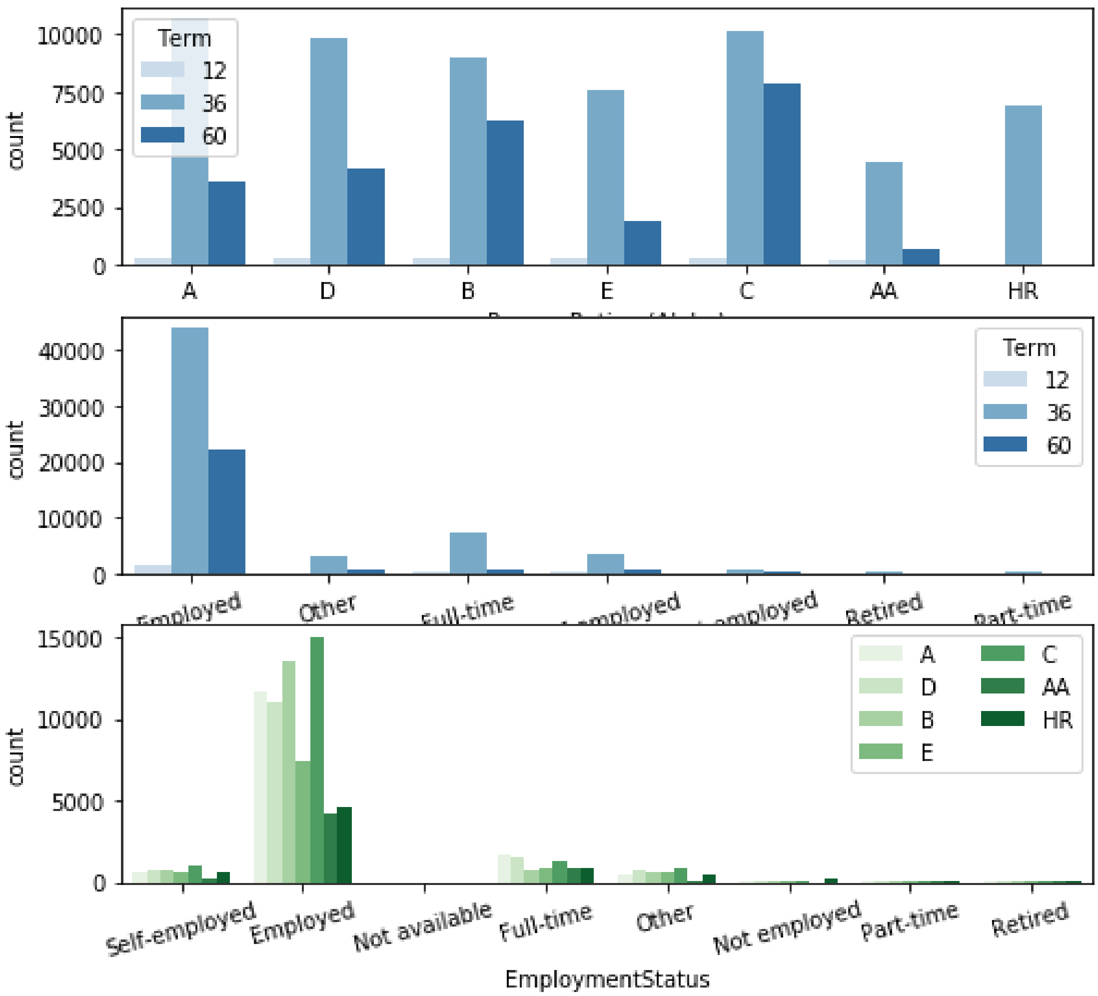
In [80]:
```
num_vars = ['BorrowerAPR', 'ProsperScore',
            'CreditScoreRangeUpper','LoanOriginalAmount','Investors']
plt.figure(figsize = [8, 5])
sb.heatmap(df2[num_vars].corr(), annot = True, fmt = '.3f',
           cmap = 'RdBu', center = 0)
plt.title('Correlation heat map Plot')
plt.show()
```



Correlation heat map Plot There is weak reverse relationship between BorrowerAPR and Investors and a relationship between BorrowerAPR and ProsperScore, CreditScoreRangeUpper and LoanOriginalAmount makes sense. because borrowers with lower score are more likely to pay higher APR next:scatter and heat plot for comparing BorrowerAPR and ProsperScore

In [81]:
```
plt.figure(figsize = [8, 10])

# Prosper rating vs term
```

```
plt.subplot(4, 1, 1)
sb.countplot(data = df2, x = 'ProsperRating (Alpha)', hue = 'Term', palette = 'Blues')

# employment status vs. term
ax = plt.subplot(4, 1, 2)
sb.countplot(data = df2, x = 'EmploymentStatus', hue = 'Term', palette = 'Blues')
plt.xticks(rotation = 15)


# Prosper rating vs. employment status, use different color palette
ax = plt.subplot(4, 1, 3)
sb.countplot(data = df, x = 'EmploymentStatus', hue = 'ProsperRating (Alpha)', palette
ax.legend(loc = 1, ncol = 2);
plt.xticks(rotation = 15);
```

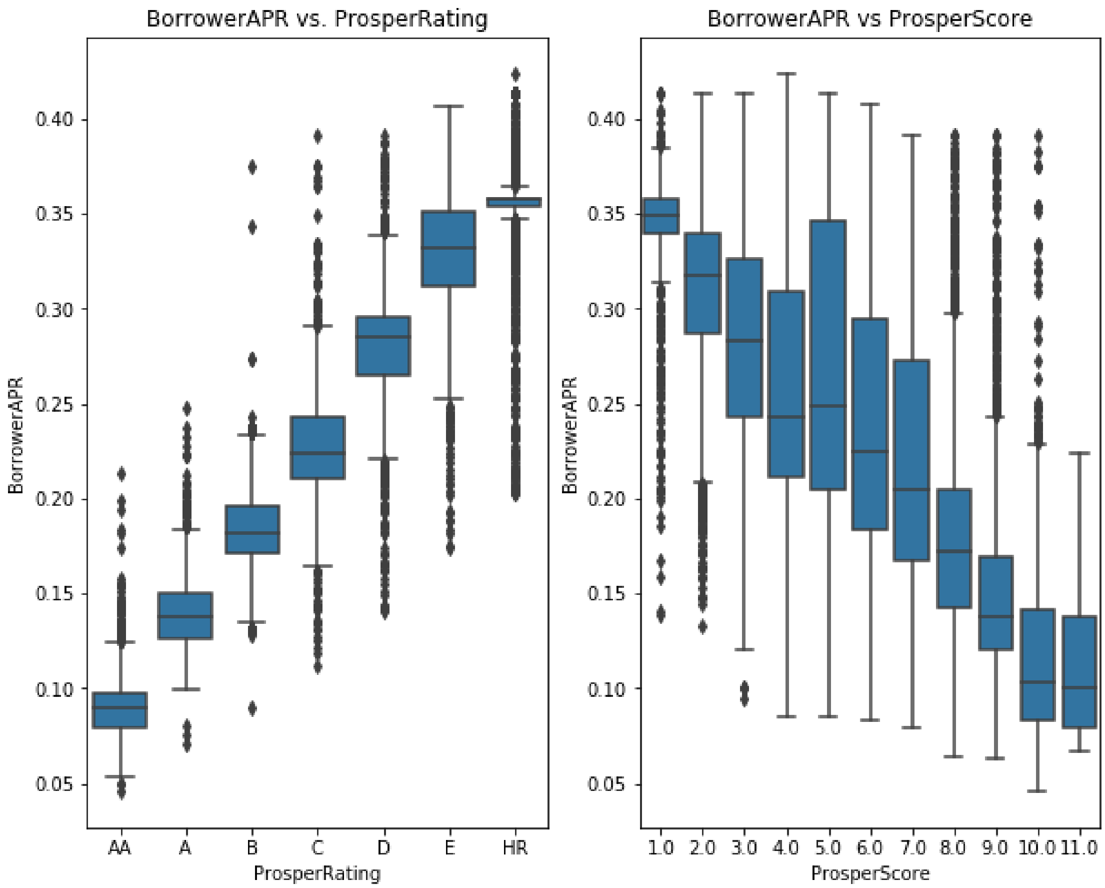


visuals show the compasiron between Prosper rating vs term, employment status vs. term, # Prosper rating vs. employment status with differ colors

In [82]:
```
rating_order = ['AA', 'A',  'B', 'C', 'D', 'E', 'HR']
# create ordered categorical variable
df2['ProsperRating (Alpha)'] = pd.Categorical(df2['ProsperRating (Alpha)'],
                                              categories= rating_order,
                                              ordered = True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """
```

In [83]:
```
plt.figure(figsize = [15, 5])
plt.gcf().set_size_inches(10, 8)


plt.subplot(1, 2, 2)
sb.boxplot(data = df2, x = 'ProsperScore', y = 'BorrowerAPR',color=base_color)
plt.title('BorrowerAPR vs ProsperScore')
plt.xlabel('ProsperScore')
plt.ylabel('BorrowerAPR');

plt.subplot(1, 2, 1)
sb.boxplot(data = df2, x = 'ProsperRating (Alpha)', y = 'BorrowerAPR',color=base_color)
plt.title('BorrowerAPR vs. ProsperRating')
plt.xlabel('ProsperRating')
plt.ylabel('BorrowerAPR');
```



There is not much overlap on ProsperRating for these two categorical variables. Good or poor scores do not reflect the percentage of APR that the creditor would earn. For ProsperScore, there is obviously a negative association with BorrowerAPR

1.5.1 Talk about some of the relationships you observed in this part of the investigation. How did the feature(s) of interest vary with other features in the dataset?

Correlation and matrix plots are a great way to understand the data and the way it works. For example, BorrowerAPR has negative relationships with almost all the columns we've used. On the other hand, the ProsperScore column has positive relationships with the same columns. We also noted that ProsperScore has more correlation than BorrowerAPR, so we should pay more attention to it because it is an important point to know how the data works.

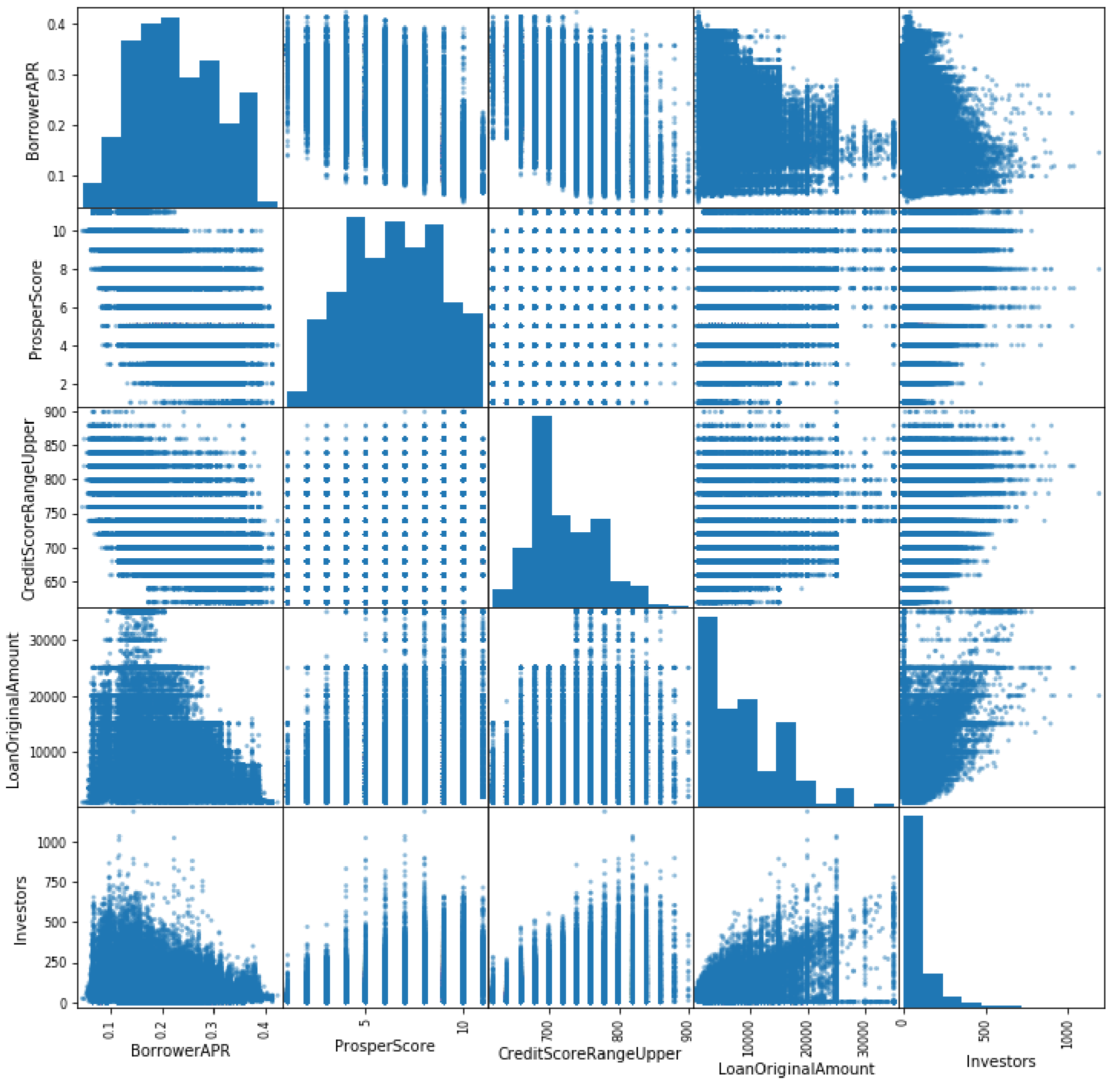
1.5.2 Did you observe any interesting relationships between the other features (not the main feature(s) of interest)?

Investors , LoanOriginalAmount and CreditScoreRangeUpper are all positive correlated to ProsperScore and negative correlated to BorrowerAPR And there's an inverse correlation between them.

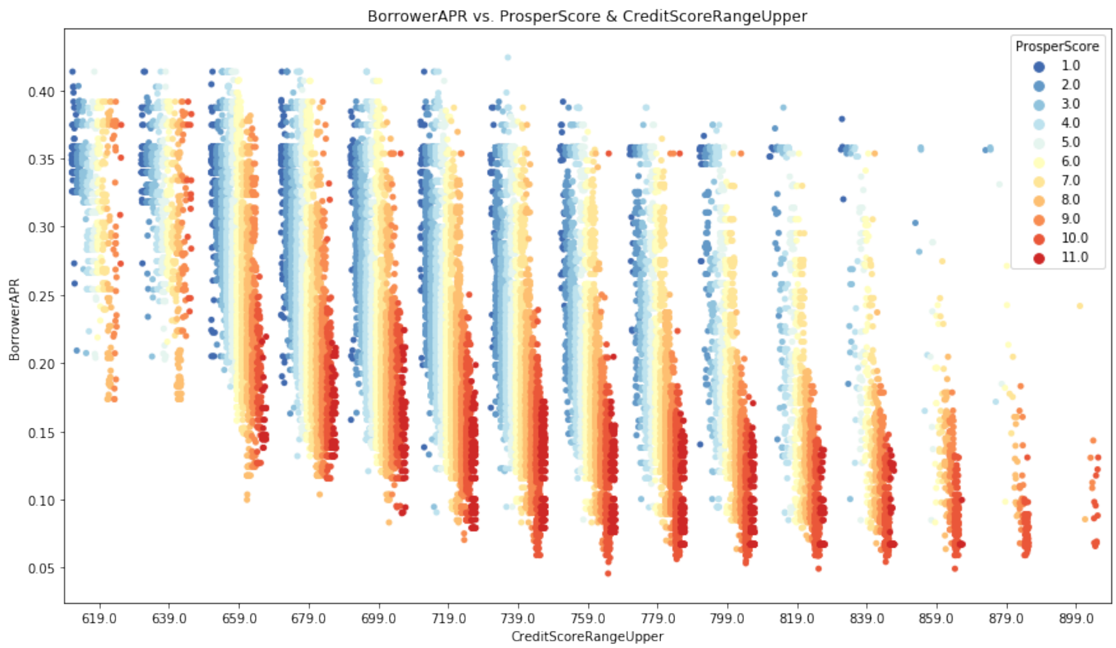
1.6 Multivariate Exploration

Create plots of three or more variables to investigate your data even further. Make sure that your investigations are justified, and follow from your work in the previous sections.

```
In [84]: pd.plotting.scatter_matrix(df2[num_vars],figsize = [12, 12]);
```




```
In [85]: plt.figure(figsize = [14.7, 8.27])
         sb.stripplot(data = df2, x = 'CreditScoreRangeUpper', y = 'BorrowerAPR', hue ='ProsperS
         plt.title('BorrowerAPR vs. ProsperScore & CreditScoreRangeUpper')
         plt.xlabel('CreditScoreRangeUpper')
         plt.ylabel('BorrowerAPR');
```



Since CreditScoreRangeUpper and ProsperScore are optimistic associated with the borrowerAPR, this visualisation allows to see the impact on the BorrowerAPR. We can see the creditScoreRangeUpper rise with the borrowerAPR fall in parcels. By adding ProsperScore to the colour encodings, the BorrowerAPR decreases as the ProsperScore rise. This proves that CreditScoreRangeUpper and ProsperScore are negative correlated with BorrowerAPR.

1.6.1 Talk about some of the relationships you observed in this part of the investigation. Were there features that strengthened each other in terms of looking at your feature(s) of interest?

we can see the extent of the correlation between the three variables and that each variable in one way or another has a role in the other variable and can be explained as explained before We can see the creditScoreRangeUpper rise with the borrowerAPR fall in parcels. By adding ProsperScore to the colour encodings, the BorrowerAPR decreases as the ProsperScore rise. This proves that CreditScoreRangeUpper and ProsperScore are adversely correlated with BorrowerAPR.

1.6.2 Were there any interesting or surprising interactions between features?

from univariate exploration to multivariate exploration, we noted that all the features are negatively correlated to BorrowerAPR, most of which were ProspoerScore, also noted that most features are positvyl correlated to prospoerScore.

1.7 Conclusions

You can write a summary of the main findings and reflect on the steps taken during the data exploration.

> Remove all Tips mentioned above, before you convert this notebook to PDF/HTML
>
> At the end of your report, make sure that you export the notebook as an html file from the `File > Download as... > HTML or PDF` menu. Make sure you keep track of where the exported file goes, so you can put it in the same folder as this notebook for project submission. Also, make sure you remove all of the quote-formatted guide notes like this one before you finish your report!

BorrowerRate increases for longer Term loans when split up by ProsperRating (Alpha). The opposite relationship would be expected as longer-term loans generally carry a lower risk profile and have a longer time to accrue interest.