Part I - (Prosper Loan Analysis)

by (Olajide Oluwatosin)

Introduction

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. See this data dictionary to understand the dataset's variables.

Preliminary Wrangling

In [1]:
```
#import necessary libraries
import warnings
warnings.filterwarnings("ignore")

import numpy as np
import pandas as pd
import missingno as mno
import matplotlib.pyplot as plt
import seaborn as sns

pd.set_option("display.max_columns",100)
pd.set_option("display.max_rows",100)

# sns.set_theme(style="white",palette=sns.color_palette("dark:grey"))
```

In [2]:
```
#import data
data = pd.read_csv("data/prosperLoanData.csv")
```

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

Out[3]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	Close
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263000000	C	36	Completed	2009- 00:
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900000000	NaN	36	Current	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090000000	HR	36	Completed	2009- 00:
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010000000	NaN	36	Current	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097000000	NaN	36	Current	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 113937 entries, 0 to 113936
Data columns (total 81 columns):
 #   Column                               Non-Null Count   Dtype
```

```
---  ------                              --------------   -----
 0   ListingKey                          113937 non-null  object
 1   ListingNumber                       113937 non-null  int64
 2   ListingCreationDate                 113937 non-null  object
 3   CreditGrade                         28953 non-null   object
 4   Term                                113937 non-null  int64
 5   LoanStatus                          113937 non-null  object
 6   ClosedDate                          55089 non-null   object
 7   BorrowerAPR                         113912 non-null  float64
 8   BorrowerRate                        113937 non-null  float64
 9   LenderYield                         113937 non-null  float64
 10  EstimatedEffectiveYield             84853 non-null   float64
 11  EstimatedLoss                       84853 non-null   float64
 12  EstimatedReturn                     84853 non-null   float64
 13  ProsperRating (numeric)             84853 non-null   float64
 14  ProsperRating (Alpha)               84853 non-null   object
 15  ProsperScore                        84853 non-null   float64
 16  ListingCategory (numeric)           113937 non-null  int64
 17  BorrowerState                       108422 non-null  object
 18  Occupation                          110349 non-null  object
 19  EmploymentStatus                    111682 non-null  object
 20  EmploymentStatusDuration            106312 non-null  float64
 21  IsBorrowerHomeowner                 113937 non-null  bool
 22  CurrentlyInGroup                    113937 non-null  bool
 23  GroupKey                            13341 non-null   object
 24  DateCreditPulled                    113937 non-null  object
 25  CreditScoreRangeLower               113346 non-null  float64
 26  CreditScoreRangeUpper               113346 non-null  float64
 27  FirstRecordedCreditLine             113240 non-null  object
 28  CurrentCreditLines                  106333 non-null  float64
 29  OpenCreditLines                     106333 non-null  float64
 30  TotalCreditLinespast7years          113240 non-null  float64
 31  OpenRevolvingAccounts               113937 non-null  int64
 32  OpenRevolvingMonthlyPayment         113937 non-null  float64
 33  InquiriesLast6Months                113240 non-null  float64
 34  TotalInquiries                      112778 non-null  float64
 35  CurrentDelinquencies                113240 non-null  float64
 36  AmountDelinquent                    106315 non-null  float64
 37  DelinquenciesLast7Years             112947 non-null  float64
 38  PublicRecordsLast10Years            113240 non-null  float64
 39  PublicRecordsLast12Months           106333 non-null  float64
 40  RevolvingCreditBalance              106333 non-null  float64
 41  BankcardUtilization                 106333 non-null  float64
 42  AvailableBankcardCredit             106393 non-null  float64
 43  TotalTrades                         106393 non-null  float64
 44  TradesNeverDelinquent (percentage)  106393 non-null  float64
 45  TradesOpenedLast6Months             106393 non-null  float64
 46  DebtToIncomeRatio                   105383 non-null  float64
 47  IncomeRange                         113937 non-null  object
 48  IncomeVerifiable                    113937 non-null  bool
 49  StatedMonthlyIncome                 113937 non-null  float64
 50  LoanKey                             113937 non-null  object
 51  TotalProsperLoans                   22085 non-null   float64
 52  TotalProsperPaymentsBilled          22085 non-null   float64
 53  OnTimeProsperPayments               22085 non-null   float64
 54  ProsperPaymentsLessThanOneMonthLate 22085 non-null   float64
 55  ProsperPaymentsOneMonthPlusLate     22085 non-null   float64
 56  ProsperPrincipalBorrowed            22085 non-null   float64
 57  ProsperPrincipalOutstanding         22085 non-null   float64
 58  ScorexChangeAtTimeOfListing         18928 non-null   float64
 59  LoanCurrentDaysDelinquent           113937 non-null  int64
 60  LoanFirstDefaultedCycleNumber       16952 non-null   float64
 61  LoanMonthsSinceOrigination          113937 non-null  int64
 62  LoanNumber                          113937 non-null  int64
 63  LoanOriginalAmount                  113937 non-null  int64
 64  LoanOriginationDate                 113937 non-null  object
```

```
 65  LoanOriginationQuarter                 113937 non-null  object
 66  MemberKey                              113937 non-null  object
 67  MonthlyLoanPayment                     113937 non-null  float64
 68  LP_CustomerPayments                    113937 non-null  float64
 69  LP_CustomerPrincipalPayments           113937 non-null  float64
 70  LP_InterestandFees                     113937 non-null  float64
 71  LP_ServiceFees                         113937 non-null  float64
 72  LP_CollectionFees                      113937 non-null  float64
 73  LP_GrossPrincipalLoss                  113937 non-null  float64
 74  LP_NetPrincipalLoss                    113937 non-null  float64
 75  LP_NonPrincipalRecoverypayments        113937 non-null  float64
 76  PercentFunded                          113937 non-null  float64
 77  Recommendations                        113937 non-null  int64
 78  InvestmentFromFriendsCount             113937 non-null  int64
 79  InvestmentFromFriendsAmount            113937 non-null  float64
 80  Investors                              113937 non-null  int64
dtypes: bool(3), float64(50), int64(11), object(17)
memory usage: 68.1+ MB
```

Date columns not in the right format

In [5]:

```
# change the datatype of date columns to datatime
date_columns = [i for i in data.columns if i[-4:] == "Date" or i[:4] == "Date"]
date_columns.append("FirstRecordedCreditLine")

for column in date_columns:
    data[column] = pd.to_datetime(data[column])

data.head()
```

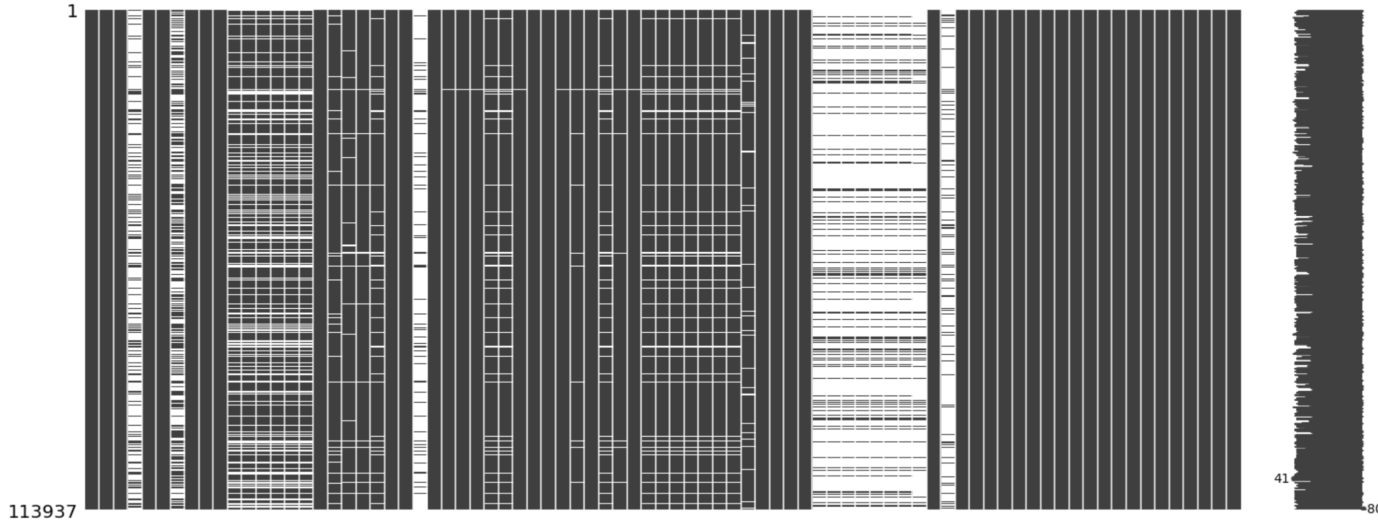
Out[5]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	Closed
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263	C	36	Completed	2009-
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900	NaN	36	Current	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090	HR	36	Completed	2009-
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010	NaN	36	Current	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097	NaN	36	Current	

Link

In [6]:

```
# visualize missing columns
mno.matrix(data);
```



From the diagram we can see different set of columns that missing values from the same row.

A lot of missing values come from columns that have consistent missing values in other columns.

In [7]:
```
data.isna().sum()/len(data)
```

Out[7]:
```
ListingKey                          0.000000
ListingNumber                       0.000000
ListingCreationDate                 0.000000
CreditGrade                         0.745886
Term                                0.000000
LoanStatus                          0.000000
ClosedDate                          0.516496
BorrowerAPR                         0.000219
BorrowerRate                        0.000000
LenderYield                         0.000000
EstimatedEffectiveYield             0.255264
EstimatedLoss                       0.255264
EstimatedReturn                     0.255264
ProsperRating (numeric)             0.255264
ProsperRating (Alpha)               0.255264
ProsperScore                        0.255264
ListingCategory (numeric)           0.000000
BorrowerState                       0.048404
Occupation                          0.031491
EmploymentStatus                    0.019792
EmploymentStatusDuration            0.066923
IsBorrowerHomeowner                 0.000000
CurrentlyInGroup                    0.000000
GroupKey                            0.882909
DateCreditPulled                    0.000000
CreditScoreRangeLower               0.005187
CreditScoreRangeUpper               0.005187
FirstRecordedCreditLine             0.006117
CurrentCreditLines                  0.066739
OpenCreditLines                     0.066739
TotalCreditLinespast7years          0.006117
OpenRevolvingAccounts               0.000000
OpenRevolvingMonthlyPayment         0.000000
InquiriesLast6Months                0.006117
TotalInquiries                      0.010172
CurrentDelinquencies                0.006117
AmountDelinquent                    0.066897
DelinquenciesLast7Years             0.008689
PublicRecordsLast10Years            0.006117
PublicRecordsLast12Months           0.066739
RevolvingCreditBalance              0.066739
```

```
BankcardUtilization                    0.066739
AvailableBankcardCredit                0.066212
TotalTrades                            0.066212
TradesNeverDelinquent (percentage)     0.066212
TradesOpenedLast6Months                0.066212
DebtToIncomeRatio                      0.075077
IncomeRange                            0.000000
IncomeVerifiable                       0.000000
StatedMonthlyIncome                    0.000000
LoanKey                                0.000000
TotalProsperLoans                      0.806165
TotalProsperPaymentsBilled             0.806165
OnTimeProsperPayments                  0.806165
ProsperPaymentsLessThanOneMonthLate    0.806165
ProsperPaymentsOneMonthPlusLate        0.806165
ProsperPrincipalBorrowed               0.806165
ProsperPrincipalOutstanding            0.806165
ScorexChangeAtTimeOfListing            0.833873
LoanCurrentDaysDelinquent              0.000000
LoanFirstDefaultedCycleNumber          0.851216
LoanMonthsSinceOrigination             0.000000
LoanNumber                             0.000000
LoanOriginalAmount                     0.000000
LoanOriginationDate                    0.000000
LoanOriginationQuarter                 0.000000
MemberKey                              0.000000
MonthlyLoanPayment                     0.000000
LP_CustomerPayments                    0.000000
LP_CustomerPrincipalPayments           0.000000
LP_InterestandFees                     0.000000
LP_ServiceFees                         0.000000
LP_CollectionFees                      0.000000
LP_GrossPrincipalLoss                  0.000000
LP_NetPrincipalLoss                    0.000000
LP_NonPrincipalRecoverypayments        0.000000
PercentFunded                          0.000000
Recommendations                        0.000000
InvestmentFromFriendsCount             0.000000
InvestmentFromFriendsAmount            0.000000
Investors                              0.000000
dtype: float64
```

In [8]:
```
data.head()
```

Out[8]:

	ListingKey	ListingNumber	ListingCreationDate	CreditGrade	Term	LoanStatus	Close
0	1021339766868145413AB3B	193129	2007-08-26 19:09:29.263	C	36	Completed	2009-
1	10273602499503308B223C1	1209647	2014-02-27 08:28:07.900	NaN	36	Current	
2	0EE9337825851032864889A	81716	2007-01-05 15:00:47.090	HR	36	Completed	2009-
3	0EF5356002482715299901A	658116	2012-10-22 11:02:35.010	NaN	36	Current	
4	0F023589499656230C5E3E2	909464	2013-09-14 18:38:39.097	NaN	36	Current	

Consider all data points after July 2009 since a quite a number of features were collected after July 2009. We will dispose all the known values for the CreditRating because only loans before 2009 had this feature. However, we can use ProperRating as a substitute to this column

In [9]:
```
data = data.query("ListingCreationDate >= '2009-08-1'")
```

In [10]:
```
(data.isna().sum()/len(data)).tail(40)
```

Out[10]:
```
BankcardUtilization                     0.000000
AvailableBankcardCredit                 0.000000
TotalTrades                             0.000000
TradesNeverDelinquent (percentage)      0.000000
TradesOpenedLast6Months                 0.000000
DebtToIncomeRatio                       0.085991
IncomeRange                             0.000000
IncomeVerifiable                        0.000000
StatedMonthlyIncome                     0.000000
LoanKey                                 0.000000
TotalProsperLoans                       0.767550
TotalProsperPaymentsBilled              0.767550
OnTimeProsperPayments                   0.767550
ProsperPaymentsLessThanOneMonthLate     0.767550
ProsperPaymentsOneMonthPlusLate         0.767550
ProsperPrincipalBorrowed                0.767550
ProsperPrincipalOutstanding             0.767550
ScorexChangeAtTimeOfListing             0.804812
LoanCurrentDaysDelinquent               0.000000
LoanFirstDefaultedCycleNumber           0.926469
LoanMonthsSinceOrigination              0.000000
LoanNumber                              0.000000
LoanOriginalAmount                      0.000000
LoanOriginationDate                     0.000000
LoanOriginationQuarter                  0.000000
MemberKey                               0.000000
MonthlyLoanPayment                      0.000000
LP_CustomerPayments                     0.000000
LP_CustomerPrincipalPayments            0.000000
LP_InterestandFees                      0.000000
LP_ServiceFees                          0.000000
LP_CollectionFees                       0.000000
LP_GrossPrincipalLoss                   0.000000
LP_NetPrincipalLoss                     0.000000
LP_NonPrincipalRecoverypayments         0.000000
PercentFunded                           0.000000
Recommendations                         0.000000
InvestmentFromFriendsCount              0.000000
InvestmentFromFriendsAmount             0.000000
Investors                               0.000000
dtype: float64
```

We could leave the `ClosingDate` column (it can be useful when analysing loans that have been closed), we will fill null values in the `Occupation` column with null, we will drop all the missing data points in `EmploymentStatusDuration` and `DebtToIncomeRatio` and fill `TotalProsperLoans` with 0.

In [11]:
```
data.Occupation.fillna("Unknown",inplace=True)

data.dropna(inplace=True,subset=["DebtToIncomeRatio","EmploymentStatusDuration"])

data.TotalProsperLoans.fillna(0,inplace=True)

data.reset_index(inplace=True,drop=True)
```

Drop columns that are duplicated or unnecessary to the analysis.

In [12]:
```
cols_to_drop = ["Term","CreditGrade","ListingKey", "ListingNumber", "ProsperRating (nume
```

```
data.drop(cols_to_drop,1,inplace=True)
```

In [13]:
```
data.isna().sum()
```

Out[13]:
```
LoanStatus                        0
ClosedDate                    54514
BorrowerAPR                       0
EstimatedLoss                     0
ProsperRating (Alpha)             0
ProsperScore                      0
ListingCategory (numeric)         0
CurrentlyInGroup                  0
CreditScoreRangeLower             0
CreditScoreRangeUpper             0
OpenCreditLines                   0
RevolvingCreditBalance            0
BankcardUtilization               0
AvailableBankcardCredit           0
DebtToIncomeRatio                 0
IncomeRange                       0
StatedMonthlyIncome               0
TotalProsperLoans                 0
LoanCurrentDaysDelinquent         0
LoanOriginalAmount                0
MonthlyLoanPayment                0
dtype: int64
```

In [14]:
```
data.LoanStatus = data.LoanStatus.apply(lambda x: "Past Due" if x[:8] == "Past Due" else
```

In [15]:
```
# drop the unemployed person since he is a single person
data.drop(data[data.IncomeRange=="Not employed"].index,inplace=True)
data.reset_index(drop=True,inplace=True)
```

What is the structure of your dataset?

Our data consists of 77376 thousand data points which include columns like 'LoanStatus', 'ClosedDate', 'BorrowerAPR', 'EstimatedLoss', 'ProsperRating (Alpha)', 'ProsperScore', 'ListingCategory (numeric)','CreditScoreRangeLower' 'OpenCreditLines''AvailableBankcardCredit', 'DebtToIncomeRatio', 'IncomeRange', 'StatedMonthlyIncome', 'TotalProsperLoans', etc

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

I want to explore loans may differ for people in different income levels.

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

The IncomeRange, StatedMonthlyIncome, LoanStatus, BorrowerAPR, OpenCreditLines, DebttoIncomeRatio, TotalProsperLoans

In [16]:
```
data.to_csv("processed.csv",index=False)
```

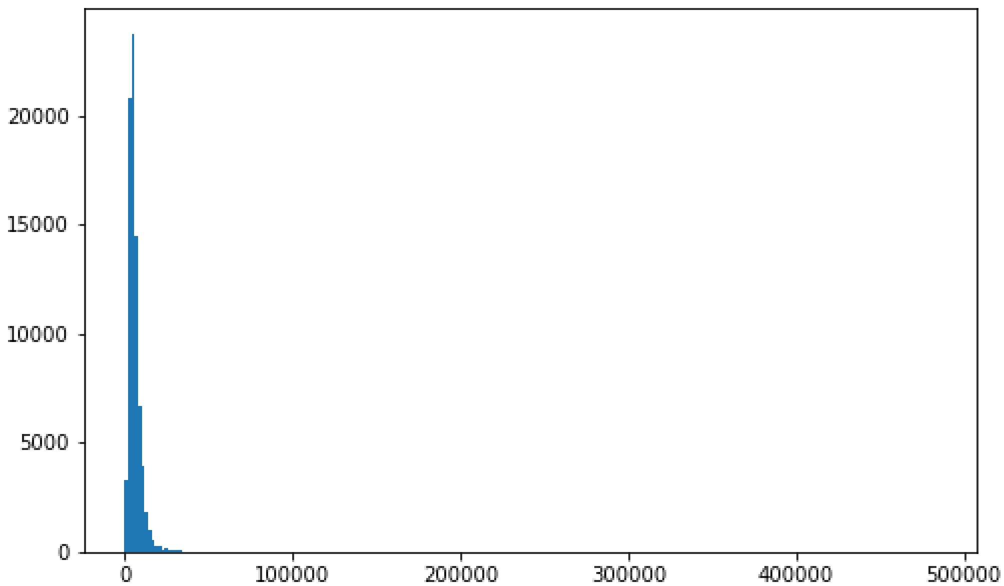
Univariate Exploration

Let's start with the the checking the distribution of the peoples monthly income

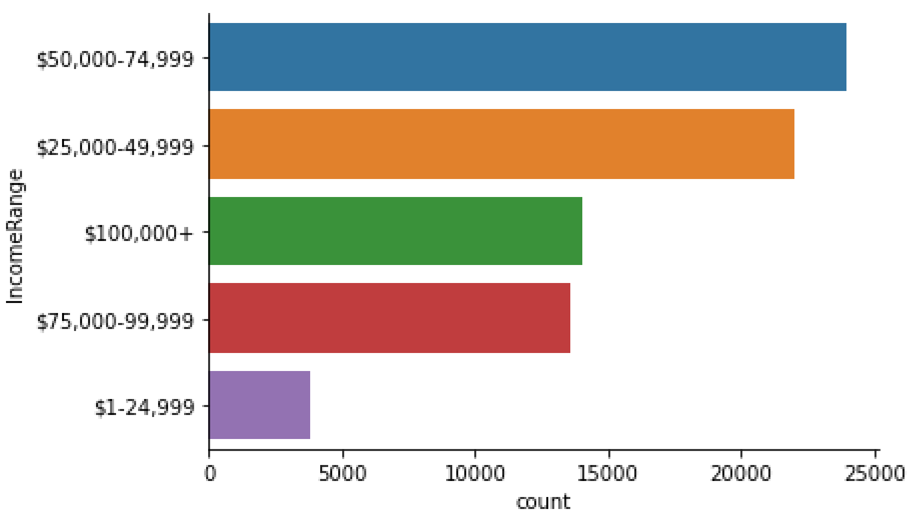
Question: What's the distribution of Income like?

In [17]:
```
# create bin size
binsize = 2000
bins = np.arange(0, data['StatedMonthlyIncome'].max()+binsize, binsize)

#plot continuous distribution of income
plt.figure(figsize=[8, 5])
plt.hist(data = data, x = 'StatedMonthlyIncome', bins = bins);
```



In [18]:
```
# Visualize in income in categories
ax = sns.countplot(y="IncomeRange", data=data, order = data.IncomeRange.value_counts().i
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False);
```



Observation

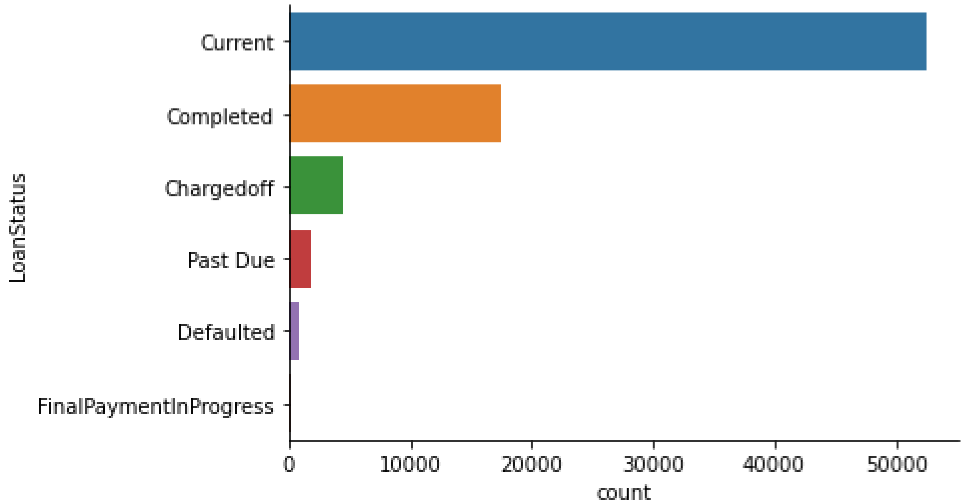
As expected, it was skewed to the left which means there are a lot more poor people. Most people fall into 25000 to 49999 or 50000 to 74999 range. A very small percentage had 1 to 24999 dollars per year.

Question: How many loans distributed on across the various loan status?

In [19]:
```
print(data.LoanStatus.value_counts())
```

```
ax = sns.countplot(y="LoanStatus", data=data, order = data.LoanStatus.value_counts().ind
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False);
```

```
Current                   52468
Completed                 17551
Chargedoff                 4434
Past Due                   1857
Defaulted                   877
FinalPaymentInProgress      189
Name: LoanStatus, dtype: int64
```



Observation

From the above we could see that a bunch of the loans are still being paid and definitely a large amount of loans are usually paid off and only about a quarter is written off a bad debt. We could look look at the distribution of people over their income ranges next.

In [20]:
```
data.head()
```

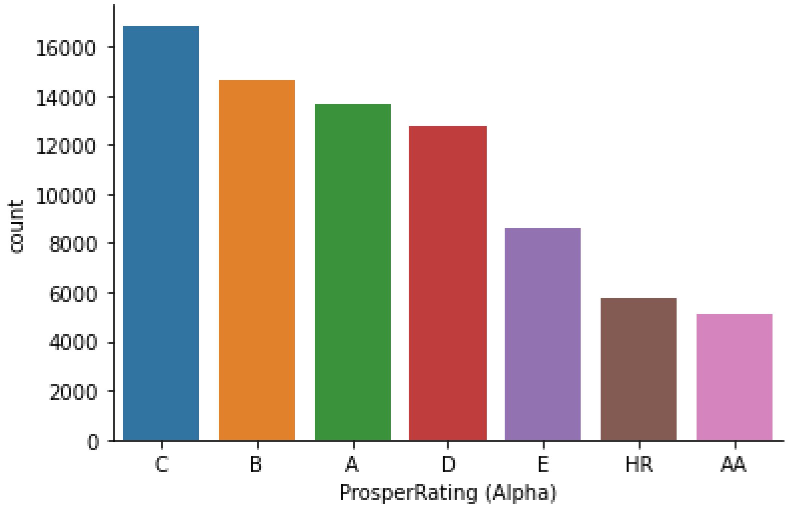
Out[20]:

	LoanStatus	ClosedDate	BorrowerAPR	EstimatedLoss	ProsperRating (Alpha)	ProsperScore	ListingCategory (numeric)	C
0	Current	NaT	0.12016	0.0249	A	7.0	2	
1	Current	NaT	0.12528	0.0249	A	9.0	16	
2	Current	NaT	0.24614	0.0925	D	4.0	2	
3	Current	NaT	0.15425	0.0449	B	10.0	1	
4	Current	NaT	0.31032	0.1275	E	2.0	1	

In [21]:
```
print(data["ProsperRating (Alpha)"].value_counts())
ax = sns.countplot(x = "ProsperRating (Alpha)", data=data, order = data["ProsperRating (
ax.spines["top"].set_visible(False)
ax.spines["right"].set_visible(False);
```

```
C      16850
B      14614
A      13663
D      12797
```

```
E       8607
HR      5727
AA      5118
Name: ProsperRating (Alpha), dtype: int64
```



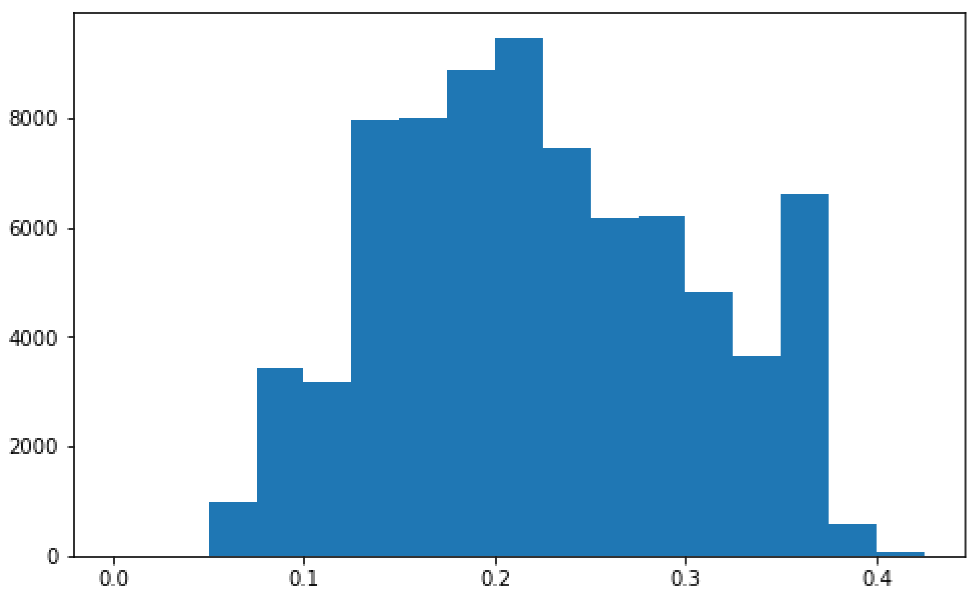
Observation

This shows that most loans are in the C category. link explains each of the categories well. Although I couldn't get a valid explanation for the HR rating. The data dictionary suggest it's the lowest ranking category.

Question: What's the distribution of the the cost of a loan?

In [22]:
```
# create bins
binsize = 0.025
bins = np.arange(0, data['BorrowerAPR'].max()+binsize, binsize)

# visualize data
plt.figure(figsize=[8, 5])
plt.hist(data = data, x = 'BorrowerAPR', bins = bins);
```



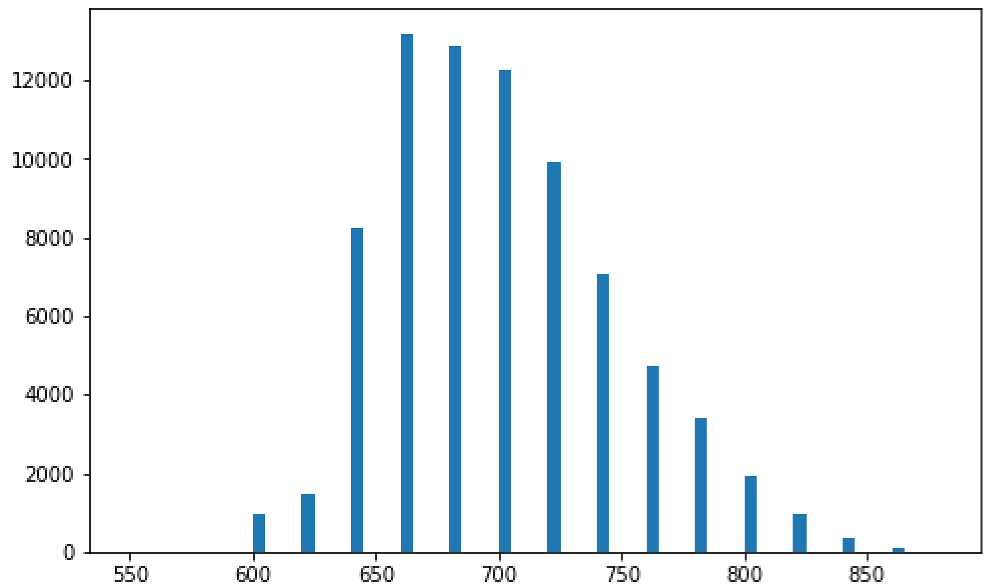
Observation

Looking at the distribution of the BorrowersAPR, we can see that it is normally distributed around 0.2.

Question: What's the distribution of the Credit Scores?

In [23]:
```
# create bins
binsize = 5
bins = np.arange(550, data['CreditScoreRangeLower'].max()+binsize, binsize)

# visualize bins
plt.figure(figsize=[8, 5])
plt.hist(data = data, x = 'CreditScoreRangeLower', bins = bins);
```



Observation

Looking at the distribution of the Credit Scores, looks categorical but that because we are using the lower limit of the credit score range to represent the credit score.

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

Most of the distributions were expected or at least not out of ordinary and there were no need for transformations. However, I noticed that there were no values of a credit score lower than 600(very close to 580 which is considered the threshold for low credit score), this means Prosper must have been playing it safe all along by not giving people with a low credit score loans.

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?

None

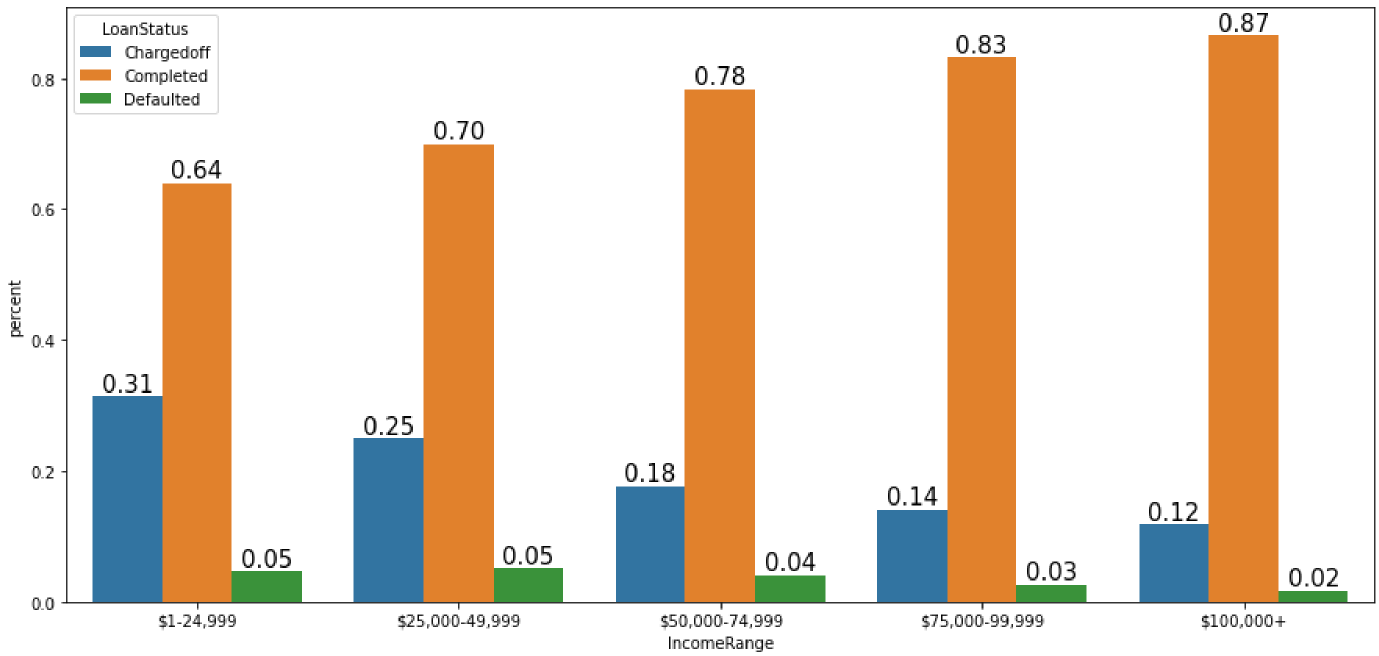
Bivariate explorations

Question what's the percent of defaulted, charged off and settled loans across income levels?

In [24]:
```
# create a dataframe of count of possible combination IncomeRange and LoanStatus categor
temp_df = data.query("~ClosedDate.isna()").groupby(["IncomeRange","LoanStatus"],as_index
```

```
temp_df.BorrowerAPR = temp_df.BorrowerAPR/temp_df.groupby("IncomeRange",as_index=False)[
temp_df.rename(columns={"BorrowerAPR":"percent"},inplace=True)

#Plot data
plt.figure(figsize=(15,7))
ax = sns.barplot(x="IncomeRange",y="percent",data=temp_df,hue="LoanStatus",order=['$1-24
for bar in ax.patches:
    ax.annotate(
        f"{bar.get_height():.2f}",
        (bar.get_x() + bar.get_width() / 2,bar.get_height()),
        ha='center',
        va='center',
        size=15,
        xytext=(0, 8),
        textcoords='offset points'
    )
```



Observation

From the above graph, we can clearly see that the percent of chargedoff and defaulted loans decreases as we go up in the income brackets

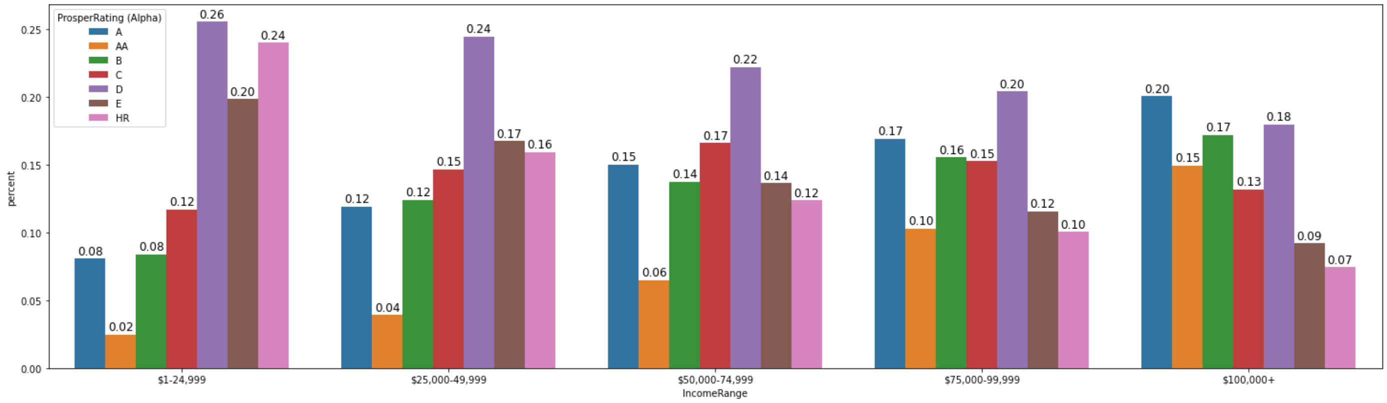
Question: Does the company recognize the risk in giving poor people loans?

In [25]:

```
# create a dataframe of count of possible combination IncomeRange and Rating categories
temp_df = data.query("~ClosedDate.isna()").groupby(["IncomeRange","ProsperRating (Alpha)
temp_df.BorrowerAPR = temp_df.BorrowerAPR/temp_df.groupby("IncomeRange",as_index=False)[
temp_df.rename(columns={"BorrowerAPR":"percent"},inplace=True)

#plot data
plt.figure(figsize=(25,7))
ax = sns.barplot(x="IncomeRange",y="percent",data=temp_df,hue="ProsperRating (Alpha)",or
for bar in ax.patches:
    ax.annotate(
        f"{bar.get_height():.2f}",
        (bar.get_x() + bar.get_width() / 2,bar.get_height()),
        ha='center',
        va='center',
        size=12,
        xytext=(0, 8),
```

```
        textcoords='offset points'
    )
```

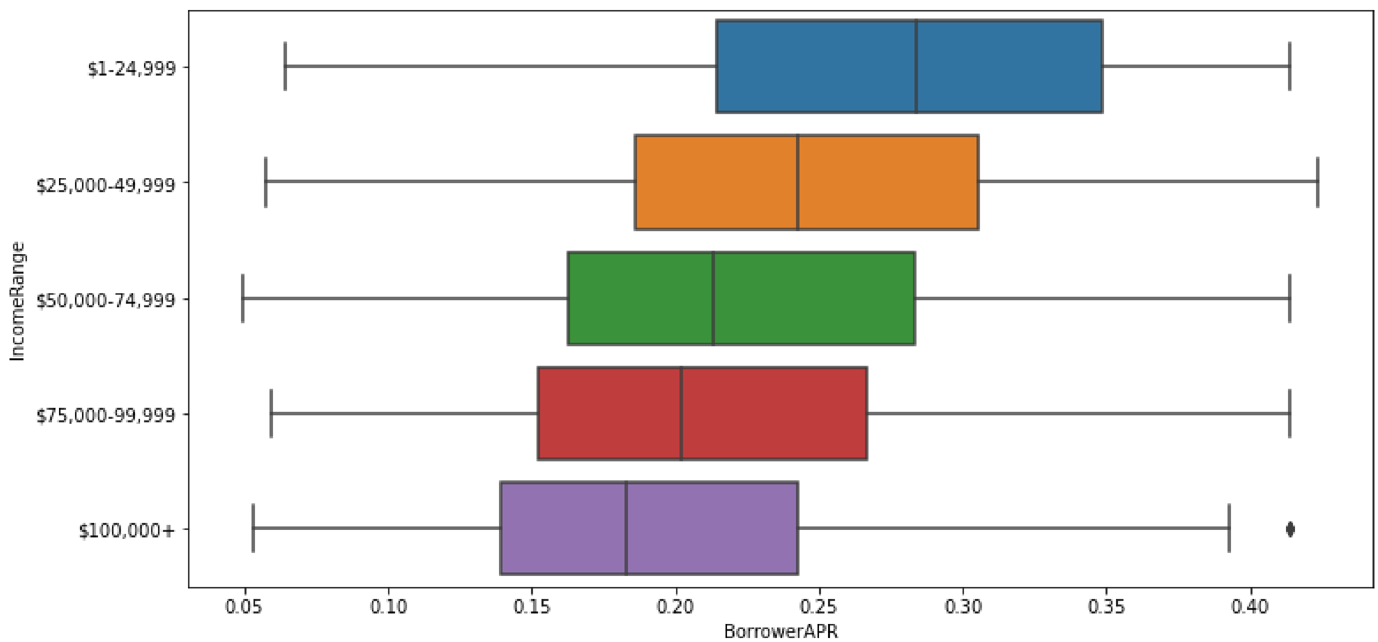


Observations

Yes they do, the rich people's loans are usually rated higher while poorer people are usually rated lower.

Question: Does this reflect in the cost of loans for poorer people?

In [26]:
```
plt.figure(figsize=(12,6))
sns.boxplot(x="BorrowerAPR",y="IncomeRange",data=data,order=['$1-24,999', '$25,000-49,99
```

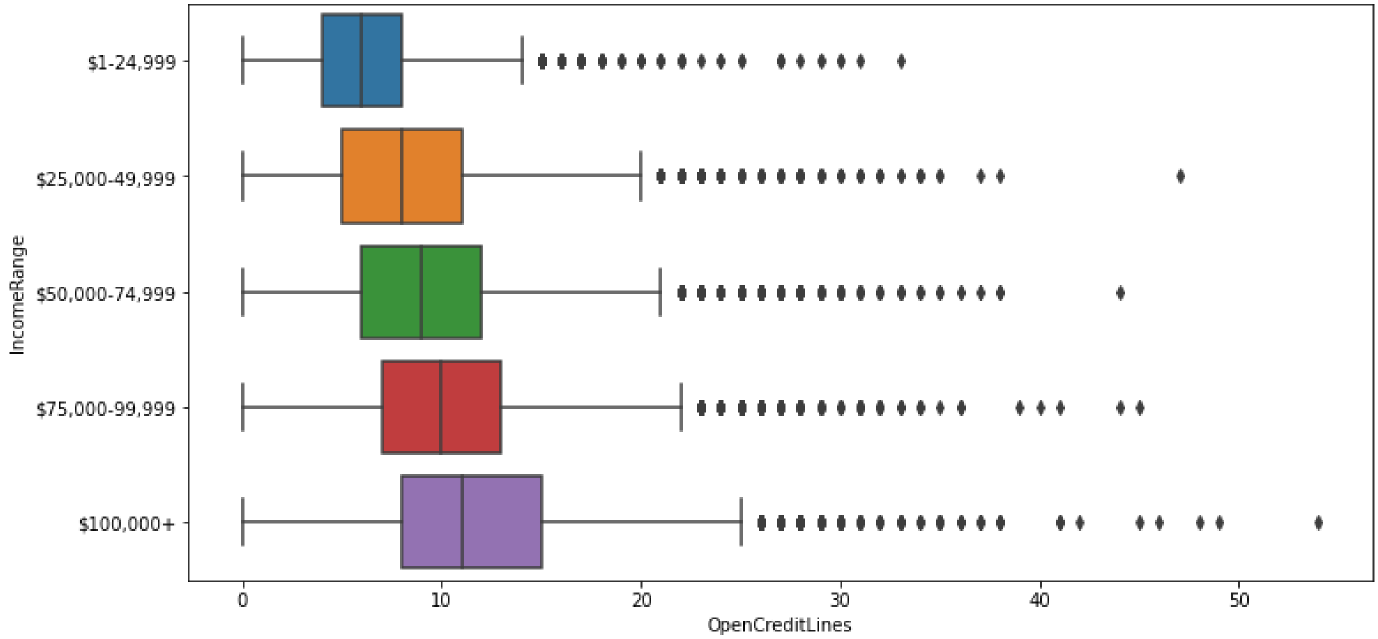


Observation

Yes it does, as a consequence of the of the risk tied to borrowing a poor person money as seen in the visualization before this. The lender increases the cost of borrowing for poorer people to compensate for the risk they are taking.

Question: Are poor people more likely not to get loans because of this cost?

In [27]:
```
plt.figure(figsize=(12,6))
sns.boxplot(x="OpenCreditLines",y="IncomeRange",data=data,order=['$1-24,999', '$25,000-4
```

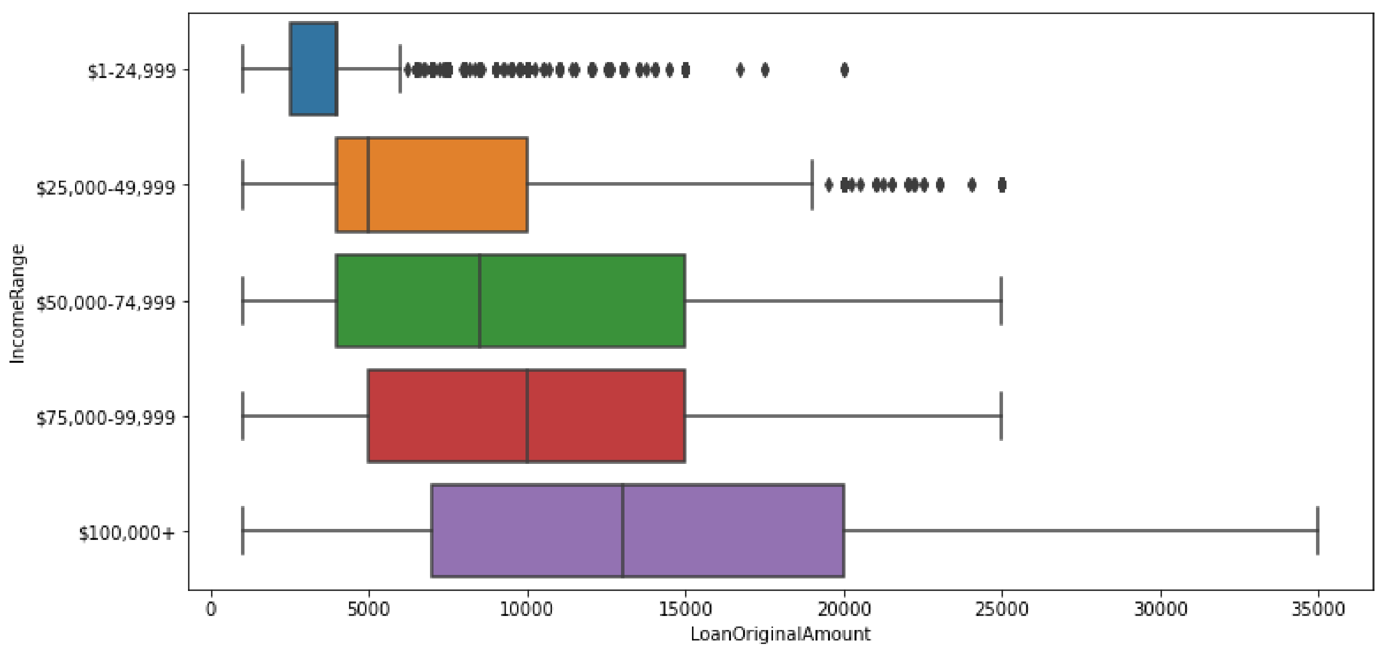


Observation

Sadly yes, the rich seems generally have access to more and cheaper loans than the poor because of they are less risky to loan too. It also follows a point Robert Kiyosaki(Rich Dad, Poor Dad) made is that instead of getting out of debt, people should use debt to their advantage.

Question: Does the loan amount vary among income levels?

In [28]:
```
plt.figure(figsize=(12,6))
sns.boxplot(x="LoanOriginalAmount",y="IncomeRange",data=data,order=['$1-24,999', '$25,00
```



Observation

Yes it does, the rich also has access to significanty larger loans.

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?

Rich people tend to settle their loans, the companies recognize that their loans are less risky therefore offers cheaper loans to the rich which then encourages the rich to take more loans.

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

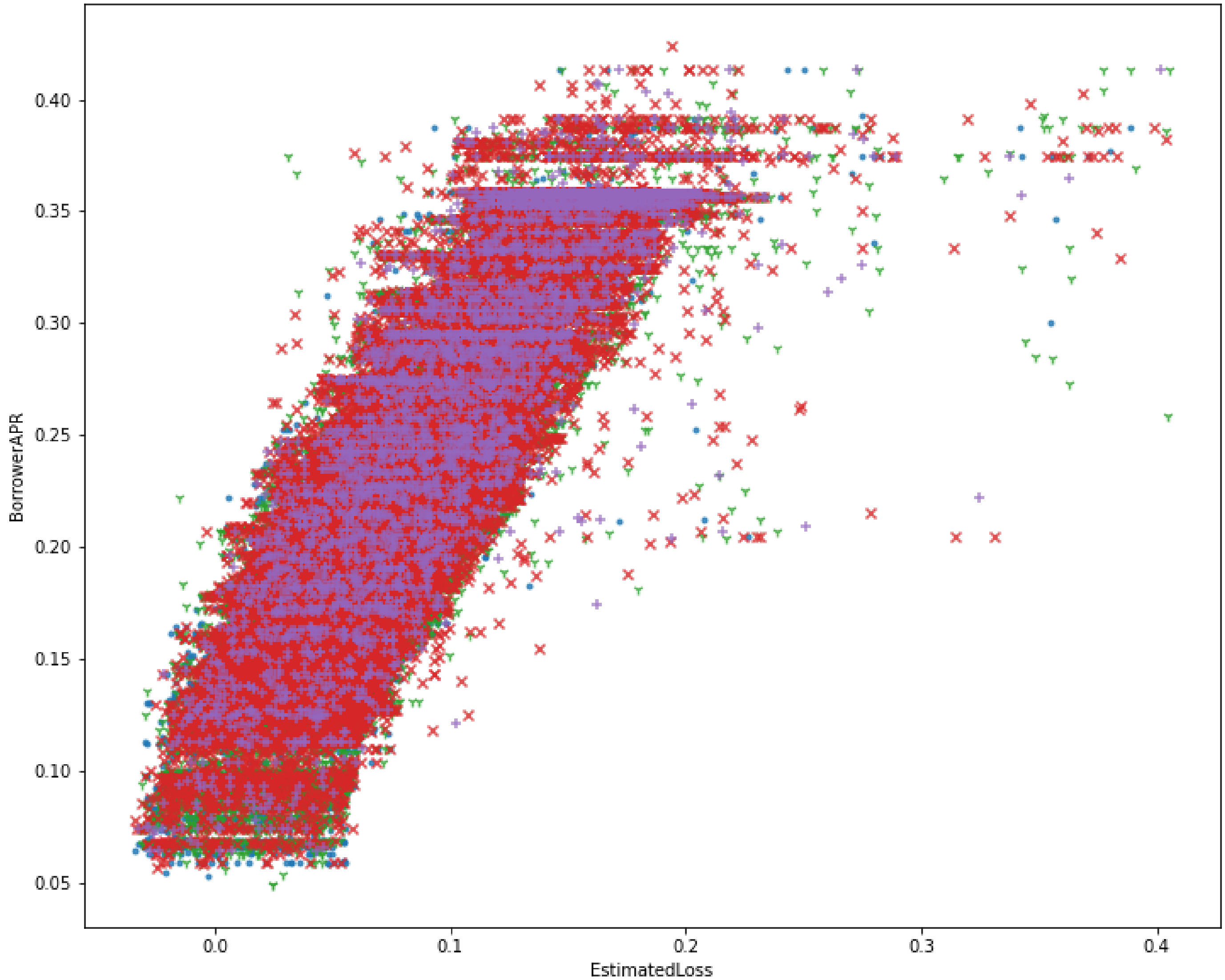
None

Multivariate Exploration

Question: How is the IncomeRange affected by the BorrowerAPR and EstimatedLoss?

In [29]:
```
# map markers to categories
markers = [
    ["$100,000+","."],
    ["'$75,000-99,999'","^"],
    ["$50,000-74,999","1"],
    ["$25,000-49,999","x"],
    ["$1-24,999","+"]
]

#plot data
plt.figure(figsize=(12,10))
for income_range,marker in markers:
    temp_df = data[data["IncomeRange"]==income_range]
    sns.regplot(data = temp_df, x = "EstimatedLoss", y = "BorrowerAPR", x_jitter= 0.04,
```



Observation:

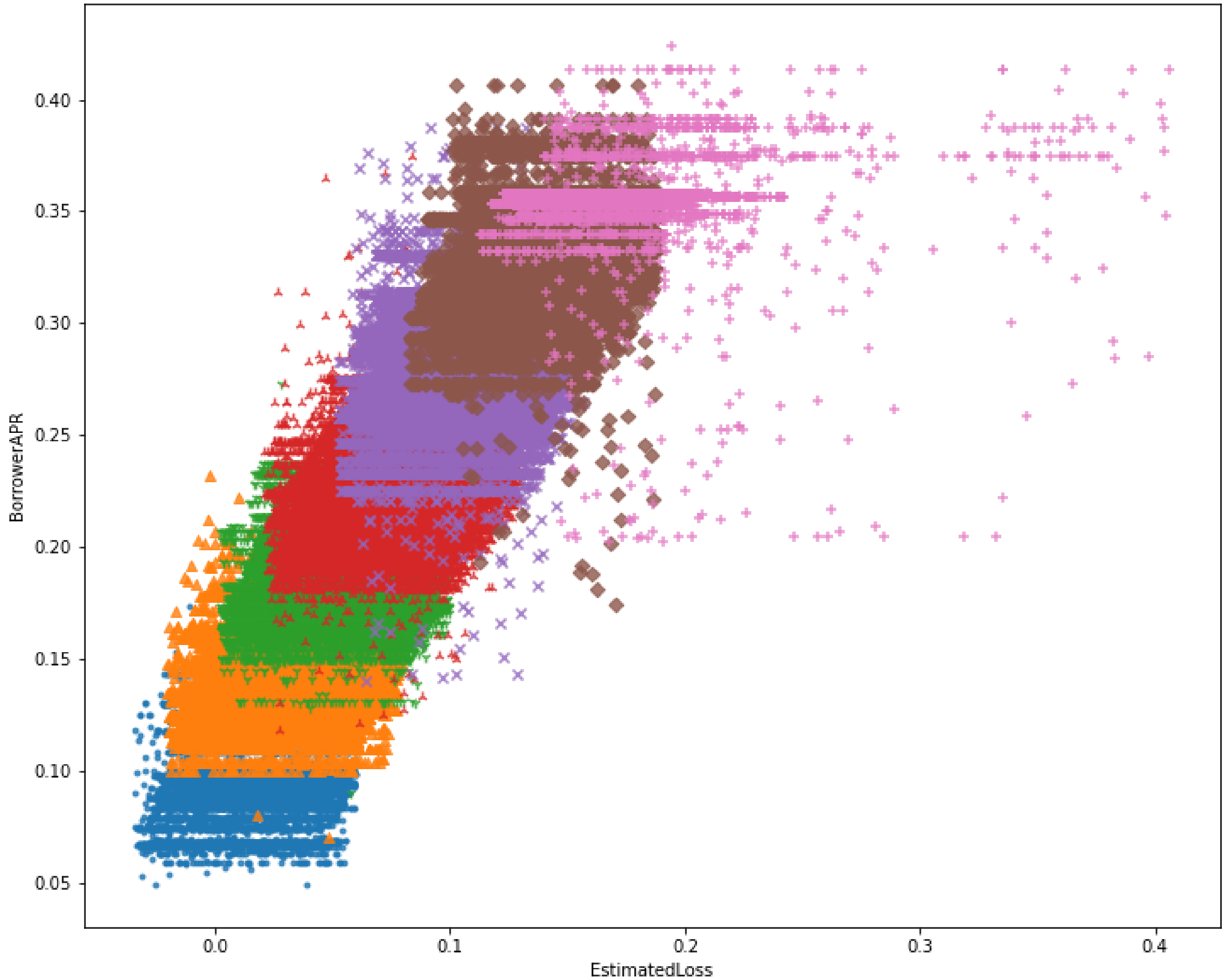
Apart from a positive correlation between the estimated loss and the cost of loan. There is no common relationship between both values and the income range

Question: How is the rating affected by the BorrowerAPR and EstimatedLoss?

In [30]:

```
# map markers to categories
markers = [
    ["AA","."],
    ["A","^"],
    ["B","1"],
    ["C","2"],
    ["D","x"],
    ["E","D"],
    ["HR","+"]
]

#plot data
plt.figure(figsize=(12,10))
for rating,marker in markers:
    temp_df = data[data["ProsperRating (Alpha)"]==rating]
    sns.regplot(data = temp_df, x = "EstimatedLoss", y = "BorrowerAPR", x_jitter= 0.04,
```



Observation

We noticed that estimated the HR rated loans have the bigges of the estimated loss and also highest cost of loans. Both gets better from E to D to C to B to A to AA.

Conclusions

In summary, loans are very much beneficial to the rich unlike their poorer counterparts.