ASSIGNMENT

DATA PREPROCESSING

AATHIL TA B160345CS

AKARSH JOICE B160148CS

Department of Computer Science & Engineering

DATA SET SELECTION:

We have selected the dataset of a bank for giving loans to their customers. The data has 615 rows and 13 columns. The attributes of the dataset are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History, and others.

Among all industries, financial domain has the most extensive use of analytics & data science methods. This data set would help us to get enough hold of working on data sets from banking companies, what challenges are faced, what strategies are used, which variables influence the outcome etc . we can make use of this dataset to automate loan allotting process, we can identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.

All the attributes are skewed ,so it's better to choose median as central measure,since we choose median as central measure,its better to use IQR or MAD(median Average Deviation).KDE plot is a better choice for visualizing since we have no idea on distribution of attributes.

Measures of Central Deviation

October 13, 2019

[1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
import pandas as pd
dframe = pd.read_csv("loan_data_set.csv")
```

[3]:
```
dframe= pd.read_csv("loan_data_set.csv")
dframe.ApplicantIncome.fillna(dframe.mean(),inplace=True)#by mean
print(dframe.ApplicantIncome.head())
```

```
0    5849
1    4583
2    3000
3    2583
4    6000
Name: ApplicantIncome, dtype: int64
```

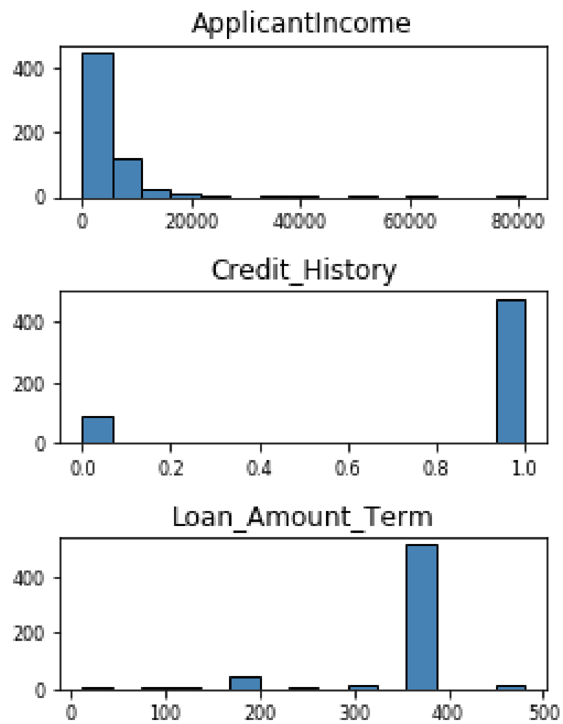
[4]:
```
dframe.mean()
```

[4]:
```
ApplicantIncome      5403.459283
CoapplicantIncome    1621.245798
LoanAmount            146.412162
Loan_Amount_Term      342.000000
Credit_History          0.842199
dtype: float64
```

[5]:
```
dframe.median()
```

[5]:
```
ApplicantIncome      3812.5
CoapplicantIncome    1188.5
LoanAmount            128.0
Loan_Amount_Term      360.0
Credit_History          1.0
dtype: float64
```

[6]:
```
dframe.hist(bins=15, color='steelblue', edgecolor='black', linewidth=1.0,
           xlabelsize=8, ylabelsize=8, grid=False)
plt.tight_layout(rect=(0, 0, 1.2, 1.2))
```



[7]:
```
dframe.dropna(inplace=True)
dframe.describe()
```

[7]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       480.000000         480.000000  480.000000        480.000000
mean       5364.231250        1581.093583  144.735417        342.050000
std        5668.251251        2617.692267   80.508164         65.212401
min         150.000000           0.000000    9.000000         36.000000
25%        2898.750000           0.000000  100.000000        360.000000
50%        3859.000000        1084.500000  128.000000        360.000000
75%        5852.500000        2253.250000  170.000000        360.000000
max       81000.000000       33837.000000  600.000000        480.000000

       Credit_History
count      480.000000
mean         0.854167
std          0.353307
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

[8]:
```
dframe.std()
```

[8]:

```
ApplicantIncome      5668.251251
CoapplicantIncome    2617.692267
LoanAmount             80.508164
Loan_Amount_Term       65.212401
Credit_History          0.353307
dtype: float64
```

[ ]:

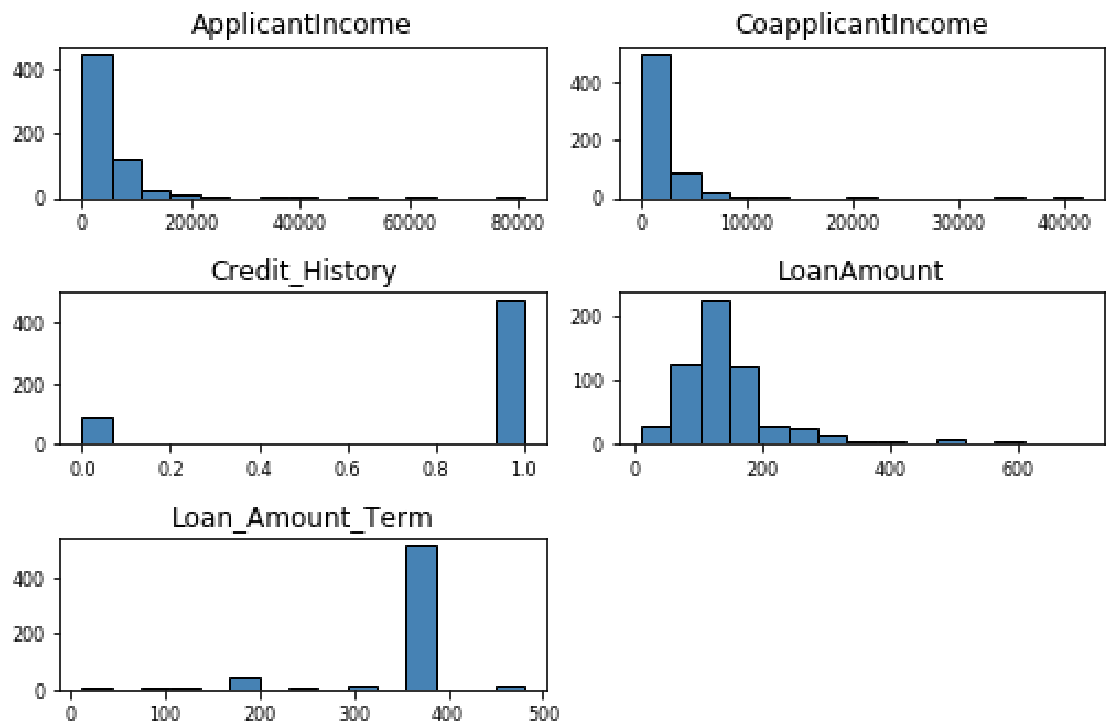
visualization: python

October 13, 2019

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import matplotlib as mpl
import numpy as np
import seaborn as sns
```

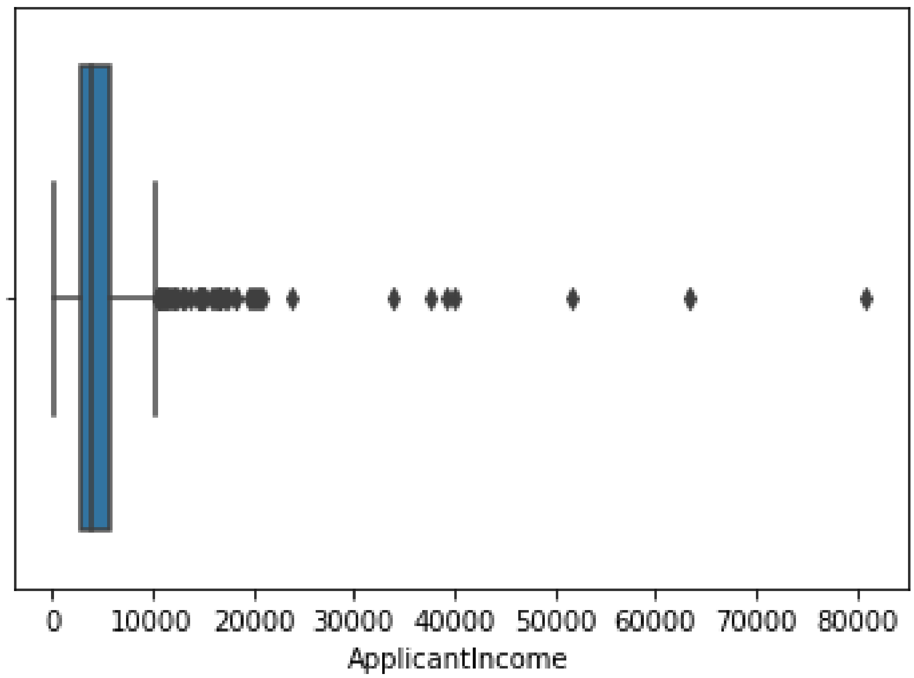
[2]:
```
df = pd.read_csv("loan_data_set.csv")
```

[3]:
```
df.hist(bins=15, color='steelblue', edgecolor='black', linewidth=1.0,
           xlabelsize=8, ylabelsize=8, grid=False)
plt.tight_layout(rect=(0, 0, 1.2, 1.2))
```

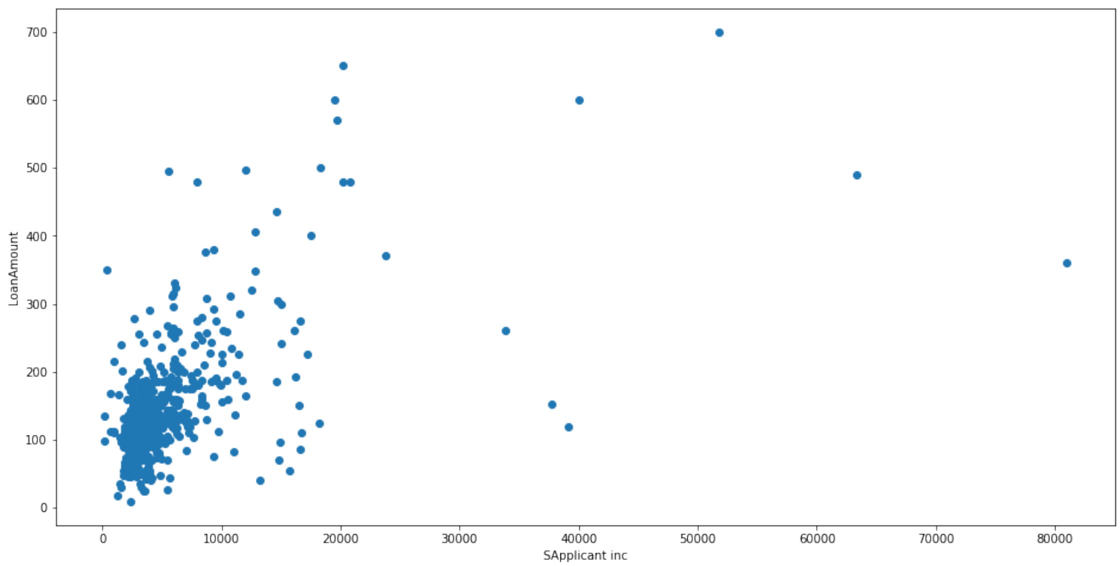


[4]:
```
#outliers...
sns.boxplot(x=df['ApplicantIncome'])
```

[4]: `<matplotlib.axes._subplots.AxesSubplot at 0x1a18fac710>`



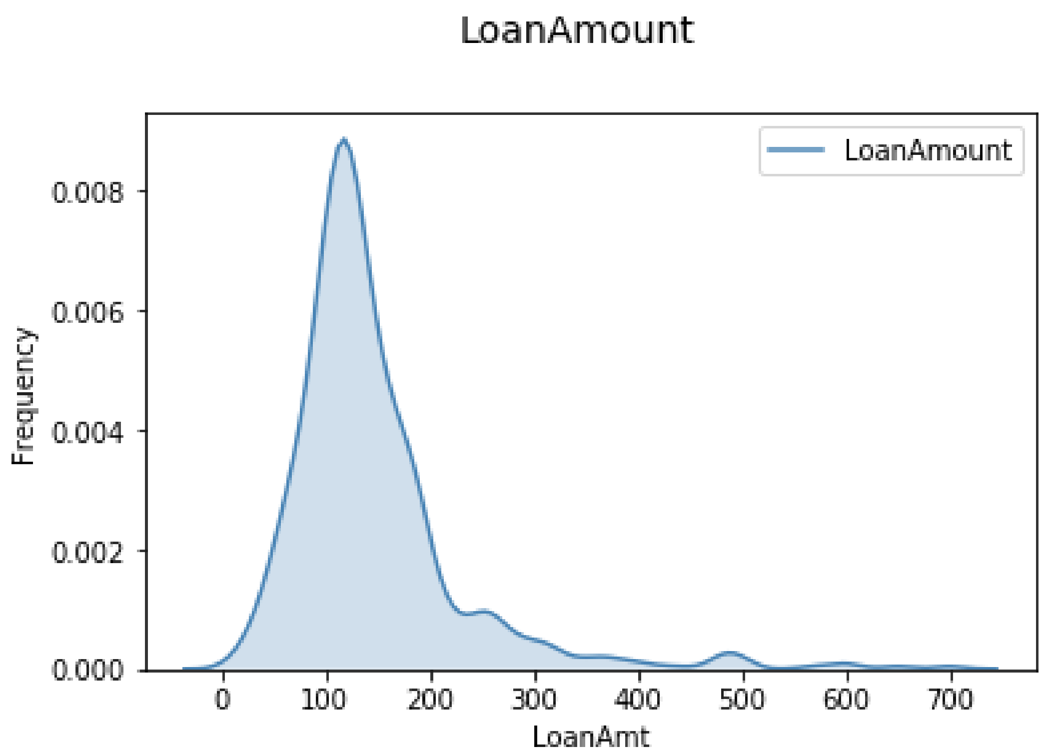
[5]:
```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(df['ApplicantIncome'], df['LoanAmount'])
ax.set_xlabel('SApplicant inc')
ax.set_ylabel('LoanAmount')
plt.show()
```



[6]:
```
fig = plt.figure(figsize = (6, 4))
title = fig.suptitle("LoanAmount", fontsize=14)
fig.subplots_adjust(top=0.85, wspace=0.3)

ax1 = fig.add_subplot(1,1, 1)
ax1.set_xlabel("LoanAmt")
ax1.set_ylabel("Frequency")
sns.kdeplot(df['LoanAmount'], ax=ax1, shade=True, color='steelblue')
```

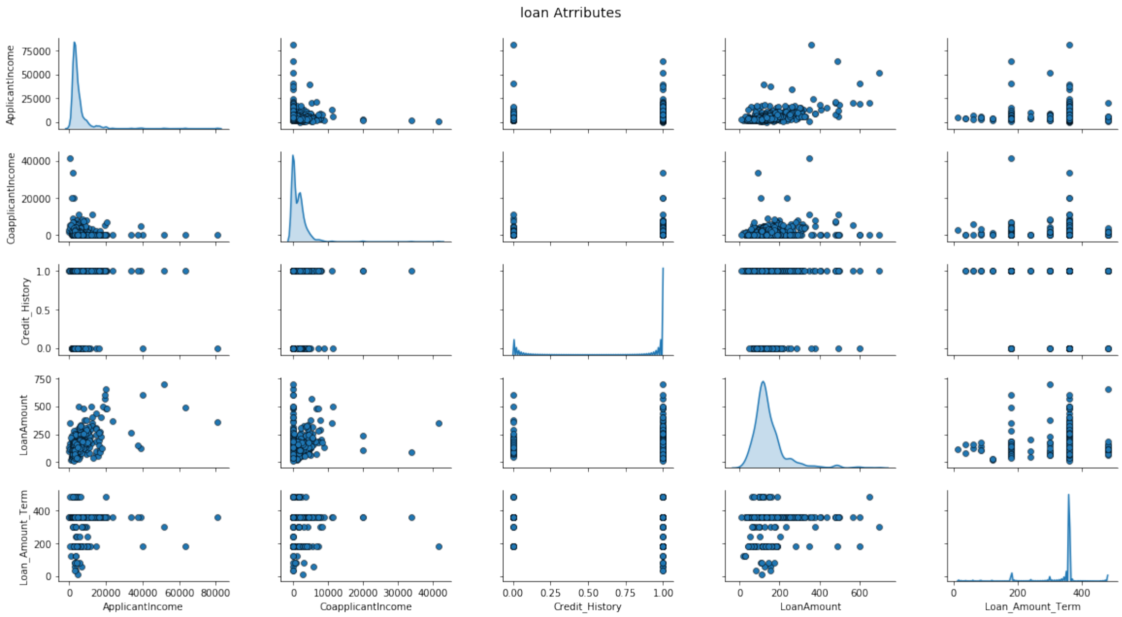
```
/Users/cluelessidiot/anaconda3/lib/python3.7/site-
packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid value
encountered in greater
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
/Users/cluelessidiot/anaconda3/lib/python3.7/site-
packages/statsmodels/nonparametric/kde.py:447: RuntimeWarning: invalid value
encountered in less
  X = X[np.logical_and(X > clip[0], X < clip[1])] # won't work for two columns.
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a199557b8>
```



[7]:
```
# Pair-wise Scatter Plots
cols = ['ApplicantIncome', 'CoapplicantIncome', 'Credit_History',␣
 ↪'LoanAmount','Loan_Amount_Term']
pp = sns.pairplot(df[cols], size=1.8, aspect=1.8,
                  plot_kws=dict(edgecolor="k", linewidth=0.5),
                  diag_kind="kde", diag_kws=dict(shade=True))

fig = pp.fig
fig.subplots_adjust(top=0.93, wspace=0.3)
t = fig.suptitle('loan Atrributes', fontsize=14)
```

```
/Users/cluelessidiot/anaconda3/lib/python3.7/site-
packages/seaborn/axisgrid.py:2065: UserWarning: The `size` parameter has been
renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```


loan Atrributes
ApplicantIncome
CoapplicantIncome
Credit_History
LoanAmount
Loan_Amount_Term

[ ]:

Removing Noise

October 13, 2019

[1]:
```
import pandas as pd
df = pd.read_csv("loan_data_set.csv")
#print (df.head())
```

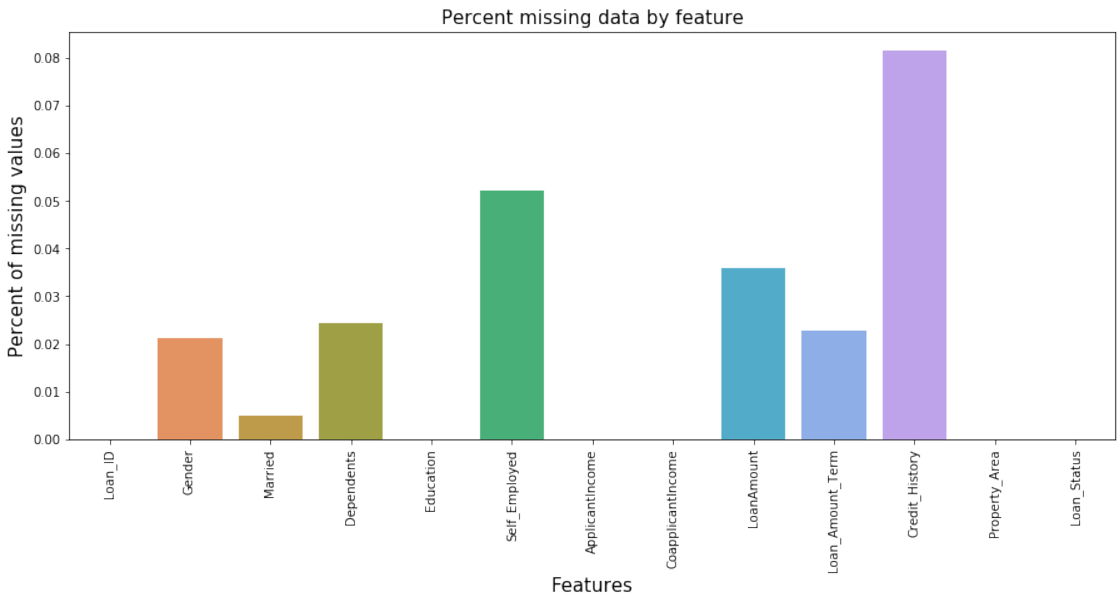
[16]:
```
#replacing all missing values by mean
#df.fillna(df.mean(),inplace=True)
#print (df.head())
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(15,8))
#sns.distplot(df.column_name, bins =30)

pmd=df
total=pmd.isnull().sum()
percent=(pmd.isnull().sum()/pmd.isnull().count())
missingData=pd.concat([total,percent],axis=1,keys=['Total','Percent'])
print (missingData)
f, ax = plt.subplots(figsize=(15, 6))
plt.xticks(rotation='90')
sns.barplot(x=missingData.index, y=missingData['Percent'])
plt.xlabel('Features', fontsize=15)
plt.ylabel('Percent of missing values', fontsize=15)
plt.title('Percent missing data by feature', fontsize=15)
missingData.head()
```

```
                   Total   Percent
Loan_ID                0  0.000000
Gender                13  0.021173
Married                3  0.004886
Dependents            15  0.024430
Education              0  0.000000
Self_Employed         32  0.052117
ApplicantIncome        0  0.000000
CoapplicantIncome      0  0.000000
LoanAmount            22  0.035831
Loan_Amount_Term      14  0.022801
Credit_History        50  0.081433
```

```
Property_Area           0  0.000000
Loan_Status             0  0.000000
```

[16]:
```
              Total   Percent
Loan_ID           0  0.000000
Gender           13  0.021173
Married           3  0.004886
Dependents       15  0.024430
Education         0  0.000000
```

```
<Figure size 1080x576 with 0 Axes>
```



[3]:
```
#data cleaning
#method 1->ignore the data row containing missing values..
dframe= pd.read_csv("loan_data_set.csv")
print(dframe['LoanAmount'].head())
dframe.isnull().count()
dframe.dropna(inplace=True)
print (dframe['LoanAmount'].head())
#dframe.count()
```

```
0      NaN
1    128.0
2     66.0
3    120.0
4    141.0
Name: LoanAmount, dtype: float64
```

```
1    128.0
2     66.0
3    120.0
4    141.0
5    267.0
Name: LoanAmount, dtype: float64
```

[4]:
```
#for back fill

dframe= pd.read_csv("loan_data_set.csv")
dframe.fillna(method='bfill',inplace=True)#for forward-fill
print(dframe['LoanAmount'].head())
dframe.fillna(method='ffill',inplace=True)#forbackward fill
```

```
0    128.0
1    128.0
2     66.0
3    120.0
4    141.0
Name: LoanAmount, dtype: float64
```

[5]:
```
#replace with constant
dframe= pd.read_csv("loan_data_set.csv")
dframe.LoanAmount.fillna(99,inplace=True)
print(dframe['LoanAmount'].head())
```

```
0     99.0
1    128.0
2     66.0
3    120.0
4    141.0
Name: LoanAmount, dtype: float64
```

[6]:
```
dframe= pd.read_csv("loan_data_set.csv")
dframe.LoanAmount.fillna(dframe.LoanAmount.mean(),inplace=True) #by mean
print(dframe.LoanAmount.head())
```

```
0    146.412162
1    128.000000
2     66.000000
3    120.000000
4    141.000000
Name: LoanAmount, dtype: float64
```

[7]:
```
dframe= pd.read_csv("loan_data_set.csv")
dframe.LoanAmount.fillna(dframe.LoanAmount.median(),inplace=True) #by median
```

```
print(dframe.LoanAmount.head())
```

```
0    128.0
1    128.0
2     66.0
3    120.0
4    141.0
Name: LoanAmount, dtype: float64
```

[9]:
```
#now we are gonna insert mean values but within +- standard deviation
import numpy as np
dframe= pd.read_csv("loan_data_set.csv")
LoanAmountAverage=dframe['LoanAmount'].mean()
LoanAmountStd=dframe['LoanAmount'].std()
LoanAmountNullCt=dframe['LoanAmount'].isnull().sum();
LoanAmountNullRl=np.random.
 ↪randint(LoanAmountAverage-LoanAmountStd,LoanAmountAverage+LoanAmountStd,size=LoanAmountNullCt
dframe['LoanAmount'][np.isnan(dframe['LoanAmount'])]=LoanAmountNullRl
print(dframe.LoanAmount.head())
dframe['LoanAmount'] = dframe['LoanAmount'].astype(int)
```

```
0     69.0
1    128.0
2     66.0
3    120.0
4    141.0
Name: LoanAmount, dtype: float64
```

```
/Users/cluelessidiot/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

[10]:
```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from numpy import nan
```

[11]:
```
df= pd.read_csv("loan_data_set.csv")
imp = IterativeImputer(max_iter=10, random_state=0)
imp.fit(df[['CoapplicantIncome','LoanAmount']])
IterativeImputer(add_indicator=False, estimator=None,
                 imputation_order='ascending', initial_strategy='mean',
                 max_iter=10, max_value=None, min_value=None,
                 missing_values=nan, n_nearest_features=None,
```

```
                    random_state=0, sample_posterior=False, tol=0.001,
                    verbose=0)
```

[11]:
```
IterativeImputer(add_indicator=False, estimator=None,
                 imputation_order='ascending', initial_strategy='mean',
                 max_iter=10, max_value=None, min_value=None,
                 missing_values=nan, n_nearest_features=None, random_state=0,
                 sample_posterior=False, tol=0.001, verbose=0)
```

[12]:
```
#print (df.head())
dq=imp.transform(df[['CoapplicantIncome','LoanAmount']])
```

[13]:
```
print (dq)
```

```
[[   0.          137.85795547]
 [1508.          128.        ]
 [   0.           66.        ]
 ...
 [ 240.          253.        ]
 [   0.          187.        ]
 [   0.          133.        ]]
```

[ ]:

[ ]:

Data Reduction

October 13, 2019

[1]:
```python
import pandas as pd
df = pd.read_csv("loan_data_set.csv")
#print (df.head())
```

[2]:
```python
from sklearn.preprocessing import StandardScaler
dframe= pd.read_csv("loan_data_set.csv")
dframe.LoanAmount.fillna(dframe.LoanAmount.median(),inplace=True)#by median
dframe.Loan_Amount_Term.fillna(dframe.Loan_Amount_Term.median(),inplace=True)#by␣
↪median
import matplotlib.pyplot as plt
```

[3]:
```python
features_for_pca=['ApplicantIncome','LoanAmount','CoapplicantIncome','Loan_Amount_Term']
```

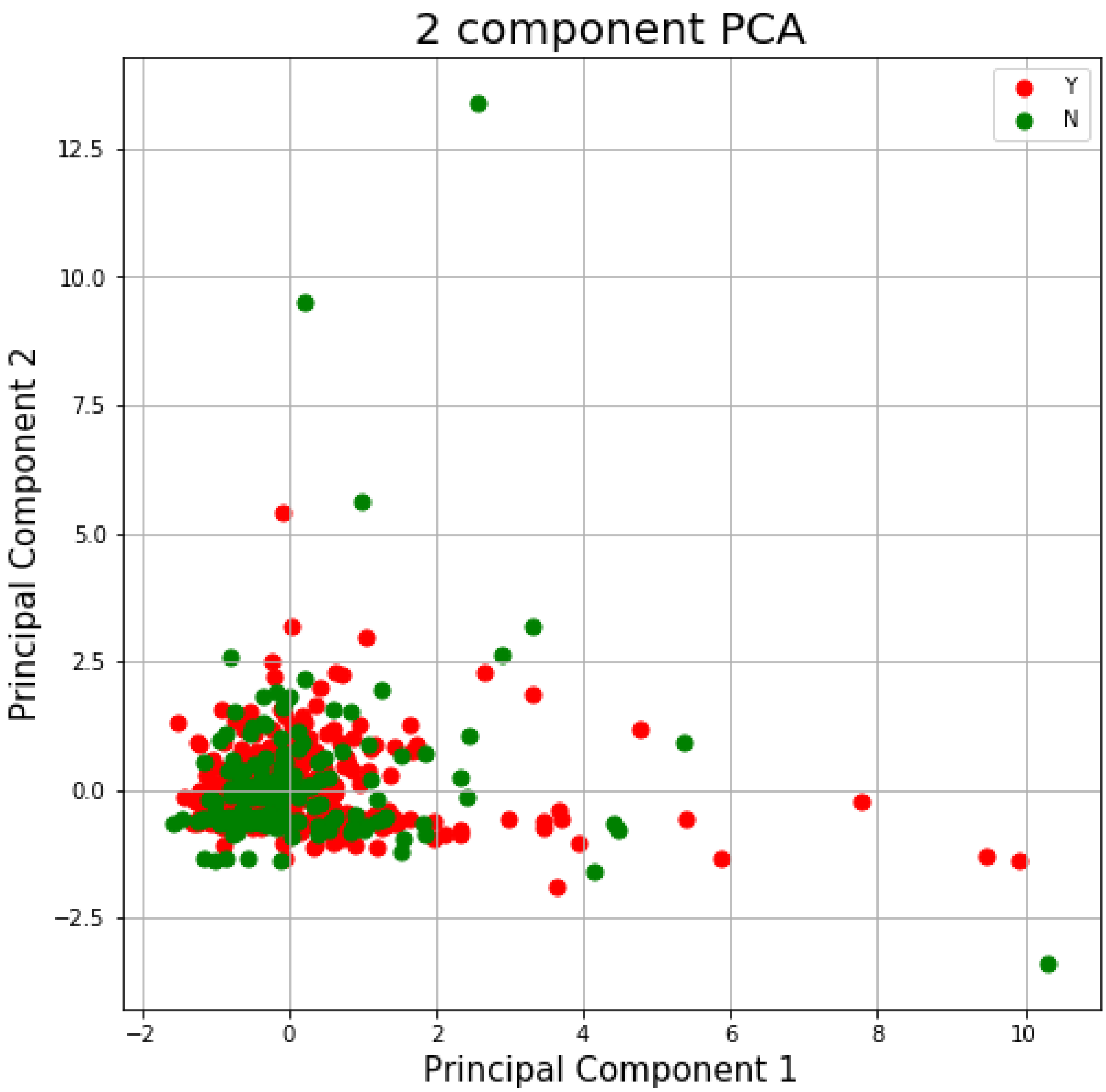
[23]:
```python
# Separating out the features
x = dframe.loc[:, features_for_pca].values
#print (x)
y = dframe.loc[:,['Loan_Status']].values
print (x)
x = StandardScaler().fit_transform(x)
print (x)
#normalization....... of selected attributes
```

```
[[5849.  128.    0.  360.]
 [4583.  128. 1508.  360.]
 [3000.   66.    0.  360.]
 ...
 [8072.  253.  240.  360.]
 [7583.  187.    0.  360.]
 [4583.  133.    0.  360.]]
[[ 0.07299082 -0.21124125 -0.55448733  0.2732313 ]
 [-0.13441195 -0.21124125 -0.03873155  0.2732313 ]
 [-0.39374734 -0.94899647 -0.55448733  0.2732313 ]
 ...
 [ 0.43717437  1.27616847 -0.47240418  0.2732313 ]
 [ 0.35706382  0.49081614 -0.55448733  0.2732313 ]
 [-0.13441195 -0.15174486 -0.55448733  0.2732313 ]]
```

[24]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)

principalDf = pd.DataFrame(data = principalComponents, columns = ['principal␣
↪component 1', 'principal component 2'])
#print (principalDf)
```

[6]:
```
finalDf = pd.concat([principalDf, dframe[['Loan_Status']]], axis = 1)
#print (finalDf)
```

[7]:
```
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Y','N']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['Loan_Status'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1'], finalDf.
↪loc[indicesToKeep, 'principal component 2'], c = color, s = 50)
ax.legend(targets)
ax.grid()
```

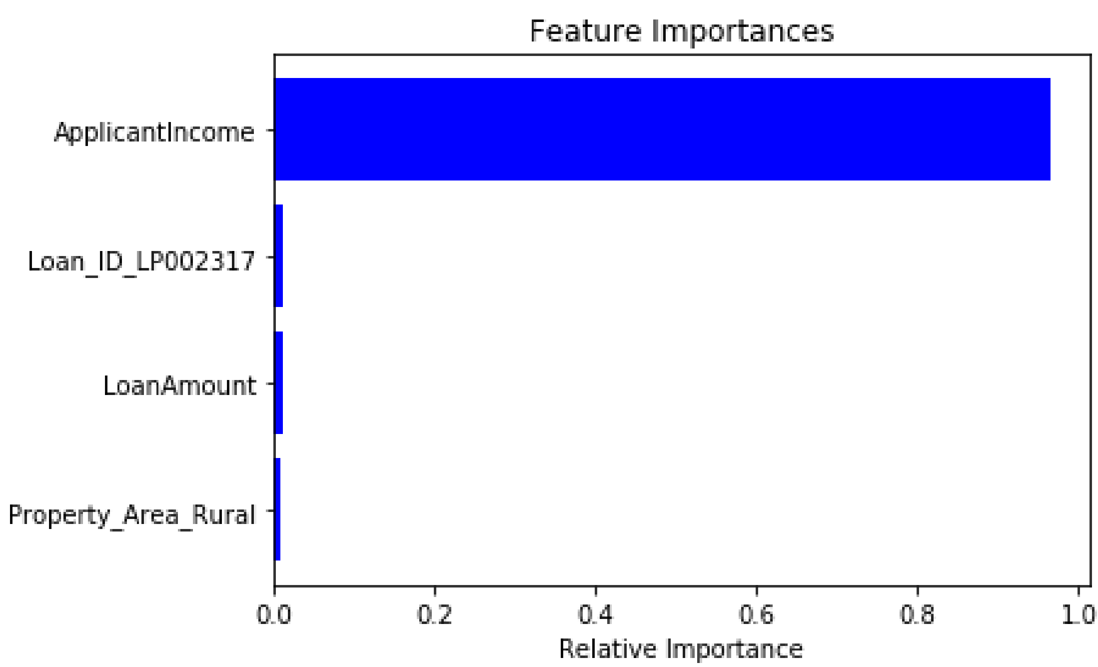


[13]:
```
from sklearn.ensemble import RandomForestRegressor
import numpy as np
df = pd.read_csv("loan_data_set.csv")
qf=pd.read_csv("loan_data_set.csv")

df.dropna(inplace=True)
conv_dict={'N':1.,'Y':2}
df['Loan_Status']=df.Loan_Status.apply(conv_dict.get)
#df=df.drop([ 'LoanAmount'], axis=0)
#df.LoanAmount.fillna(df.LoanAmount.median(),inplace=True)#by median
model = RandomForestRegressor(random_state=1, max_depth=3)
df=pd.get_dummies(df)
model.fit(df,df.ApplicantIncome)
```

```
/Users/cluelessidiot/anaconda3/lib/python3.7/site-
packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of
n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

[13]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=3,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=1, verbose=0,
                      warm_start=False)
```

[14]:
```
features = df.columns
importances = model.feature_importances_
indices = np.argsort(importances)[-4:]  # top 10 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



[15]:
```
df=pd.read_csv("loan_data_set.csv")
df.corr()
```

[15]:
```
                   ApplicantIncome  CoapplicantIncome  LoanAmount  \
ApplicantIncome           1.000000          -0.116605    0.570909
```

```
CoapplicantIncome        -0.116605           1.000000    0.188619
LoanAmount                0.570909           0.188619    1.000000
Loan_Amount_Term         -0.045306          -0.059878    0.039447
Credit_History           -0.014715          -0.002056   -0.008433

                   Loan_Amount_Term  Credit_History
ApplicantIncome           -0.045306       -0.014715
CoapplicantIncome         -0.059878       -0.002056
LoanAmount                 0.039447       -0.008433
Loan_Amount_Term           1.000000        0.001470
Credit_History             0.001470        1.000000
```

[16]:
```
#from above table ,correleation of .5 greater either one attributes is selected
```

[18]:
```
from sklearn.feature_selection import f_regression
df=pd.read_csv("loan_data_set.csv")
qf=df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History','
df=df[['ApplicantIncome','CoapplicantIncome','LoanAmount','Loan_Amount_Term','Credit_History']]
conv_dict={'N':1.,'Y':2}
qf['Loan_Status']=qf.Loan_Status.apply(conv_dict.get)
df.dropna(inplace=True)
qf.dropna(inplace=True)
ffs = f_regression(df,qf.Loan_Status)
```

[19]:
```
variable = [ ]
for i in range(0,len(df.columns)-1):
    print (ffs[0][i])
    if ffs[0][i] >=.4:
       variable.append(df.columns[i])
```

```
0.02079342985136059
0.9923690484110282
0.7085056246888057
0.4314605380860428
```

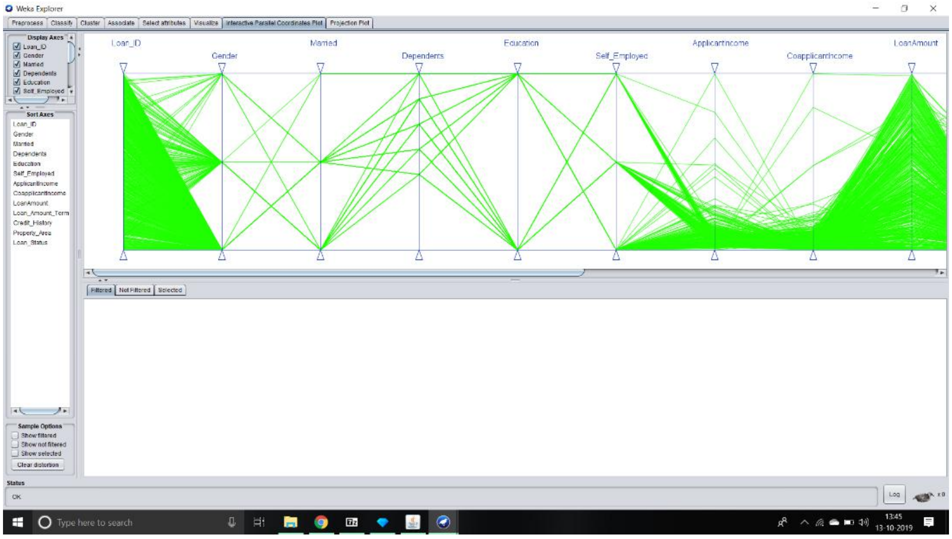
[20]:
```
print (variable)
```

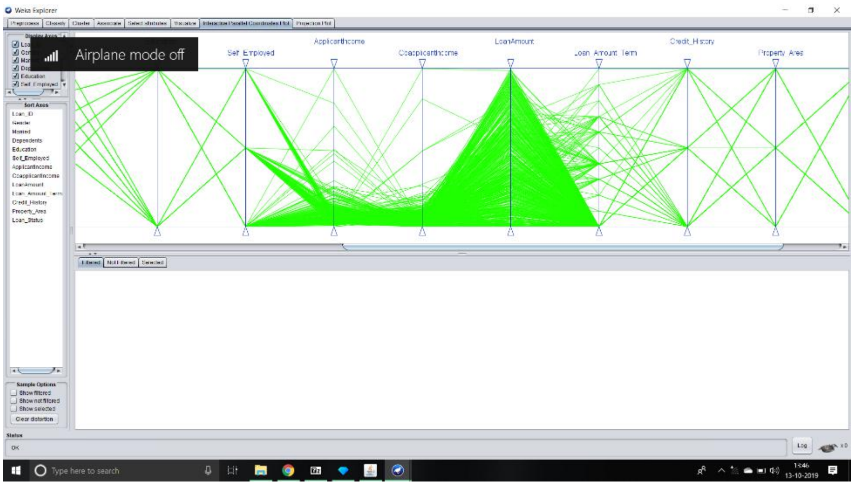
```
['CoapplicantIncome', 'LoanAmount', 'Loan_Amount_Term']
```

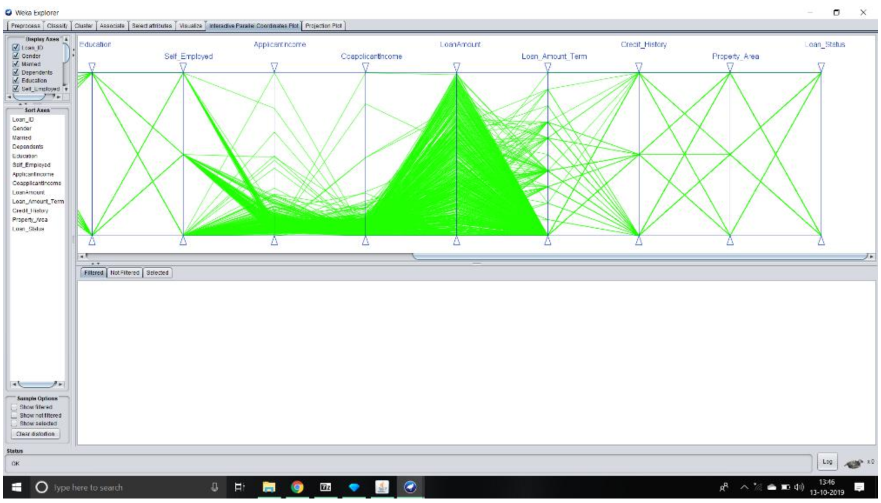
[ ]:
```
```

DATA VISUALISATION:

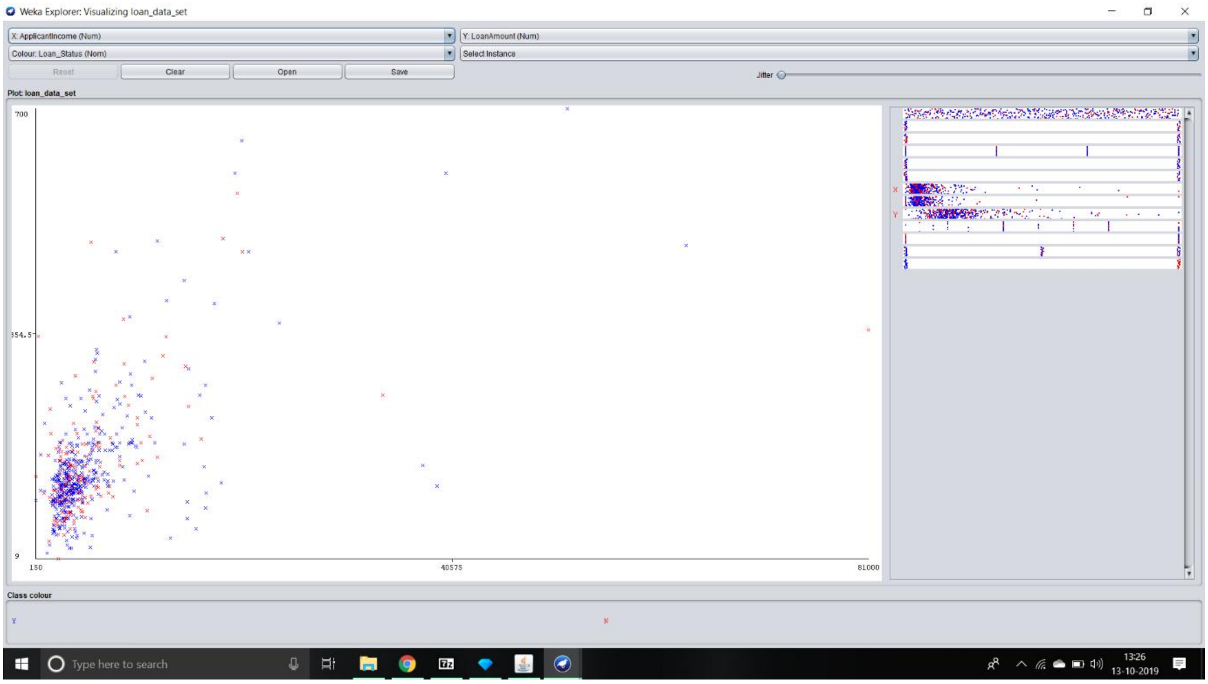
1.PARALLEL COORDINATES:



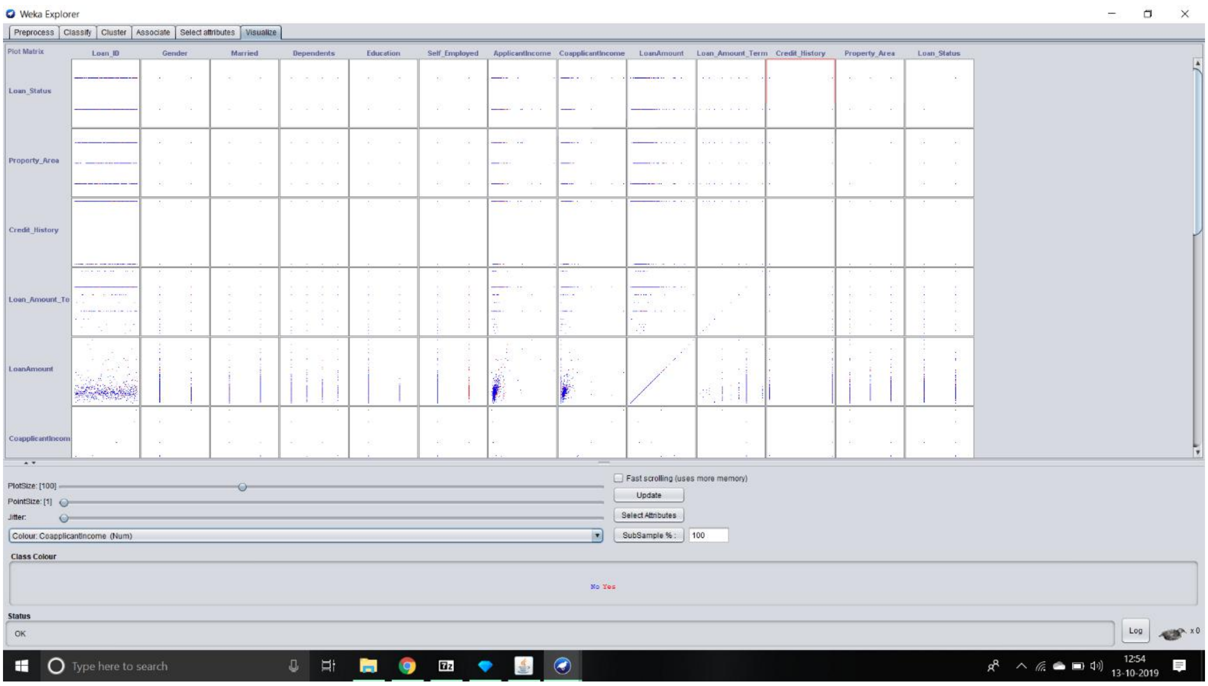




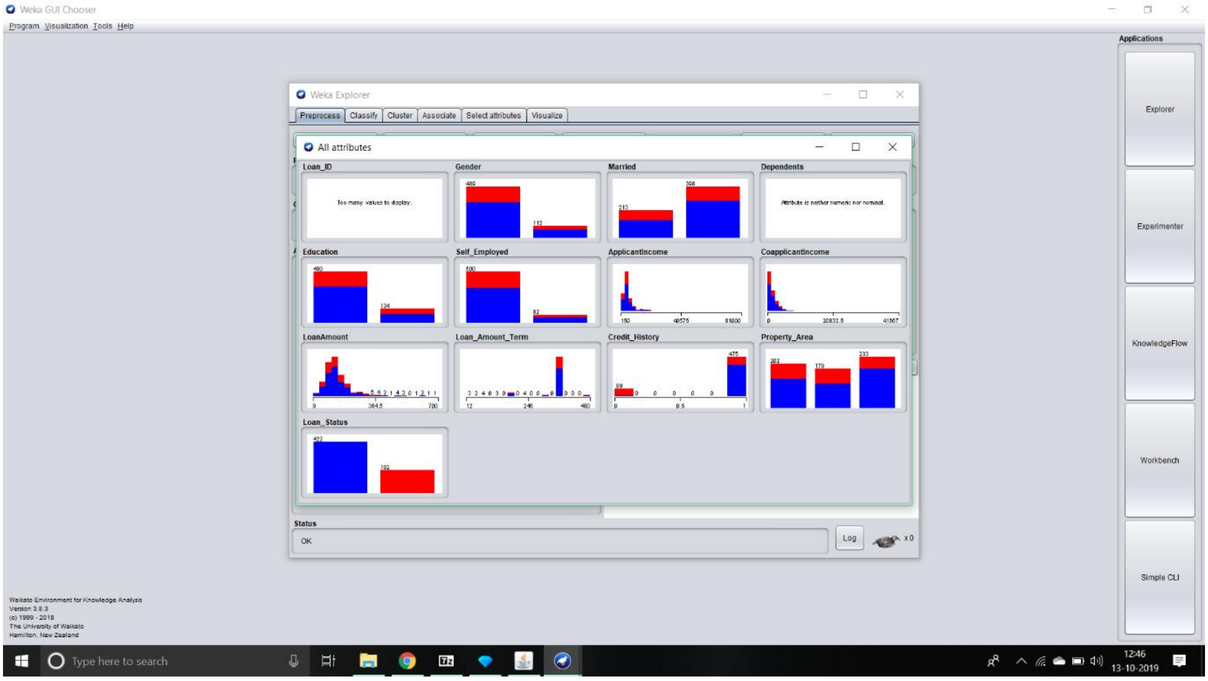
2.SCATTER PLOT:

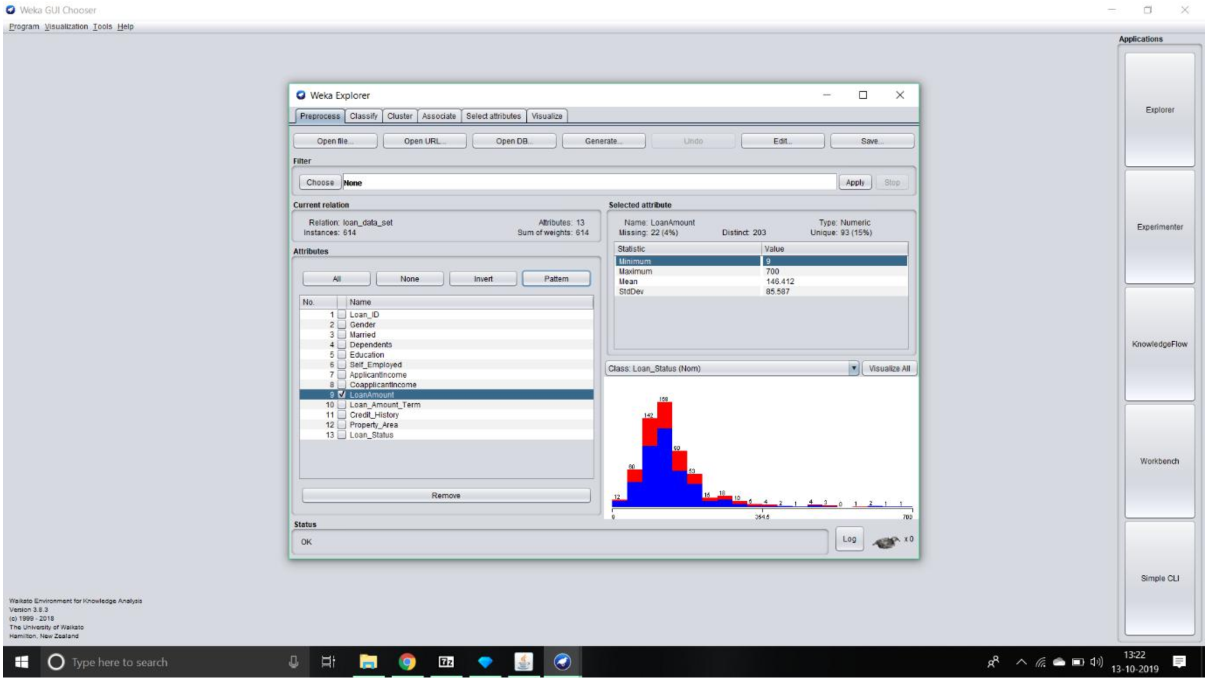


3.SCATTER MATRIX:



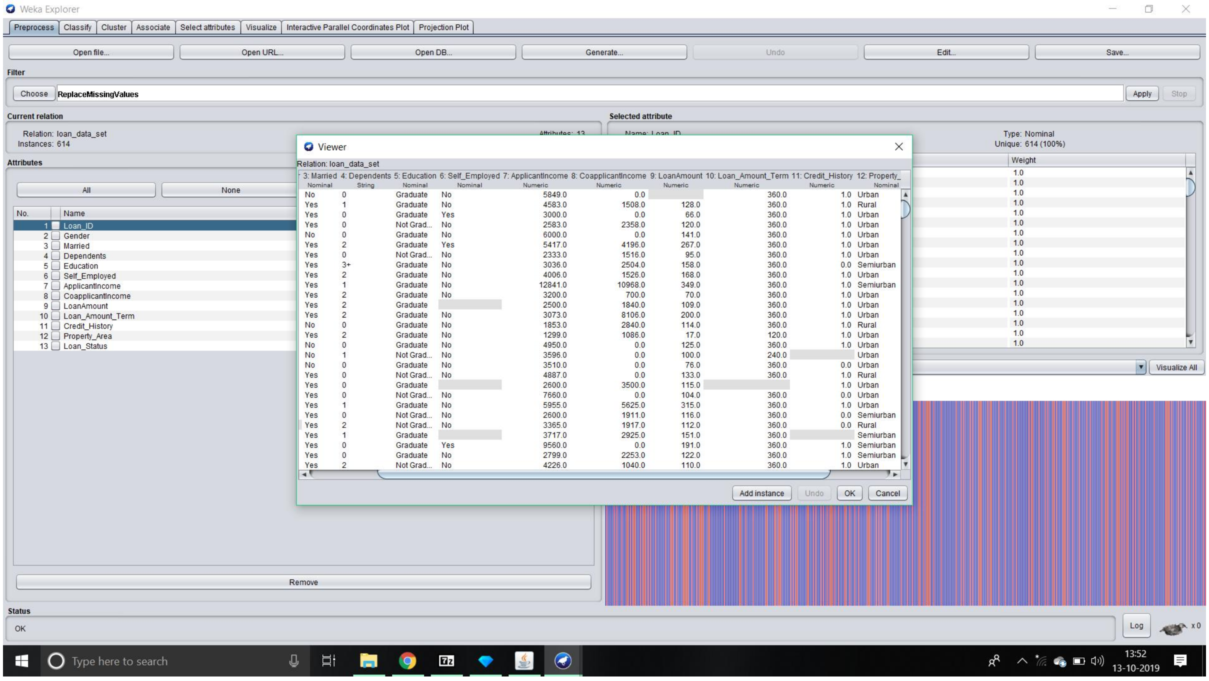
4.HISTOGRAM:



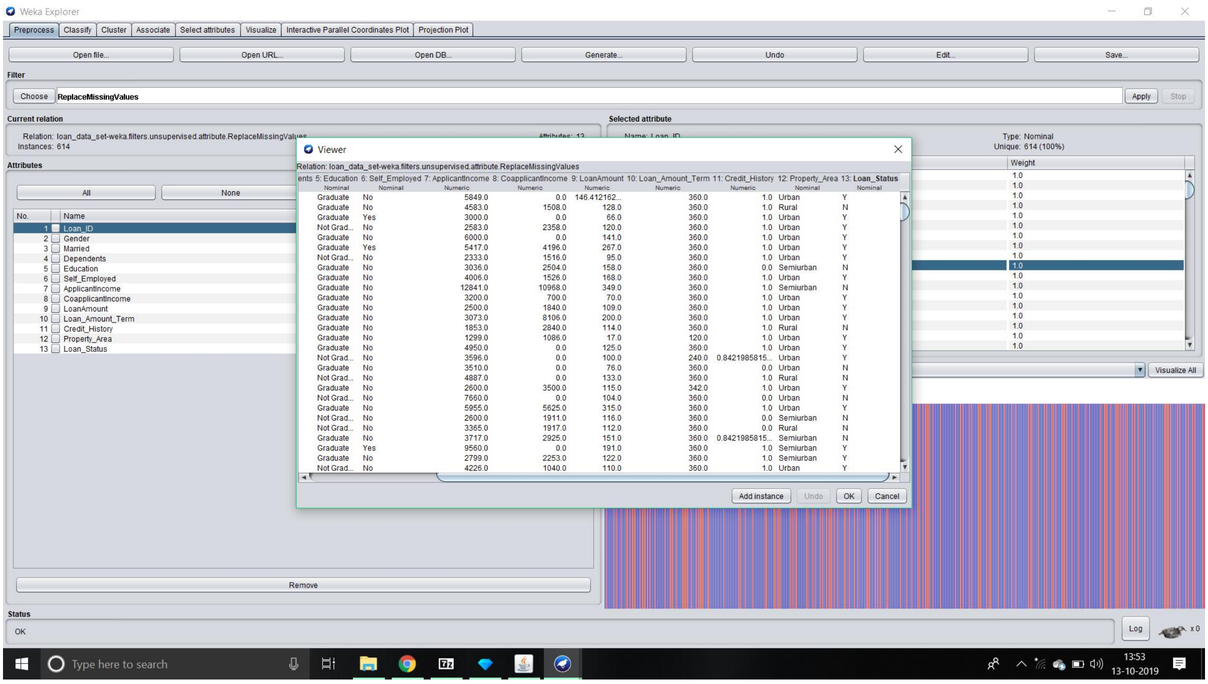


DATA PREPOCESSING:

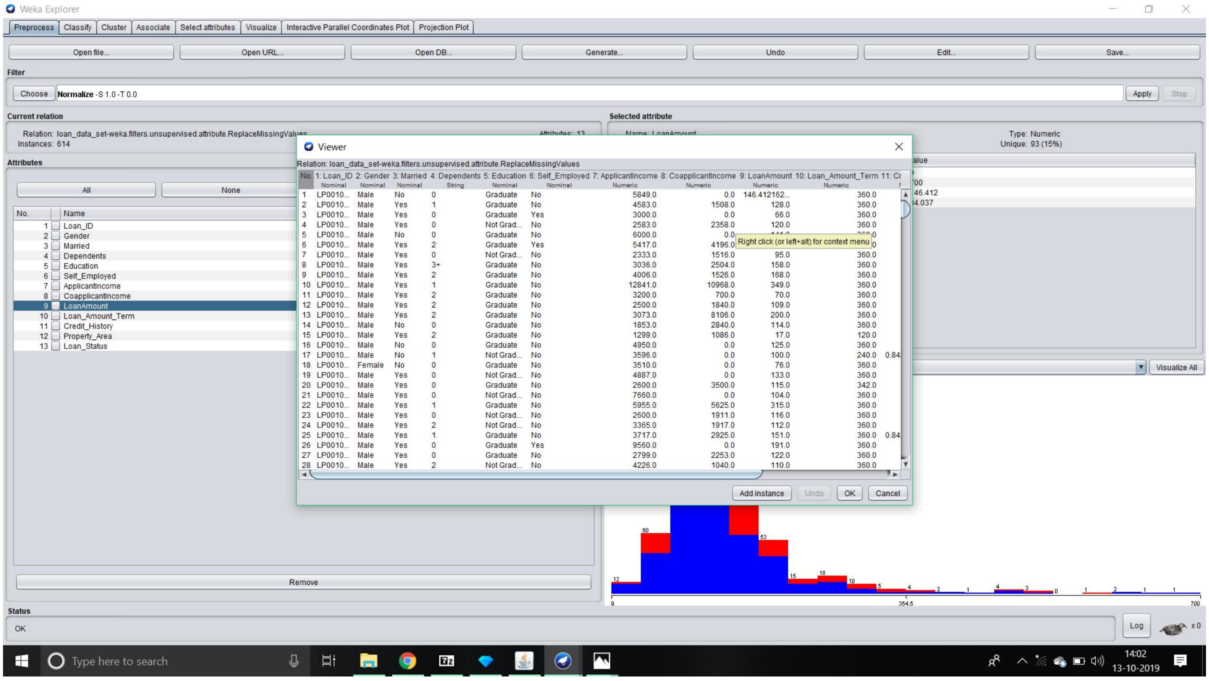
1. BEFORE REPLACE MISSING WITH MEAN



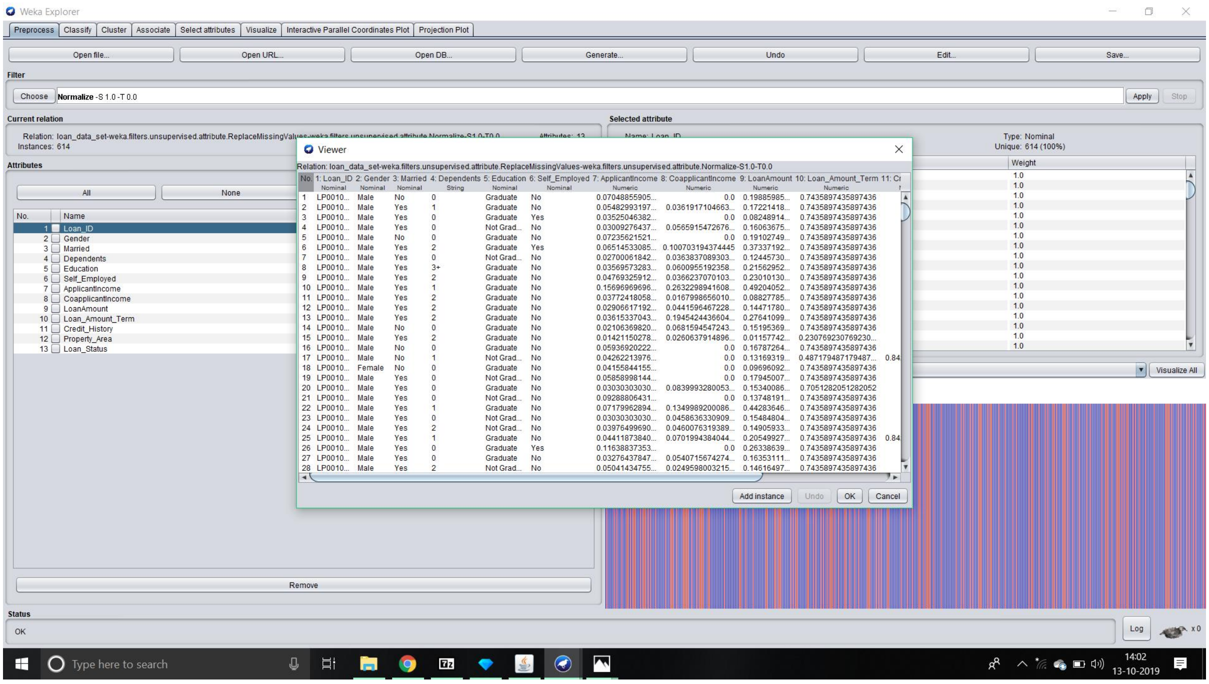
REPLACE MISSING WITH MEAN



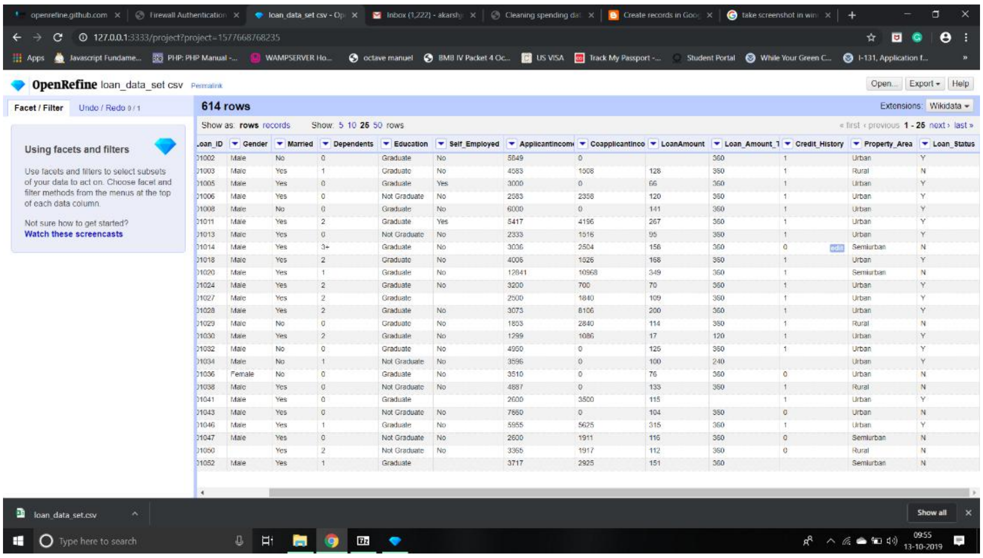
2. Before Normalizing



After Normalizing



3. Before preprocessing



Removing tuple with missing values(loan_amount)