Data Exploration

The file itself, consists of 45000 datasets with 18 columns. Within those columns 17 of them are features and the final column is the indicator whether the loan is at its default state.

First step is trying to figure out what type of data are.

```
In [2]: loan_data = pd.read_csv("C:\\Users\\xiexi\\OneDrive\\Desktop\\Part 2. loan_data_final.csv")
        loan_data = loan_data.iloc[:,1:]
        loan_data.dtypes

Out[2]: person_age                       int64
        person_gender                   object
        person_education                object
        person_income                    int64
        person_emp_exp                   int64
        person_home_ownership           object
        loan_intent                     object
        loan_int_rate                  float64
        loan_percent_income            float64
        cb_person_cred_hist_length       int64
        credit_score                     int64
        previous_loan_defaults_on_file  object
        loan_to_income_ratio           float64
        loan_type                       object
        dependents_count                 int64
        regional_unemployment_rate     float64
        borrower_risk_score            float64
        loan_status                      int64
        dtype: object
```

Next step is making sure there are no empty data here. Based on this line of code, we can tell there are none.

```
: loan_data.isnull().sum()

: person_age                       0
  person_gender                    0
  person_education                 0
  person_income                    0
  person_emp_exp                   0
  person_home_ownership            0
  loan_intent                      0
  loan_int_rate                    0
  loan_percent_income              0
  cb_person_cred_hist_length       0
  credit_score                     0
  previous_loan_defaults_on_file   0
  loan_to_income_ratio             0
  loan_type                        0
  dependents_count                 0
  regional_unemployment_rate       0
  borrower_risk_score              0
  loan_status                      0
  dtype: int64
```

Based on this result we found out that the data consists a mixture of numerical and categorical. And the results are binary. Solely based on this, the most straightforward

decision is using tree based models. Firstly, I have to split the data into numerical and categorical and analysis them separately.

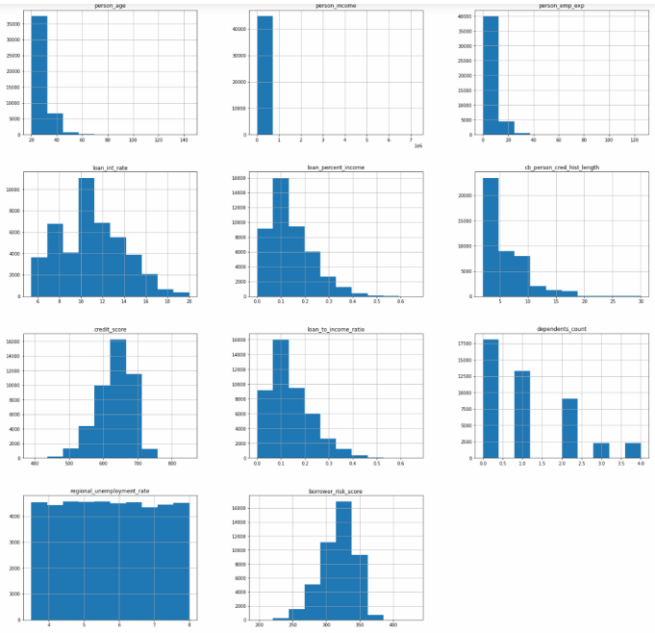
```
numerical_features = X.select_dtypes(include=['int64','float64']).columns.tolist()
categorical_features = X.select_dtypes(include=['object']).columns.tolist()
```

First, we start with the analysis of numerical data.

Analysing Numerical Columns

	person_age	person_income	person_emp_exp	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	loan_to_income_ratio	de
count	45000.000000	4.500000e+04	45000.000000	45000.000000	45000.000000	45000.000000	45000.000000	45000.000000	
mean	27.764178	8.031905e+04	5.410333	11.006606	0.139725	5.867489	632.608756	0.139726	
std	6.045108	8.042250e+04	6.063532	2.978808	0.087212	3.879702	50.435865	0.087212	
min	20.000000	8.000000e+03	0.000000	5.420000	0.000000	2.000000	390.000000	0.000000	
25%	24.000000	4.720400e+04	1.000000	8.590000	0.070000	3.000000	601.000000	0.070000	
50%	26.000000	6.704800e+04	4.000000	11.010000	0.120000	4.000000	640.000000	0.120000	
75%	30.000000	9.578925e+04	8.000000	12.990000	0.190000	8.000000	670.000000	0.190000	
max	144.000000	7.200766e+06	125.000000	20.000000	0.660000	30.000000	850.000000	0.660000	

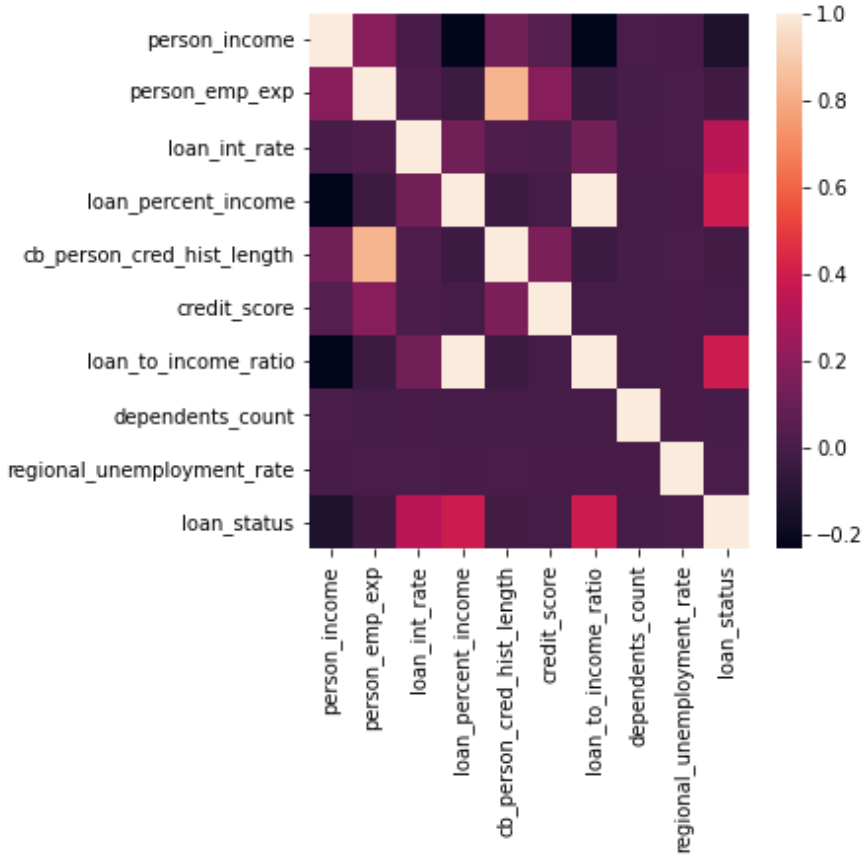
Now, we check the data distribution.



The data itself doesn't seem to be very well distributed. A lot of them are heavily skewed. When treating the data we'd prefer using median than mean due to this nature. Further

more, when choosing the models, **we have to use a model that's independent of the data's statistical distribution.**

Next we look at the correlation of the data. The initial analysis:

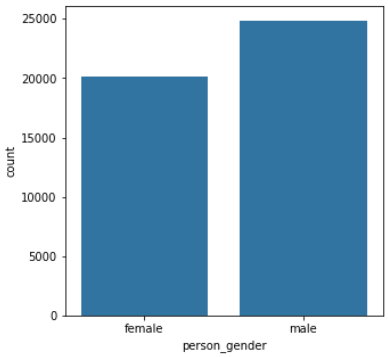


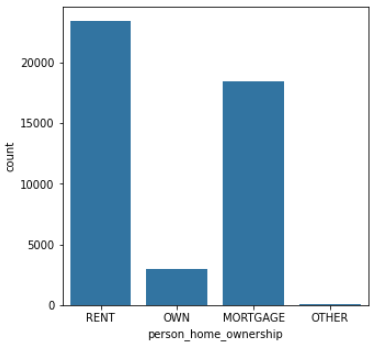
Most of the features are not very correlated to one another, except for person's age and person's employment experience and credit history length. Borrower's risk score and credit score are also very correlated. Most of the models do not favor highly correlated features so we might need to get rid of some.

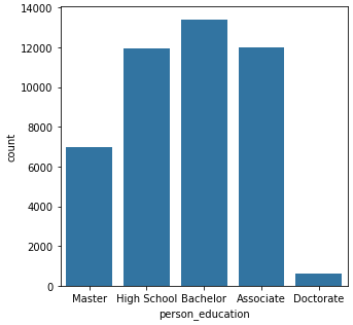
For our Y, there are some correlation between loan interest rates, loan percent income and loan to income ratio, but not super strong. Ideally linear regression might also work but those data are not very normally distributed which failed the criteria.

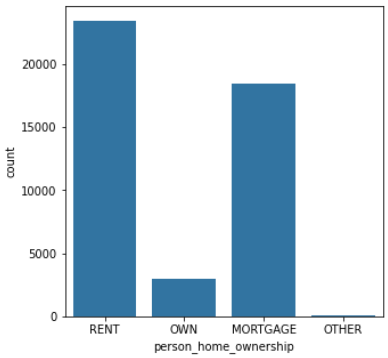
Next we do an analysis on categorical features.

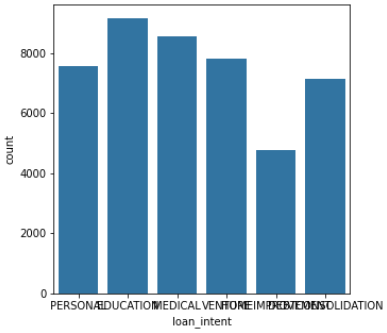
Here we do a histogram analysis on each features. No particular features is very unevenly distributed in terms of category.

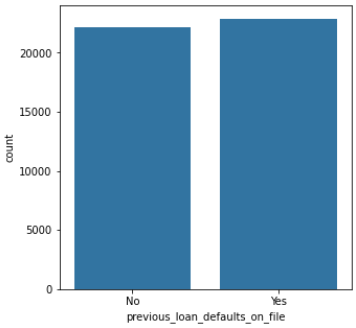


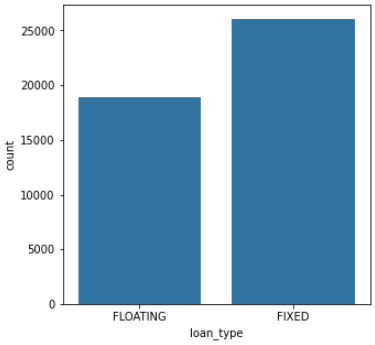












Feature Engineering

Before model training, first step is preparing the data. For numerical parts, during the previous analysis we realized that age and risk scores are very correlated to some other features. To make the model more robust, we will drop those 2.

For numerical data, we also need to scale them. As we observed earlier, a lot of features are quite heavily skewed, so over here we are using median in stead of mean for the scaler.

Next step for categorical data we need to do some encoding: Most machine-learning models handle only integer values or other understandable formats. We need to change categorical data into integer format so that data with transformed categorical values can be fed into models to increase accuracy.

```
In [14]: numerical_features = numerical_features[1:10]
```

```
In [15]: #data is heavily skewed, will use median instead of mean
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),  # Fill missing values
    ('onehot', OneHotEncoder(handle_unknown='ignore'))  # One-hot encode
])
preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_transformer, numerical_features),
    ('cat', categorical_transformer, categorical_features)
])
```

Then before we putting data into the model, for testing purposes and parameters tuning, need to split the data into testing and training sets: here I'm using 20%, 80% split:

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify = Y)
```

Model Training

Based on the analysis of the data itself, the decision is to use random forest as models. Default status and some loan based features do have some correlation but the requirements for it to be normally distributed and categorical inputs made linear regression out of the picture. The data is skewed, but not very correlated to one another. But the amount and nature of the features made decision tree easy to overfit.

So a more robust random forest will be suitable. On the other hand, we do want to know how features are contributing to default rate to make the analysis more transparent, random forest's relative feature significance will enable us to look through.

During the model training, there are different parameters for random forest models. For criterion I'm using Gini as the Y input is categorical as well. For n_estimators I'm using 500 with the luxury of our large data size. For min_samples_splits and min_samples_leaf I'm using 10 and 5 given the number of features. I trialed and errored a few other numbers but the results weren't too different. Ideally, an optimizer like GridSearch can be used here.

For the results, we achieved some quite accurate predictions:

	precision	recall	f1-score	support
0	0.93	0.98	0.95	7000
1	0.91	0.74	0.82	2000
accuracy			0.93	9000
macro avg	0.92	0.86	0.89	9000
weighted avg	0.93	0.93	0.92	9000

Then we look at which feature played an important part in the default.

	Feature	Importance
11	previous_loan_defaults_on_file	0.106356
7	loan_int_rate	0.052233
5	person_home_ownership	0.034678
3	person_income	0.028678
6	loan_intent	0.015722
12	loan_to_income_ratio	0.013322
8	loan_percent_income	0.012756
10	credit_score	0.006156
1	person_gender	0.001300
15	regional_unemployment_rate	0.001089
9	cb_person_cred_hist_length	0.000878
14	dependents_count	0.000689
2	person_education	0.000644
4	person_emp_exp	0.000256
13	loan_type	0.000078
0	person_age	0.000000
16	borrower_risk_score	0.000000

Without surprise, previous loan defaults on file made it a high target. But might also because the data itself is binary. Loan interest rate also made it happen due to the potential cashflow difficulties as well as the person's risk level. Person home ownership made into top 3 as well which reflects the person's financial stability.