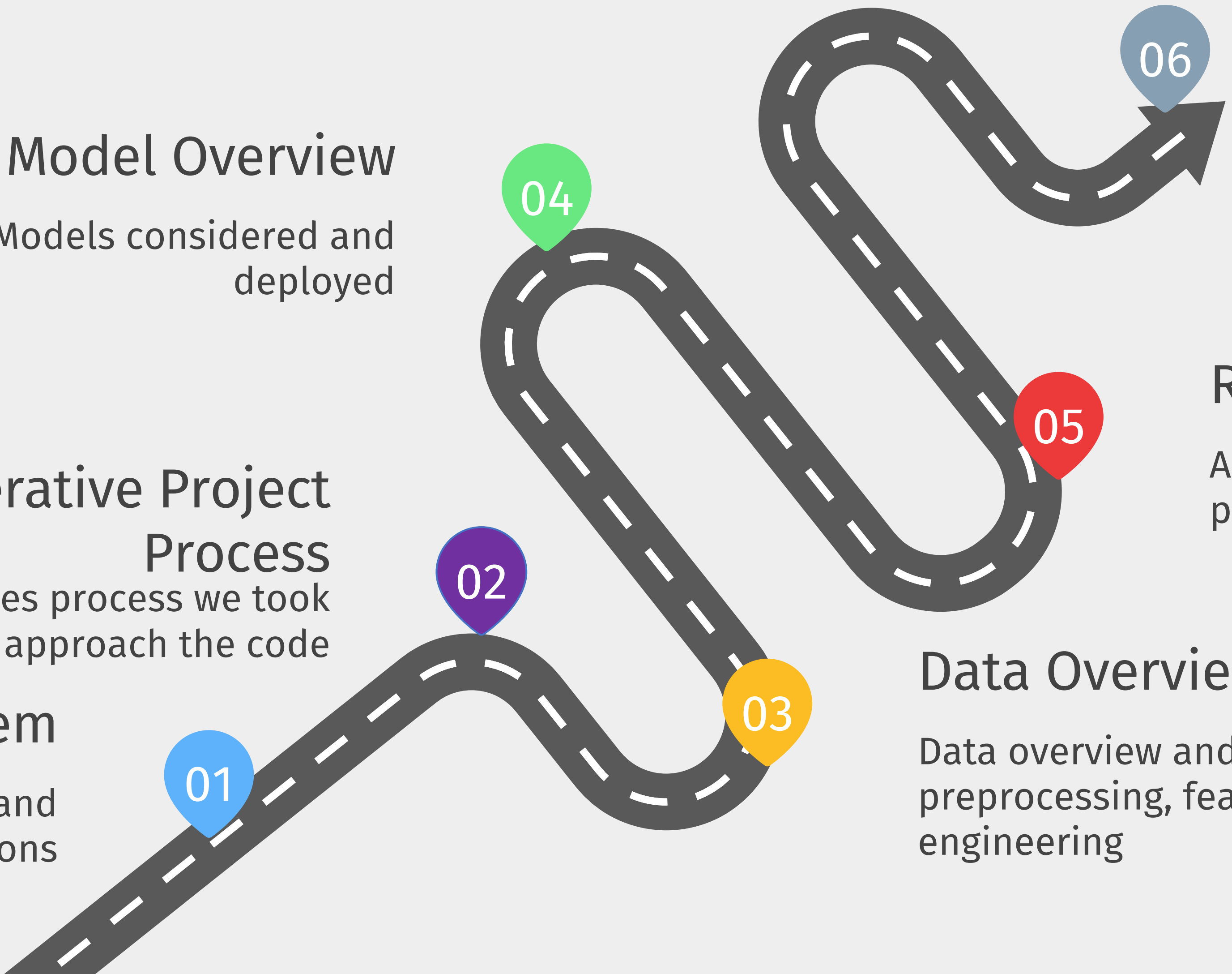
Predicting Loan Upper Limit

Anh Nguyen, Eli Friedmann, Hunter Lampley,
Joey Ripcho, Nay Tjatur

Agenda



Business Problem

Sets up background and business implications

Iterative Project Process

Describes process we took to approach the code

Data Overview & Processing

Data overview and dictionary, preprocessing, feature selection and engineering

Model Overview

Models considered and deployed

Results

Analysis of model performance

Next Steps

Recommendations and reflections

BUSINESS UNDERSTANDING

SERVICE

lend money to businesspeople, small businesses, working professionals, students, and others.

OBJECTIVE

make a profit by lending money and making sure there are a minimum # of defaults among cust.

OUTCOME

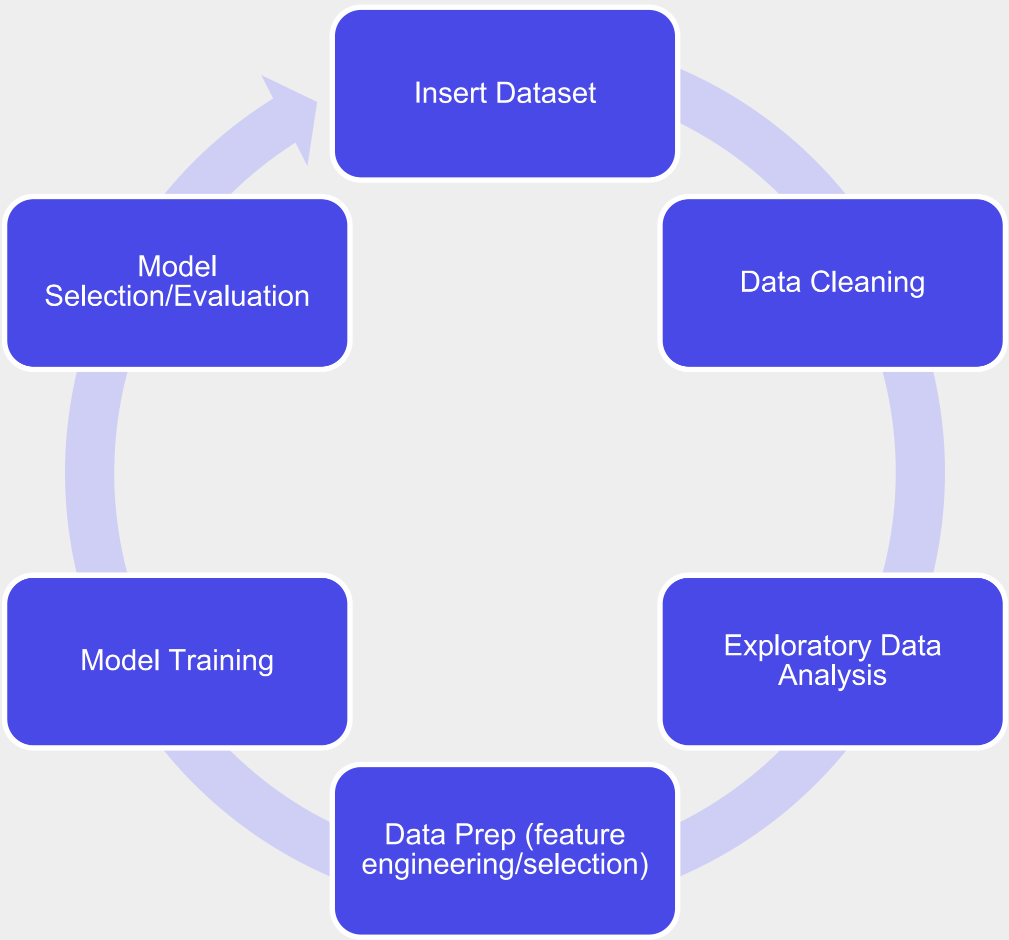
Predict the upper limit of the loan amount to provide to cust.

→

MODEL

Multiple Linear Regression

Iterative Project Process


Insert Dataset
Data Cleaning
Exploratory Data Analysis
Data Prep (feature engineering/selection)
Model Training
Model Selection/Evaluation

In [3]:

```
df = pd.read_excel('Credit_Card_Information.xlsx', index_col = 0, parse_dates = True)
df.head()
```

Out[3]:

	No_of_Credit_Cards	Unnamed: 2	Credit_Card_Payment_Per_Month (USD)	Name of the customer	Unnamed: 5	Unnamed: 6	Customer_Address
0	4	NaN	1149.946696	##REDACTED##	NaN	NaN	##REDACTED##
1	5	NaN	1350.901893	##REDACTED##	NaN	NaN	##REDACTED##
2	4	NaN	1216.62937	##REDACTED##	NaN	NaN	##REDACTED##
3	2	NaN	617.964971	##REDACTED##	NaN	NaN	##REDACTED##
4	4	NaN	1202.738744	##REDACTED##	NaN	NaN	##REDACTED##

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12000 entries, 0 to 11999
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   No_of_Credit_Cards                      11994 non-null  object
 1   Unnamed: 2                              0 non-null      float64
 2   Credit_Card_Payment_Per_Month (USD)     12000 non-null  object
 3   Name of the customer                    12000 non-null  object
 4   Unnamed: 5                              0 non-null      float64
 5   Unnamed: 6                              0 non-null      float64
 6   Customer_Address                        12000 non-null  object
 7   Upper_Loan_Limit (in thousands of USD)  12000 non-null  object
dtypes: float64(3), object(5)
memory usage: 843.8+ KB
```

Import Credit_Card_Information dataset first

In [3]:
```
df = pd.read_excel('Credit_Card_Information.xlsx', index_col = 0, parse_dates = True)
df.head()
```

Out[3]:

	No_of_Credit_Cards	Unnamed: 2	Credit_Card_Payment_Per_Month (USD)	Name of the customer	Unnamed: 5	Unnamed: 6	Customer_Address
0	4	NaN	1149.946696	##REDACTED##	NaN	NaN	##REDACTED##
1	5	NaN	1350.901893	##REDACTED##	NaN	NaN	##REDACTED##
2	4	NaN	1216.62937	##REDACTED##	NaN	NaN	##REDACTED##
3	2	NaN	617.964971	##REDACTED##	NaN	NaN	##REDACTED##
4	4	NaN	1202.738744	##REDACTED##	NaN	NaN	##REDACTED##

In [6]:
```
#drop blank and redacted columns
df = df.drop(['Unnamed: 2', 'Unnamed: 5', 'Unnamed: 6', 'Name of the customer', 'Customer_Address'
```

In [7]:
```
df
```

Out[7]:

	No_of_Credit_Cards	Credit_Card_Payment_Per_Month (USD)	Upper_Loan_Limit (in thousands of USD)
0	4	1149.946696	92.548079
1	5	1350.901893	85.805414
2	4	1216.62937	78.187597
3	2	617.964971	47.129856
4	4	1202.738744	108.301048497146 USD
...	...	...	...
11995	2	567.747412	64.775335
11996	2	547.878406	52.244325
11997	4	1178.751467	66.462858
11998	2	476.219855	43.638415
11999	4	1197.533054	USD 83.1859859369225

12000 rows × 3 columns

In [8]:
```
#using .unique() to find all distinct values from 'No_of_Credit_Cards'
df['No_of_Credit_Cards'].unique()
```

Out[8]:
```
array([4, 5, 2, 3, 6, nan, 'MISSING', 'unknown', 'Unknown!', 'Missing',
       '###?###', 'Missing??', 'UNKnoWN', 'MISSing!', 'Missing???',
       'Value MIsing', 'Value Missing!!', 'Missing Value!!!',
       'Unknown Value!!!', '!!Value!!!', 'Missing!!!', 'NOT KNOwn',
       '!!!UNKNOWN!!!', 'NOVALUE!!!', ' ', 'VALUE UNKNOWN', '___MISSING#',
       'CARDS???', 'NO OF CARDS UNKNOWN', 'NOT PROVIDED!!!!',
       'CARDS MISSING!!!', 'MISSING CARDS', 'MISSING VALUE',
       'VALUE MIssING', '!!!!!UNKNOWN VALUE!!!'], dtype=object)
```

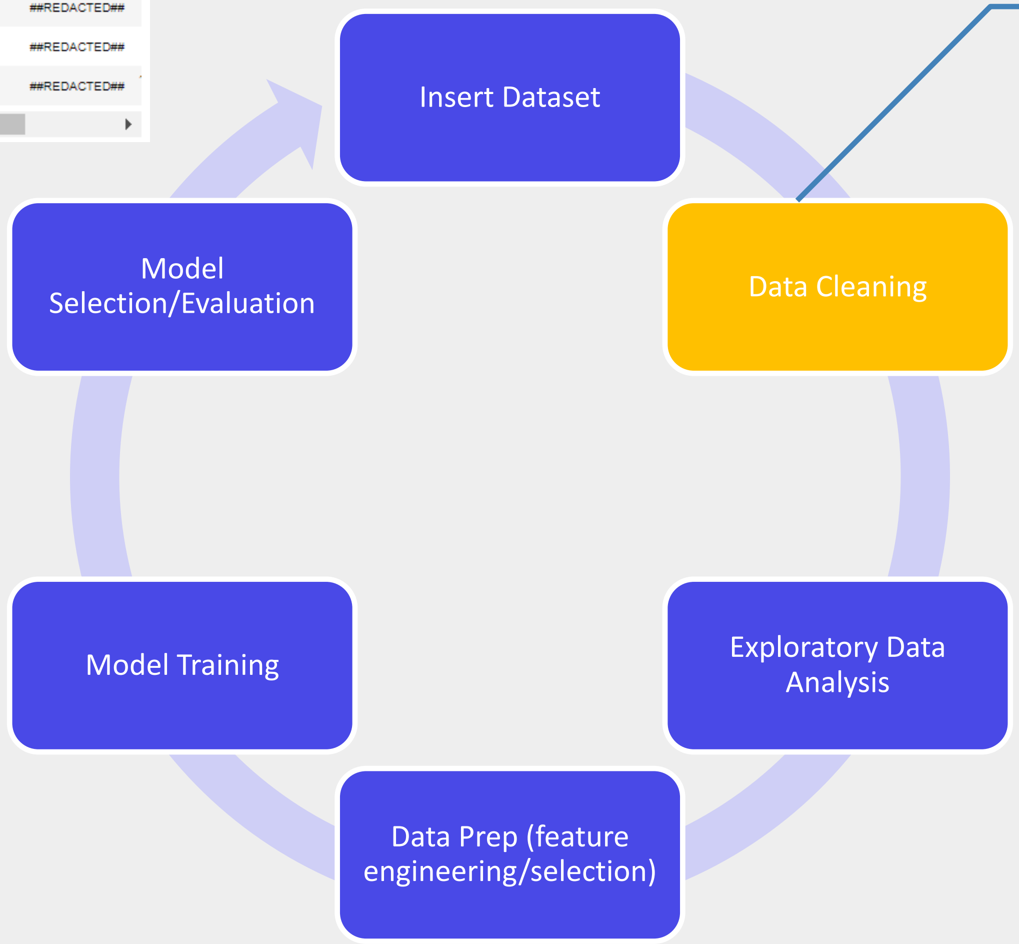
In [9]:
```
#using .unique() to find all distinct values from 'No_of_Credit_Cards'
missing = ['MISSING', 'unknown', 'Unknown!', 'Missing',
       '###?###', 'Missing??', 'UNKnoWN', 'MISSing!', 'Missing???',
       'Value MIsing', 'Value Missing!!', 'Missing Value!!!',
       'Unknown Value!!!', '!!Value!!!', 'Missing!!!', 'NOT KNOwn',
       '!!!UNKNOWN!!!', 'NOVALUE!!!', ' ', 'VALUE UNKNOWN', '___MISSING#',
       'CARDS???', 'NO OF CARDS UNKNOWN', 'NOT PROVIDED!!!!',
       'CARDS MISSING!!!', 'MISSING CARDS', 'MISSING VALUE',
       'VALUE MIssING', '!!!!!UNKNOWN VALUE!!!']
```

In [10]:
```
#replace the list missing with null
df = df.replace(missing, np.NaN)
```

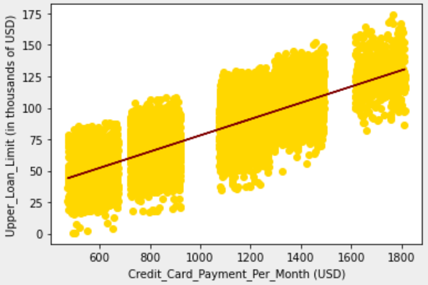
In [11]:
```
df['No_of_Credit_Cards'].mode()
```

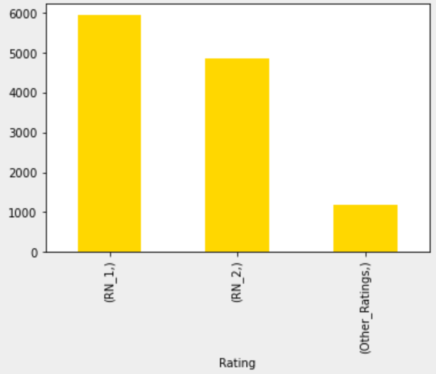
Out[11]:
```
0    4.0
dtype: float64
```

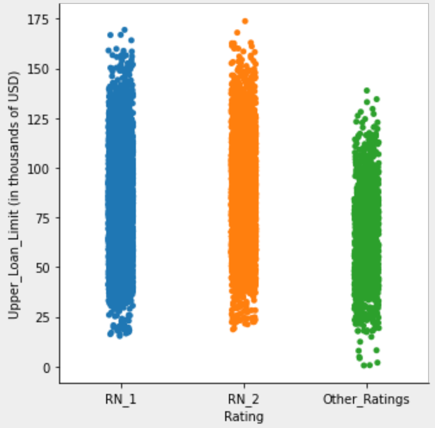
In [12]:
```
#fill in all missing data with mode of 'No_of_Credit_Cards'
df['No_of_Credit_Cards'].fillna(df['No_of_Credit_Cards'].mode()[0], inplace=True)
```

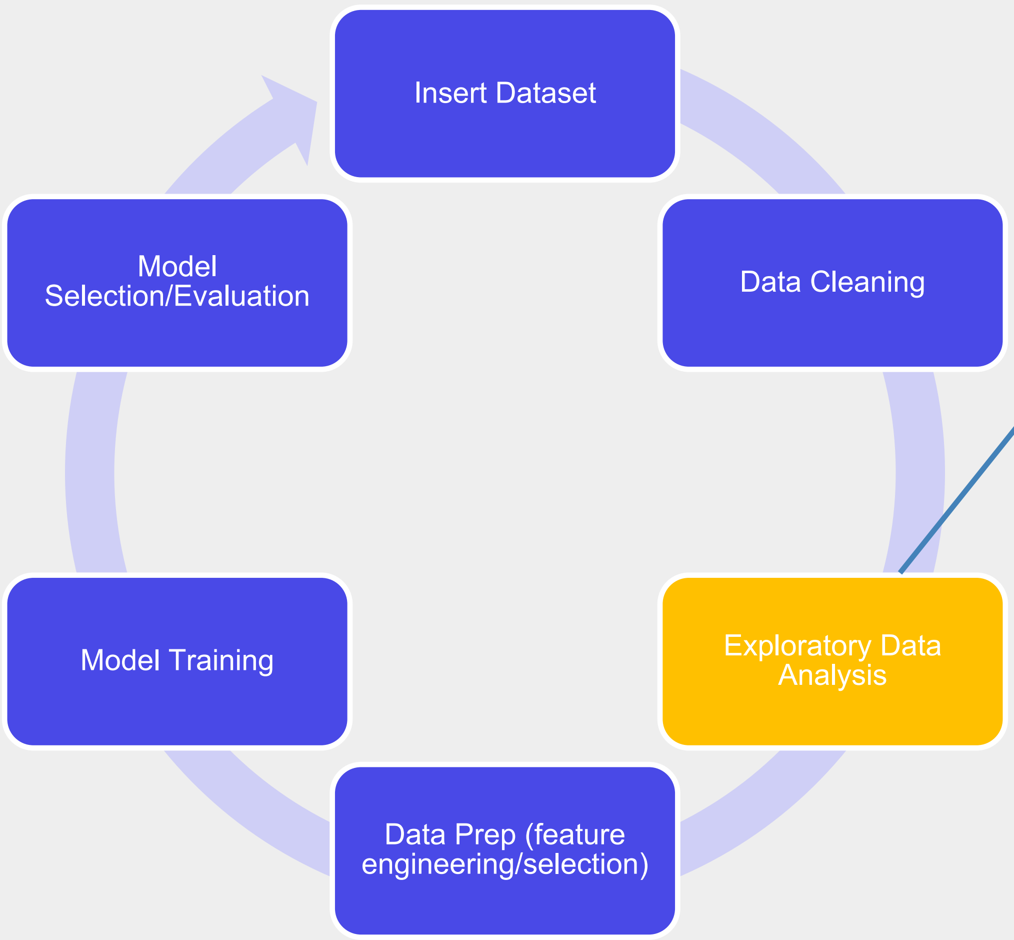


- Remove empty/redacted columns
- Impute missing/unique values with mean or mode









- Bar charts and scatter plots to make sense of data
- Double check for intuitive accuracy

In [29]:
```python
pro='Profession'
other=np.full((1), 'Other_Occupations')
ohe=OneHotEncoder(categories='auto', drop=other, sparse=False)
prof=pi[pro].array.reshape(-1,1)
prof1=pd.DataFrame(ohe.fit_transform(prof))
prof1
```

Out[29]:

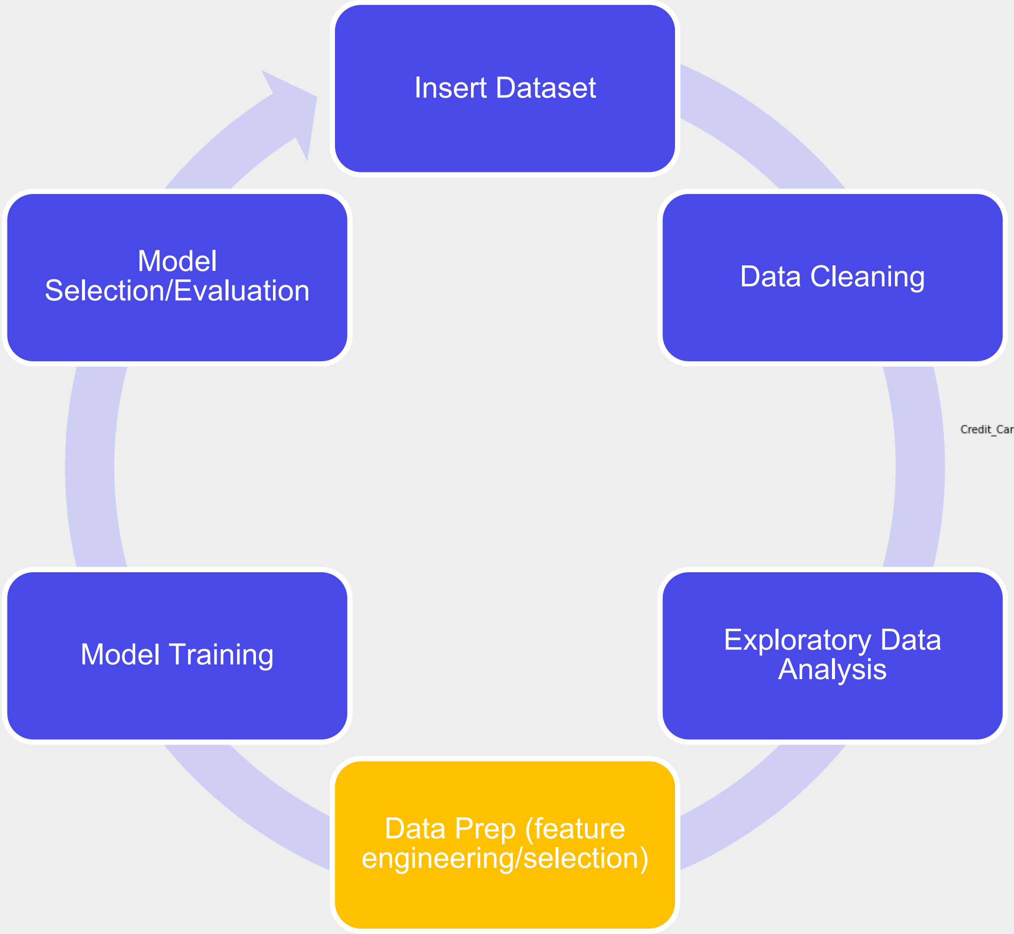
	0	1
0	0.0	1.0
1	0.0	1.0
2	0.0	1.0
3	1.0	0.0
4	0.0	1.0
...	...	...
11995	0.0	1.0
11996	0.0	1.0
11997	0.0	0.0
11998	1.0	0.0
11999	0.0	1.0

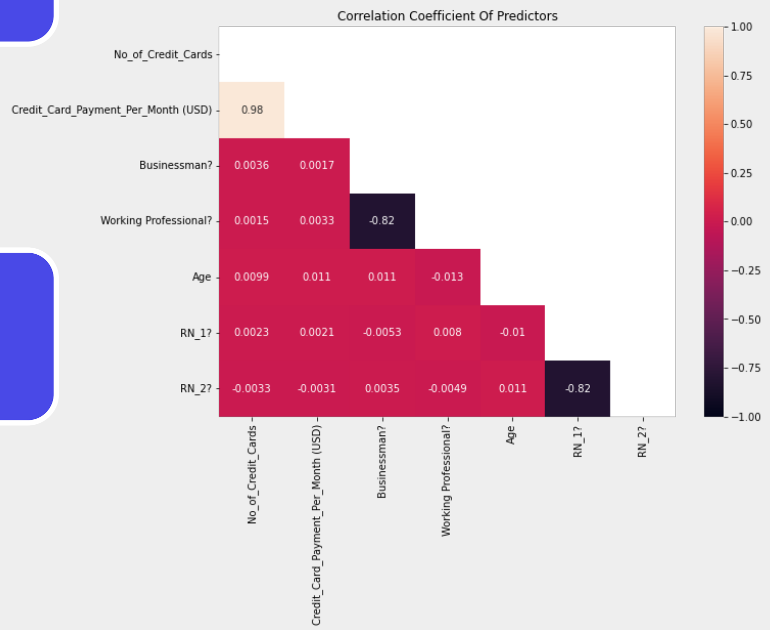
12000 rows × 2 columns

In [16]:
```python
ccpay='Credit_Card_Payment_Per_Month (USD)'
loanlimit='Upper_Loan_Limit (in thousands of USD)'
scaler = MinMaxScaler()
df[['Credit_Card_Payment_Per_Month (USD)_scaled']] = scaler.fit_transform(df[[ccpay]])
print(ccpay,':', scaler.data_min_,'min,',scaler.data_max_,'max')
df[['Upper_Loan_Limit (in thousands of USD)_scaled']] = scaler.fit_transform(df[[loanlimit]])
print(loanlimit,':', scaler.data_min_,'min,', scaler.data_max_,'max')
df
```

```
Credit_Card_Payment_Per_Month (USD) : [473.75241725] min, [1814.26340553] max
Upper_Loan_Limit (in thousands of USD) : [0.53995619] min, [173.80581301] max
```

Credit_Card_Payment_Per_Month (USD)_scaled	Upper_Loan_Limit (in thousands of USD)_scaled
0.504430	0.531023
0.654340	0.492108
0.554174	0.448142
0.107580	0.268893
0.543812	0.621941
...	...
0.070119	0.370733
0.055297	0.298411
0.525918	0.380473
0.001841	0.248742
0.539929	0.476990

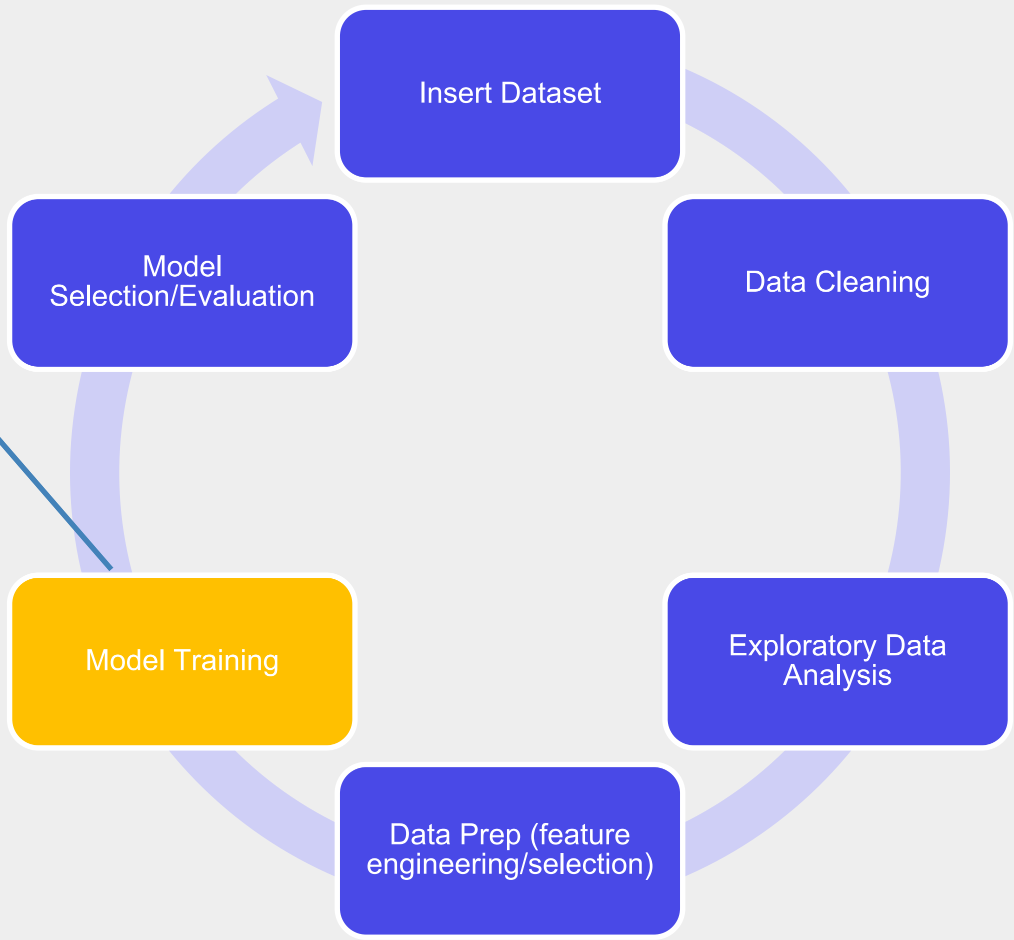




Feature Engineering:
- Scaling/Encoding various columns

Feature Selection:
- Remove non-scaled/encoded versions of variables
- # of CC and Monthly CC Pay

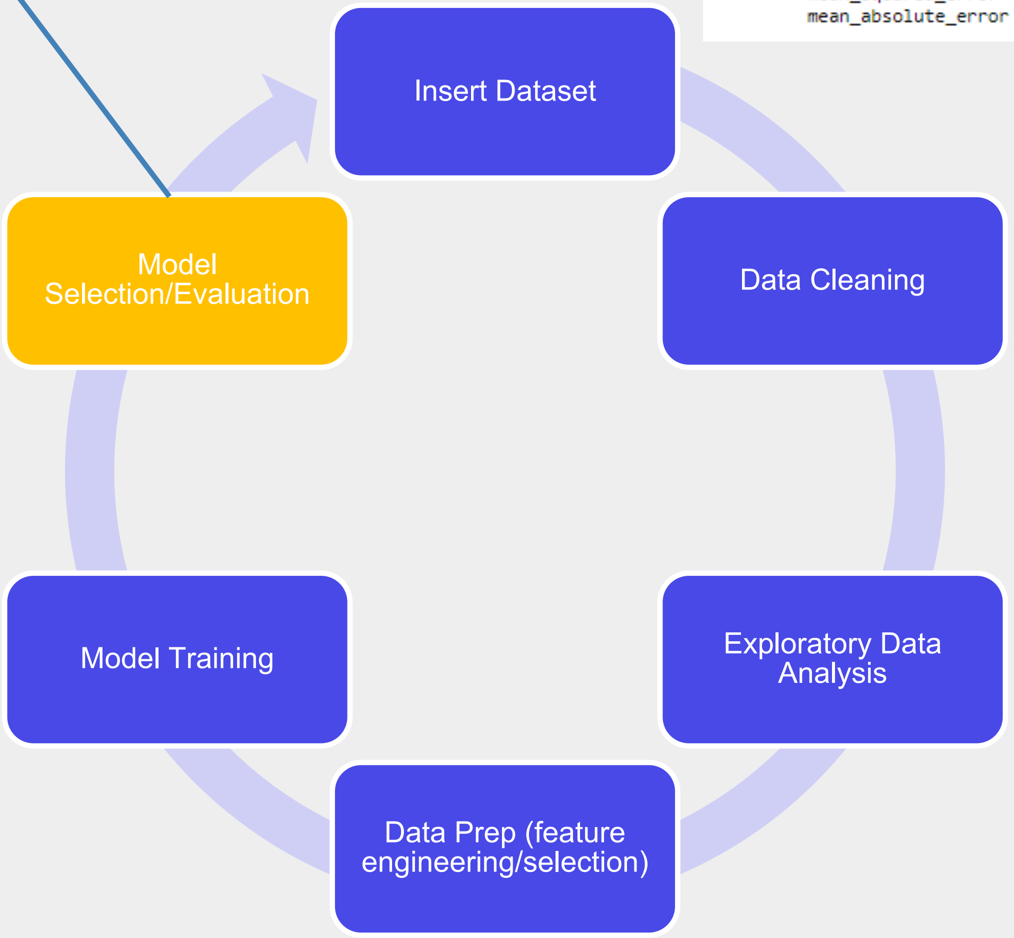
- 70-30 training-test split
- Random NumPy Seed: 603479
- Dropped non-scaled/encoded versions of columns




```
In [18]:  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=603479)
          model = LinearRegression(fit_intercept=True)
          model.fit(X_train, y_train)

Out[18]:  LinearRegression()
```

- Compare MSE/MAE of various models
- Rule: Improved MSE, improved model



In [19]:
```
predictions = model.predict(X_test)
print('Intercept:', model.intercept_)
print('Coefficients: ',model.coef_)
print('mean_squared_error : ', mean_squared_error(y_test, predictions))
print('mean_absolute_error : ', mean_absolute_error(y_test, predictions))
```
```
Intercept: 0.13651152283861379
Coefficients:  [0.06508629 0.20746799]
mean_squared_error :  0.007381988491060714
mean_absolute_error :  0.06716589655041498
```

In [32]:
```

model2 = LinearRegression()

model2.fit(x2_train, y2_train)

y2_pred = model2.predict(x2_test)
print('Intercept:', model2.intercept_)
print('Coefficients :', model2.coef_)
print('MSE =',mean_squared_error(y2_test, y2_pred))
print('MAE =',mean_absolute_error(y2_test, y2_pred))
```
```
Intercept: 0.01849284508855359
Coefficients : [0.06450817 0.20825765 0.13755598 0.12941389]
MSE = 0.005869914419459934
MAE = 0.06158971793846204
```

In [46]:
```
y3_pred = model3.predict(x3_test)
print('Intercept:', model3.intercept_)
print('Coefficients:', model3.coef_,)
print('MSE =',mean_squared_error(y3_test, y3_pred))
print('MAE =',mean_absolute_error(y3_test, y3_pred))
```
```
Intercept: -0.048242810602848796
Coefficients: [0.06616429 0.19758712 0.13615976 0.1293948  0.20980979]
MSE = 0.0018532880430741777
MAE = 0.03311307763073609
```

In [61]:
```
y4_pred = model4.predict(x4_test)
print('Intercept:', model4.intercept_)
print('Coefficients:', model4.coef_,)
print('MSE =',mean_squared_error(y4_test, y4_pred))
print('MAE =',mean_absolute_error(y4_test, y4_pred))
```
```
Intercept: -0.13983785845790248
Coefficients: [0.06525312 0.20289717 0.13557406 0.12923478 0.20878529 0.09379275
 0.11603866]
MSE = 0.0008276000174591291
MAE = 0.023817517777583067
```

```
In [20]: pi = pd.read_excel('Professional_Information.xlsx', index_col = 0, parse_dates = True)
         pi.head()
```

Out[20]:

	Unnamed: 1	Profession
0	NaN	Working_Professional
1	NaN	Working_Professional
2	NaN	Working_Professional
3	NaN	Businessman
4	NaN	Working_Professional

```
In [33]: bi = pd.read_excel('Biographical_Information.xlsx', index_col = 0, parse_dates = True)
         bi.head()
```

Out[33]:

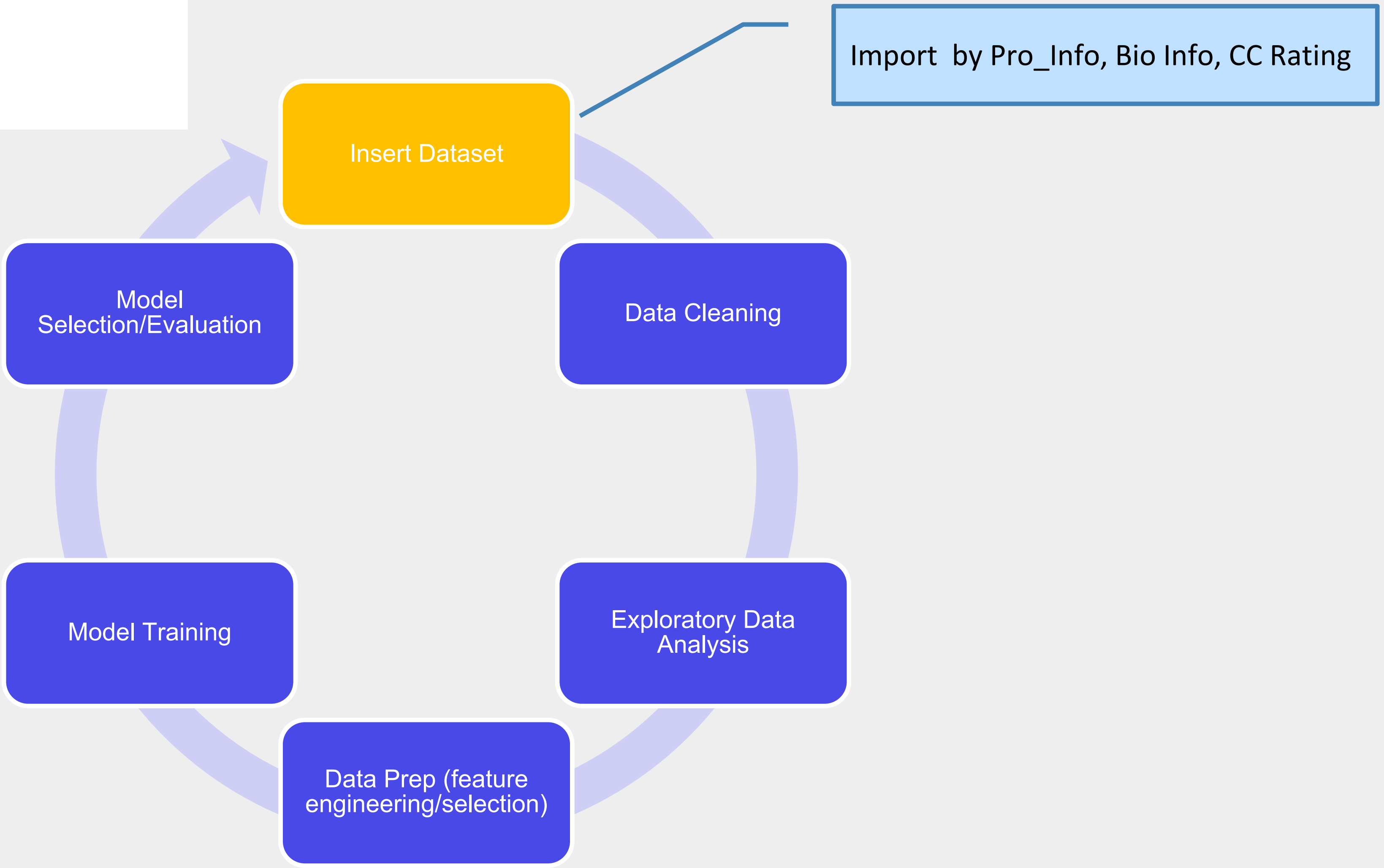
	Unnamed: 1	Age (Years)
0	NaN	60
1	NaN	25
2	NaN	30
3	NaN	25
4	NaN	55

```
In [47]: cr= pd.read_excel('Credit_Agency_Rating.xlsx', index_col=0)
         cr
```

Out[47]:

	Unnamed: 1	Rating
0	NaN	RN_1
1	NaN	RN_1
2	NaN	RN_1
3	NaN	RN_1
4	NaN	RN_2
...	...	...
11995	NaN	RN_2
11996	NaN	RN_2
11997	NaN	RN_1
11998	NaN	RN_1
11999	NaN	RN_1

12000 rows × 2 columns

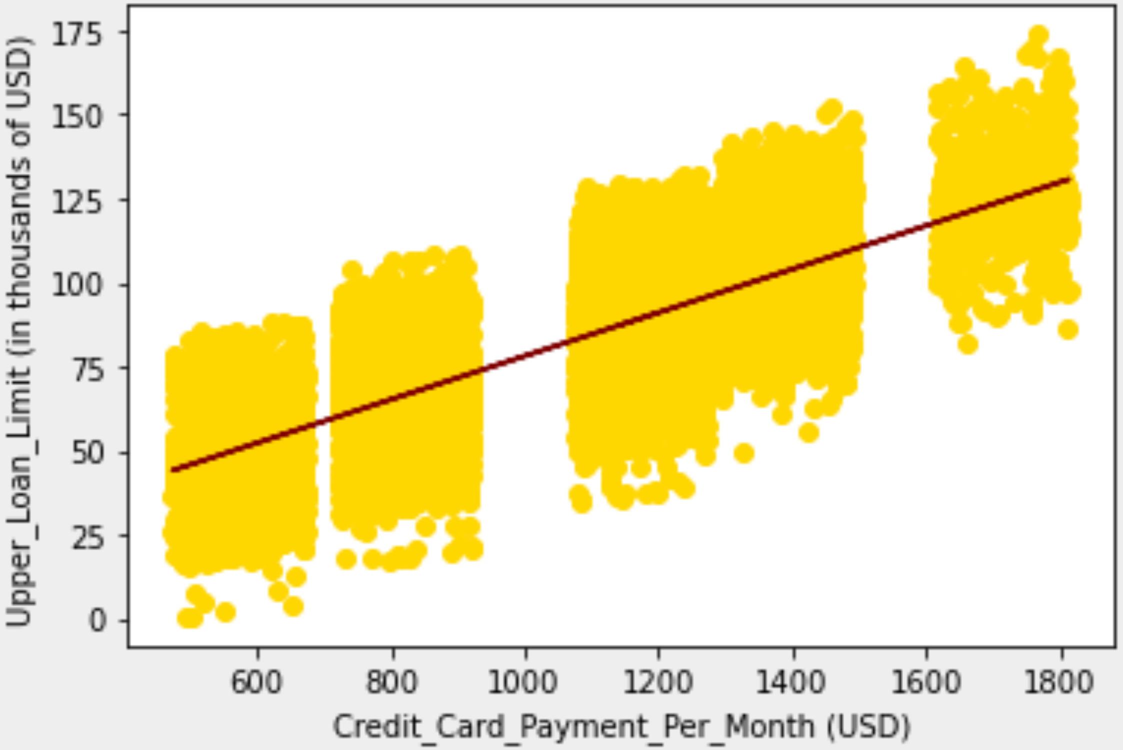


Data Overview

Exploratory Data Analysis

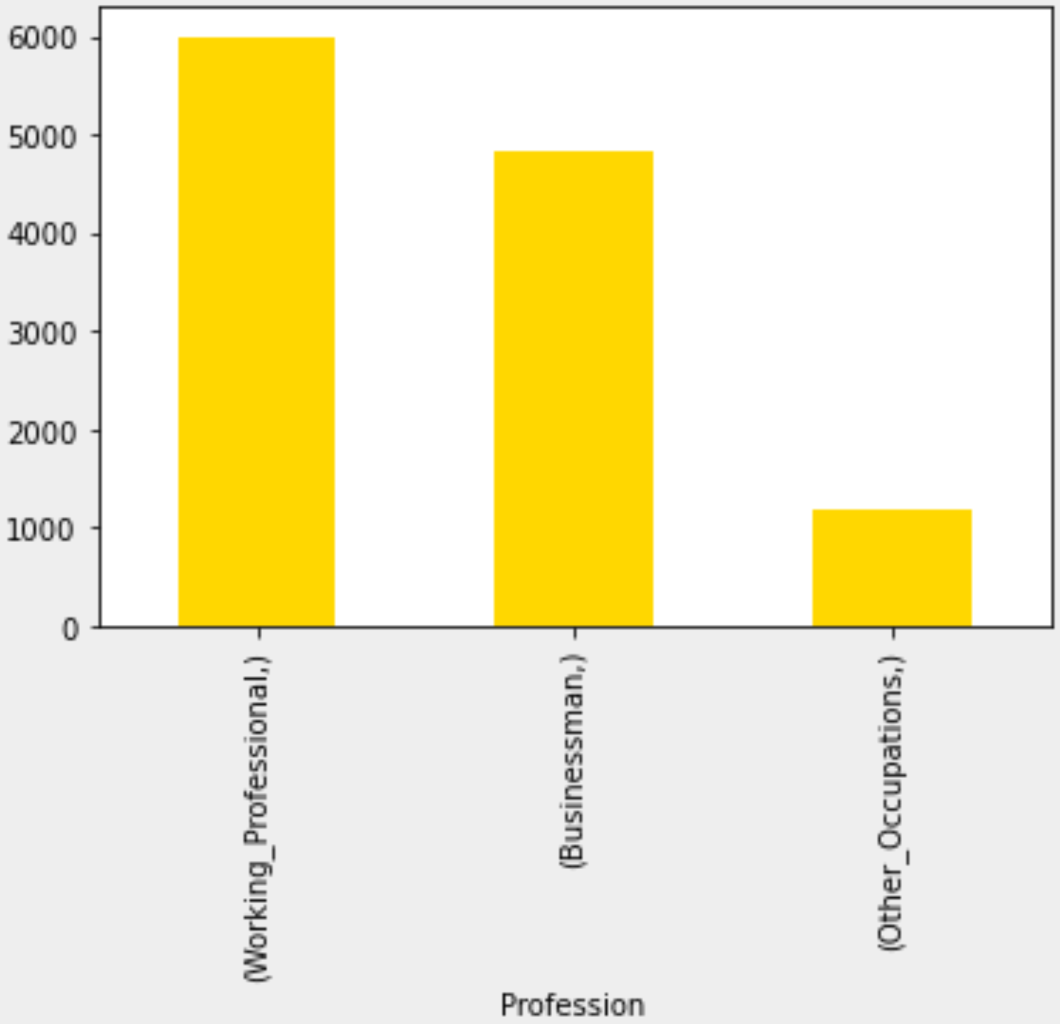
Data Dictionary

DataFrame	Feature	Description	Type	Sample data
Credit_Card_ Information	No_of_Credit_Cards	Number of credit cards owned	int	4
	Credit_Card_Payment_ Per_Month	Payment amount by credit card per month	float	1149.95
	Upper_Loan_Limit	Highest loan that can be awarded to an individual	float	92.55
Professional_ Information	Profession	Describes each individual's profession from either ‘businessman’, ‘working professional’, or ‘other occupations’	string	Working_ Professional
Biographical_ Information	Age	Describes the age of each individual	int	60
Credit_Agency_ Rating	Rating	Distinguishes an individual's credit card rating from either ‘rating 1’, ‘rating 2’, or ‘other ratings’	string	RN_1

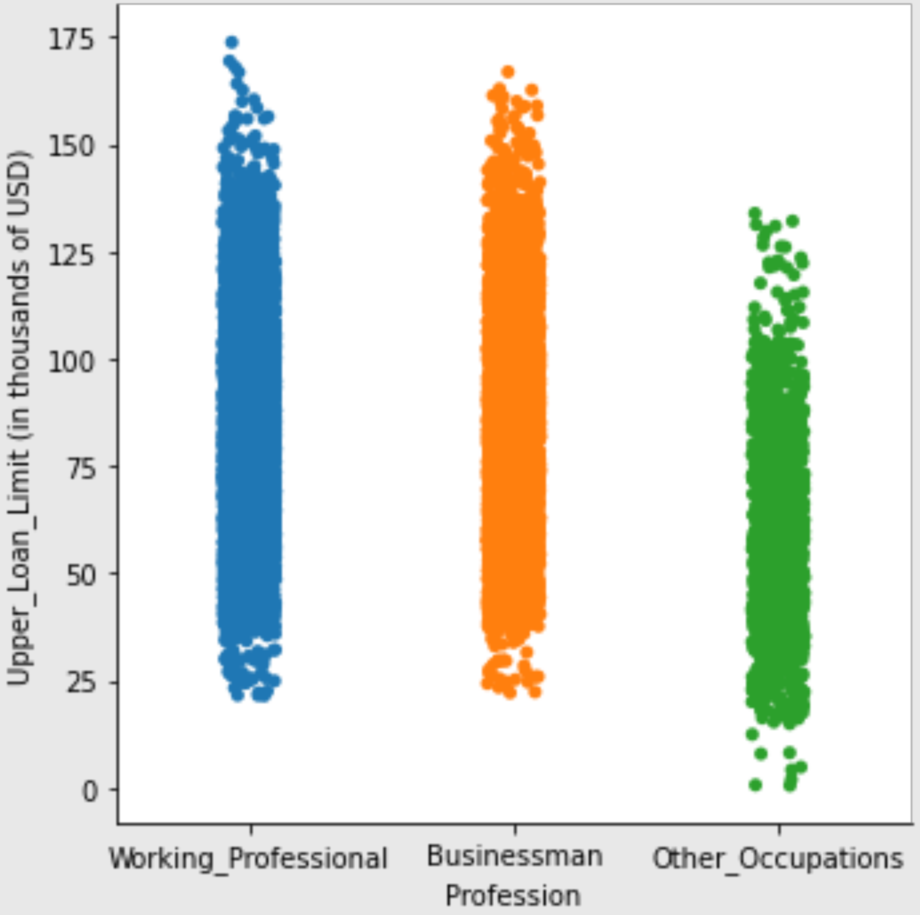


Positive relationship between monthly credit card payment and loan limit.

If your monthly payments are higher, you are more likely to get approved for a larger loan.

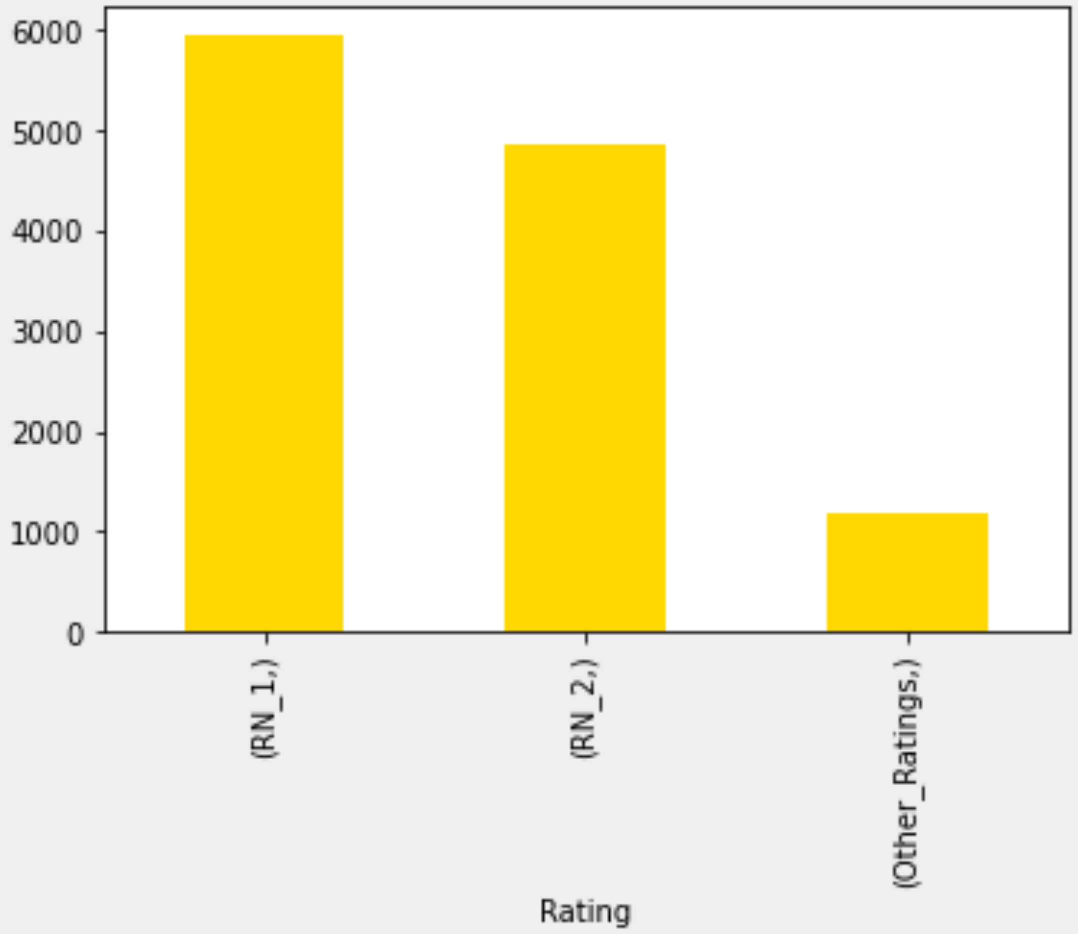


There appears to be three main types of occupations in our data set.

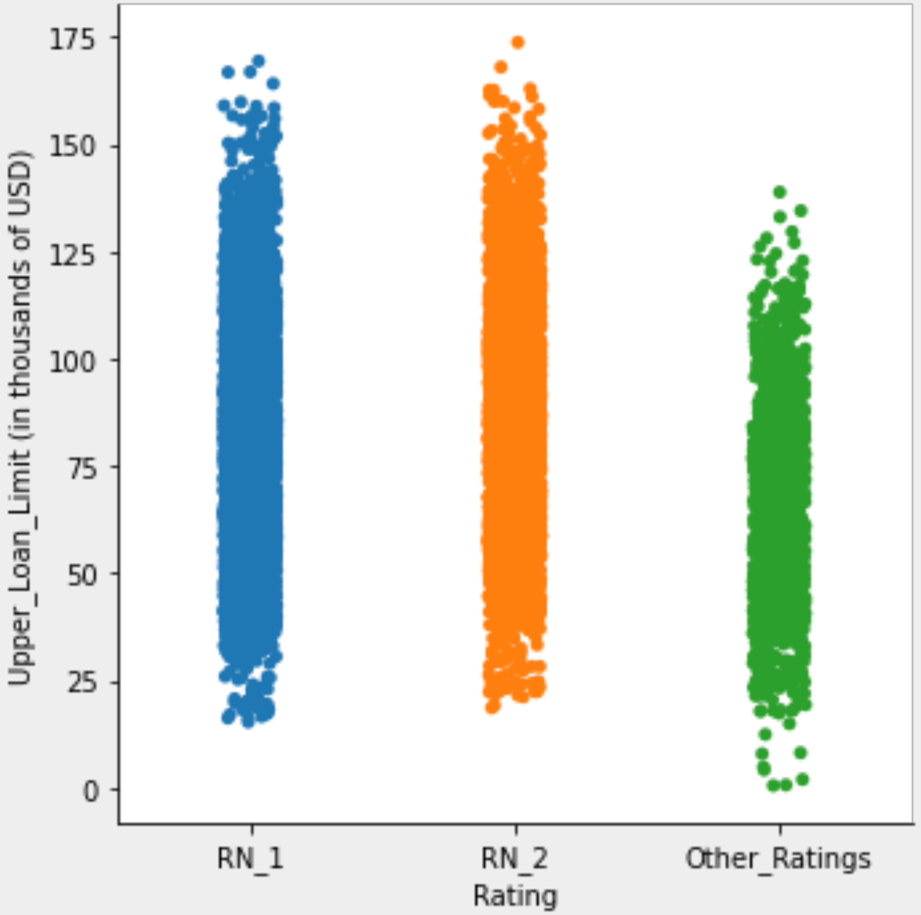


Both ***Working Professionals*** and ***Businessmen*** have roughly the same distribution of loan limits.

Other Occupations has a lower distribution of loan limits.

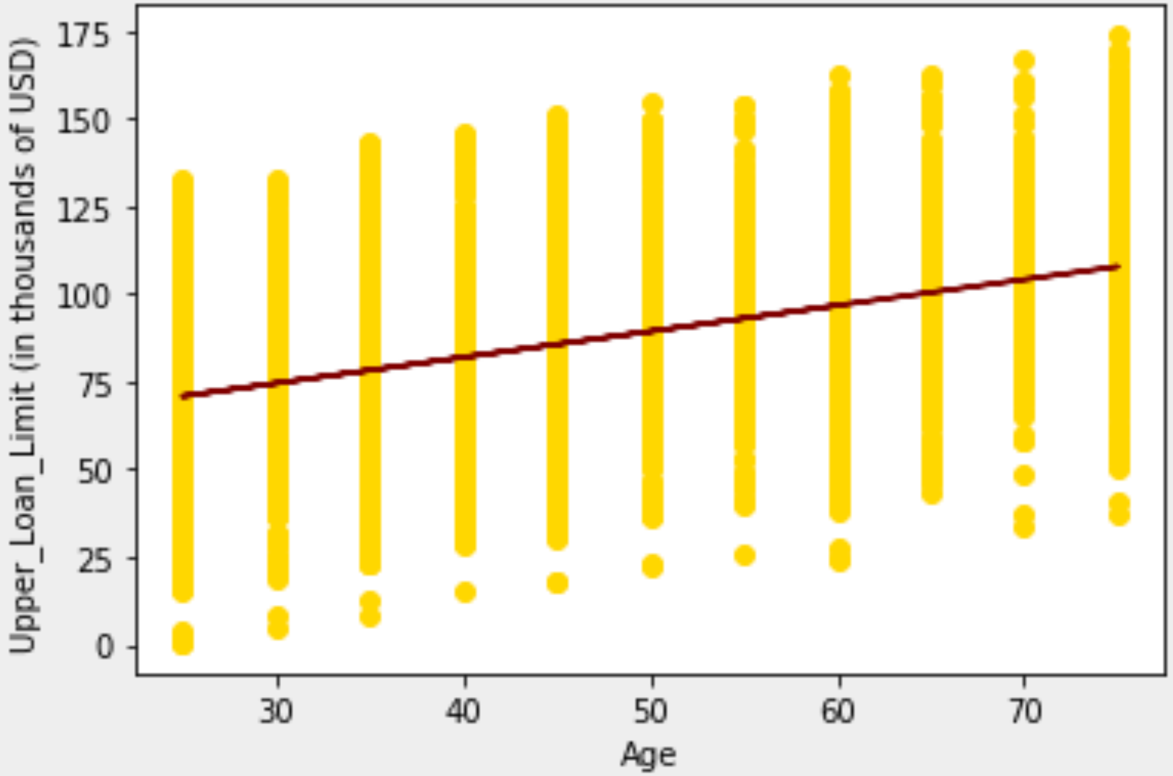


There appears to be three main types of credit ratings.



Both ***RN 1*** and ***RN 2*** have roughly the same distribution of loan limits.

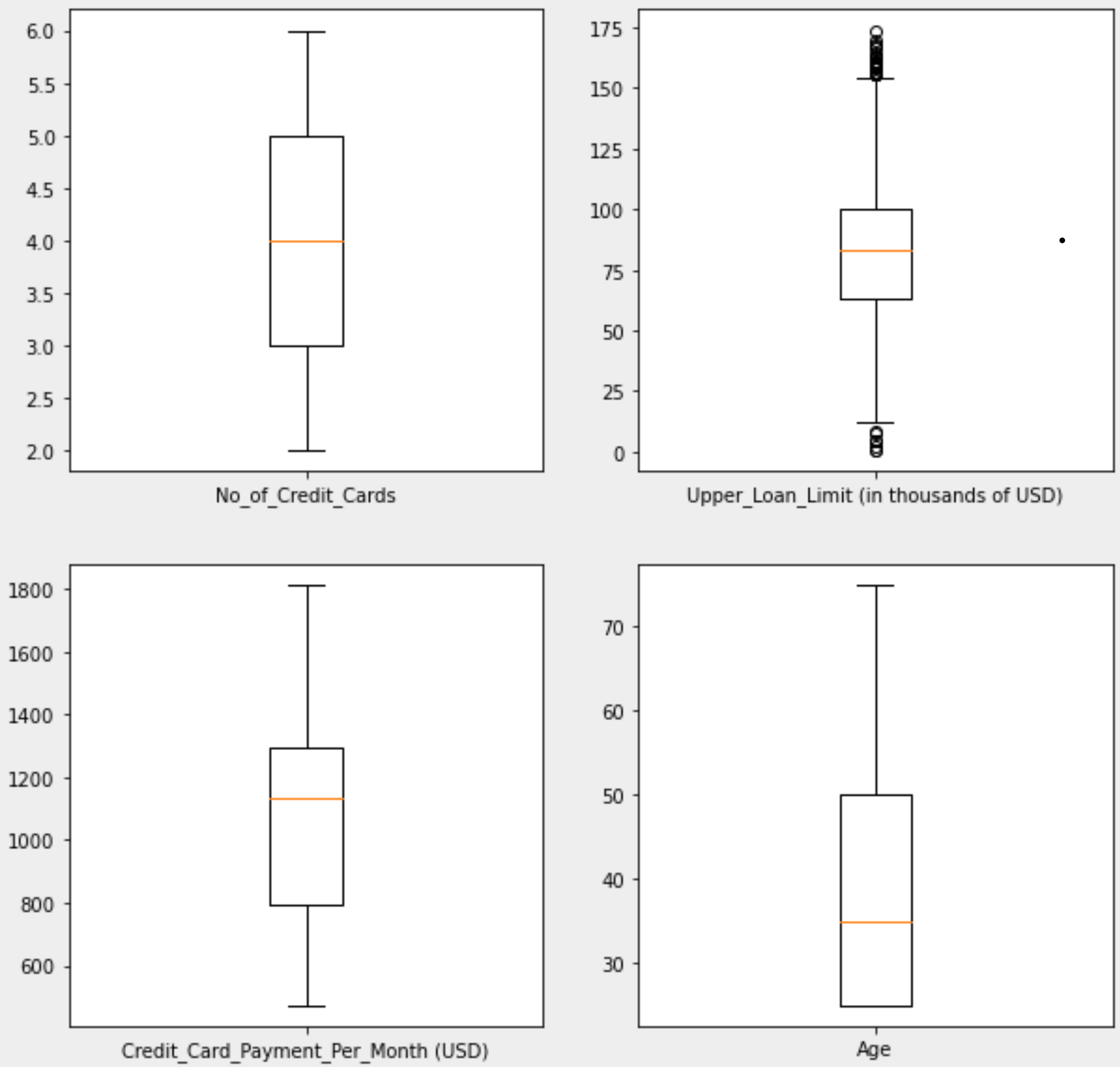
Other Ratings has a lower distribution of loan limits.



Positive relationship between age and loan limit.

If you are older, you are more likely to get approved for a larger loan.


Correlation Coefficient Of Predictors
No_of_Credit_Cards
Credit_Card_Payment_Per_Month (USD)
Businessman?
Working Professional?
Age
RN_1?
RN_2?
0.98
0.0036
0.0017
0.0015
0.0033
-0.82
0.0099
0.011
0.011
-0.013
0.0023
0.0021
-0.0053
0.008
-0.01
-0.0033
-0.0031
0.0035
-0.0049
0.011
-0.82
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00


6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
No_of_Credit_Cards
175
150
125
100
75
50
25
0
Upper_Loan_Limit (in thousands of USD)
1800
1600
1400
1200
1000
800
600
Credit_Card_Payment_Per_Month (USD)
70
60
50
40
30
Age

Feature Engineering

Numeric

- Scaling

Categorical

- Dummy Encoding
  - Two features with three possible responses (1,0) (0,1) (0,0)

Model Overview

Models Considered

ML Model	Features	MSE	MAE
Multiple Linear Regression 1	# of Credit Cards, Monthly Payment	0.0074	0.067
Multiple Linear Regression 2	# of Credit Cards, Monthly Payment, Occupation	0.0059	0.062
Multiple Linear Regression 3	# of Credit Cards, Monthly Payment, Occupation, Age	0.0018	0.033
Multiple Linear Regression 4	# of Credit Cards, Monthly Payment, Occupation, Age, Credit Rating	0.00083	0.024

Models Deployed

- Multiple linear regression with 5 features
- We created a GUI/user interface based on our deployed model to determine loan approval/rejection

Features:

1. Number of credit cards
2. Amount of monthly credit card payments made to the bank
3. Occupation (Working Professional vs. Businessman)
4. Age
5. Credit rating (RN 1 vs. RN 2)

*Categorical variables

Results

Linear Regression Equation

Y = -0.140 + 0.065(nocc) + 0.203(ccpay_scaled) + 0.136(businessman) + 0.129(working professional) + 0.209(age_scaled) + 0.094(RN1) + 0.116(RN2)

Age and **CCPay** highest coefficients

NoCC was not scaled

- Coeff 0.06 indicates strong relationship with Y

All positive coefficients

```
Intercept: -0.1398378584579027
MSE = 0.000827600017459128 6
MAE = 0.02381751777758305 3
```

	0	1
0	No_of_Credit_Cards	0.065253
1	Credit_Card_Payment_Per_Month (USD)_scaled	0.202897
2	Businessman?	0.135574
3	Working Professional?	0.129235
4	Age_scaled	0.208785
5	RN_1?	0.093793
6	RN_2?	0.116039

Model Performance Analysis

- The categorical features (**Credit Rating** and **Occupation)** were not optimal, as we could only use 3 categories that each had a similar relationship to the output variable
- **Number of credit cards** and **monthly payments** were strongly correlated – presenting a multicollinearity issue
    - However, dropping **number of credit cards** did not appear to improve the model performance, so we decided to keep it in the model

	0	1
0	No_of_Credit_Cards	0.065253
1	Credit_Card_Payment_Per_Month (USD)_scaled	0.202897
2	Businessman?	0.135574
3	Working Professional?	0.129235
4	Age_scaled	0.208785
5	RN_1?	0.093793
6	RN_2?	0.116039

Recommendations and Next Steps

- Collect more value for features like profession or rating
- Consider more features: salary and income, credit utilization, length of credit history, recent inquiries, etc.
- Regularize the model: Ridge and Lasso
- Try different types of models: decision tree

- Try out our GUI!

Thank you