REPORT

INFO 7390 Assignment 2 – Fall 2017 (Lending Club dataset)
Professor - Sri Krishnamurthy

By - TEAM 03
Sonali Chaudhari
Madhumitha Prakash

Problem Statement

You are working at a bank and you are considering investing in Lending club. Since there are no standard models, you are expected to build prediction models that will help you predict the interest rates based on various parameters users would input.

Part 1: Data wrangling and exploratory data analysis
1(a)Data Download and pre-processing:

Data Download:

Steps:

Data Preprocessing:

Steps:

1. Loaded each programmatically downloaded CSVs in separate dataframe.
2. Concatenated each dataframe in a single dataframe 'loan_data'.

```
In [4]: loan_data.shape
Out[4]: (1524072, 151)
```

3. Going through each column and understanding the meaning of each column through the dictionary provided on the LC website 'LCDataDictionary.xlsx'
4. Getting rid of the columns have more than 70% missing data. During this process we get rid of the column 'id' which has all rows null.

```
In [7]: loan_df.shape
Out[7]: (1524072, 93)
```

5. Out of the 151 columns most of the columns are generated after the loans lending procedure has taken place. So these columns leak data from the future. We need columns data that potential customer provided to LC during their application.

Feature Engineering

6. Created a list 'f' a features that are important for and received during initial borrower's application.

```
In [52]: f
Out[52]: ['application_type',
          'fico_range_low',
          'fico_range_high',
          'emp_length',
          'dti',
          'annual_inc',
          'dti_joint',
          'annual_inc_joint',
          'grade',
          'sub_grade',
          'int_rate',
          'loan_amnt',
          'issue_d',
          'purpose',
          'addr_state',
          'zip_code']
```

7. Created a dataframe ‘df’ consisting of all the features in the list ‘f’ and generated a column ‘id’ for uniquely identifying each entry.

```
In [11]: del loan_data['id']
         s = loan_data.reset_index()
         s['index'] = s.index +1
         s=s.rename(columns = {'index':'id'})
```

```
In [16]: # df is the main dataframe required for working
         df = s[f]
```

8. Retrieved the count of nulls in each column in ‘df’ to deal with missing data in every column.

```
In [49]: df.isnull().sum()
Out[49]: id                       0
         application_type         1
         fico_range_low           1
         fico_range_high          1
         emp_length               1
         dti                    174
         annual_inc               5
         dti_joint          1503556
         annual_inc_joint   1503552
         grade                    1
         sub_grade                1
         int_rate                 1
         loan_amnt                1
         issue_d                  1
         purpose                  1
         addr_state               1
         zip_code                 2
         dtype: int64
```

9. Getting rid of the row having most of the columns null.

```
In [19]: df = df[df.application_type.notnull()]
```

10. Getting the dataframe to have consistent formats for each column entry and logically replacing the NaN values for some categorial columns

```
In [48]: df.head()
```

Out[48]:

id	application_type	fico_range_low	fico_range_high	emp_length	dti	annual_inc	dti_joint	annual_inc_joint	grade	sub_grade	int_rate	loan_a
1	Individual	735.0	739.0	10+ years	27.65	24000.0	NaN	NaN	B	B2	10.65%	5000.0
2	Individual	740.0	744.0	< 1 year	1.00	30000.0	NaN	NaN	C	C4	15.27%	2500.0
3	Individual	735.0	739.0	10+ years	8.72	12252.0	NaN	NaN	C	C5	15.96%	2400.0
4	Individual	690.0	694.0	10+ years	20.00	49200.0	NaN	NaN	C	C1	13.49%	10000.(
5	Individual	695.0	699.0	1 year	17.94	80000.0	NaN	NaN	B	B5	12.69%	3000.0

1. The columns like emp_length, int_rates, zipcode, dti… are of type object and has no consistent format

Emp Length

In [20]:
```
df['emp_length']=np.where(df['emp_length']=='10+ years','10',df['emp_length'])
df['emp_length']=np.where(df['emp_length']=='< 1 year','0',df['emp_length'])
df['emp_length']=np.where(df['emp_length']=='1 year','1',df['emp_length'])
df['emp_length'] = df['emp_length'].map(lambda x: x.rstrip('years'))
```

Zipcode

In [21]:
```
df['zip_code'] = df['zip_code'].astype(str).map(lambda x: x.rstrip('xx'))
```

Int Rate

In [27]:
```
df['int_rate'] = df['int_rate'].astype(str).map(lambda x: x.rstrip('%'))
df['int_rate'] = df['int_rate'].astype(float)
```

2. Replaced dti_joint, annual_inc_joint null values to '0' for the application_type 'Individual' since those application has no co-borrower and hence dti_joint, annual_inc_joint becomes 0.

In [63]:
```
df['dti_joint'] = np.where(df['application_type']=='Individual',0,df['dti_joint'])
```

In [65]:
```
df['annual_inc_joint'] = np.where(df['application_type']=='Individual',0,df['annual_inc_joint'])
```

3. Filled the remaining null values for annual_inc with the mode annual_inc value in a particular sub_grade column that the row belongs to.

annual_inc

In [74]:
```
part = df[['id','sub_grade', 'annual_inc']]
Mode = df['annual_inc'].mode()[0]
a = list(df['sub_grade'].unique())

for c in a:
    t = part[part.sub_grade == c]
    if( t.annual_inc.isnull().sum()==0):
            break
    else:
            if(t.shape == t[t.isnull().any(axis=1)].shape ):
                t['annual_inc'].fillna(Mode, inplace=True)
            else:
                c = t['annual_inc'].mode()[0]
                t['annual_inc'].fillna(c, inplace=True)
            part.loc[part.id.isin(t.id), ['annual_inc']] = t[['annual_inc']]
part['annual_inc'].fillna(part['annual_inc'].mode()[0], inplace = True)
df.loc[df.id.isin(part.id), ['annual_inc']] = part[['annual_inc']]
```

4. Similarly for dti_joint column

dti_joint filling

In [69]:
```
part = df[['dti_joint', 'sub_grade','id']][df.application_type=='Joint App']
Mode = df['dti_joint'].mode()[0]
a = list(df['sub_grade'].unique())

for c in a:
    t = part[part.sub_grade == c]
    if( t.dti_joint.isnull().sum()==0):
            break
    else:
            if(t.shape == t[t.isnull().any(axis=1)].shape ):
                t['dti_joint'].fillna(Mode, inplace=True)
            else:
                c = t['dti_joint'].mode()[0]
                t['dti_joint'].fillna(c, inplace=True)
            part.loc[part.id.isin(t.id), ['dti_joint']] = t[['dti_joint']]
part['dti_joint'].fillna(part['dti_joint'].mode()[0], inplace = True)
df.loc[df.id.isin(part.id), ['dti_joint']] = part[['dti_joint']]
```

5. There are columns ‘fico_range_high’ and ‘fico_range_low’; so generated a new column ‘fico_avg’ having the average of both the column values

Fico columns

In [39]:
```
df['fico_avg'] = (df['fico_range_high']+df['fico_range_low'])/2
```

6. Using Knn to predict values for ‘dti’ column where its null.

Knn for dti filling

In [44]:
```
base = 'id','dti','int_rate','dti_joint','annual_inc_joint','Year'
target_column = 'dti'

null = df[target_column].isnull()
not_null = ~null
num_miss = null.sum()
num_miss

X_train = df.loc[not_null, base].sample(frac = 1)
X=X_train[['id','int_rate','dti_joint','annual_inc_joint','Year']].values

y = X_train[target_column].astype(int)
y = y.values

Y_target =df.loc[null, base].sample(frac = 1)
Y =Y_target[['id','int_rate','dti_joint','annual_inc_joint','Year']].values
```

In [*]:
```
clf = neighbors.KNeighborsClassifier()
clf.fit(X,y)
accuracy = clf.score(X, y)
print(accuracy)
prediction = clf.predict(Y)

Y_target.loc[Y_target.dti.isnull(),'dti'] = prediction
df.loc[df.id.isin(Y_target.id), ['dti']] = Y_target[['dti']]
```

11. After the above cleaning checking if there is any null values.

1(b)Exploratory Data analysis: