https://www.kaggle.com/ajay1735/hmeq-data

Context: bad loans

The consumer credit department of a bank wants to automate the decisionmaking process for approval of home equity lines of credit. To do this, they will follow the recommendations of the Equal Credit Opportunity Act to create an empirically derived and statistically sound credit scoring model. The model will be based on data collected from recent applicants granted credit through the current process of loan underwriting. The model will be built from predictive modeling tools, but the created model must be sufficiently interpretable to provide a reason for any adverse actions (rejections).

Content

The Home Equity dataset (HMEQ) contains baseline and loan performance information for 5,960 recent home equity loans. The target (BAD) is a binary variable indicating whether an applicant eventually defaulted or was seriously delinquent. This adverse outcome occurred in 1,189 cases (20%). For each applicant, 12 input variables were recorded.

Inspiration

What if you can predict clients who default on their loans.

BAD: 1 = client defaulted on loan 0 = loan repaid

LOAN = Amount of the loan request

MORTDUE = Amount due on existing mortgage

VALUE = Value of current property

REASON

DebtCon = debt consolidation

HomeImp = home improvement

JOB = Six occupational categories

YOJ = Years at present job

DEROG = Number of major derogatory reports

DELINQ = Number of delinquent credit lines

CLAGE = Age of oldest trade line in months

NINQ = Number of recent credit lines

CLNO = Number of credit lines

DEBTINC = Debt-to-income ratio

Problem

Hypothesis regarding problem

Step. 1

1. Split data - join

Step . 2

Basic clean - what do we need to df / review for base line model? (adv is do to features of df)

1. Review imported data - df shape

Loaded df.shape: (5969, 14)

1. Review d.type of columns - numeric and nonnumeric? For machine learninig modelmodel

Numeric Column labels:

['ID' 'AMOUNT\_REQUESTED' 'EXIST\_MORTG\_DEBT' 'EXIST\_PROPERTY\_VALUE'

'EMPLOYED\_YEARS' 'DEROG\_REPORTS' 'DELINQ\_CR\_LINES' 'CR\_LINES\_AGE(MTS)'

'NO\_OF\_RECENT\_CR\_LINES' 'NO\_OF\_CR\_LINES' 'DEBT\_TO\_INCOME']

Non-Numeric Column labels:

['BAD\_LOAN' 'LOAN\_REASON' 'JOB']

Number of Columns: 14

Number of Numeric columns: 11

Number of Non-Numeric columns: 3

Number of columns = MATCH

1. EDA - descriptive (Shape/describe all)

DataFrame shape(): (5969, 14)

Dataframe describe(all): hard to read on this IDE

ID BAD\_LOAN ... NO\_OF\_CR\_LINES DEBT\_TO\_INCOME

count 5969.000000 5969 ... 5747.000000 4702.000000

unique NaN 2 ... NaN NaN

top NaN Repaid ... NaN NaN

freq NaN 4774 ... NaN NaN

mean 2976.035684 NaN ... 21.284670 33.782288

std 1723.184772 NaN ... 10.137312 8.595290

min 1.000000 NaN ... 0.000000 0.524499

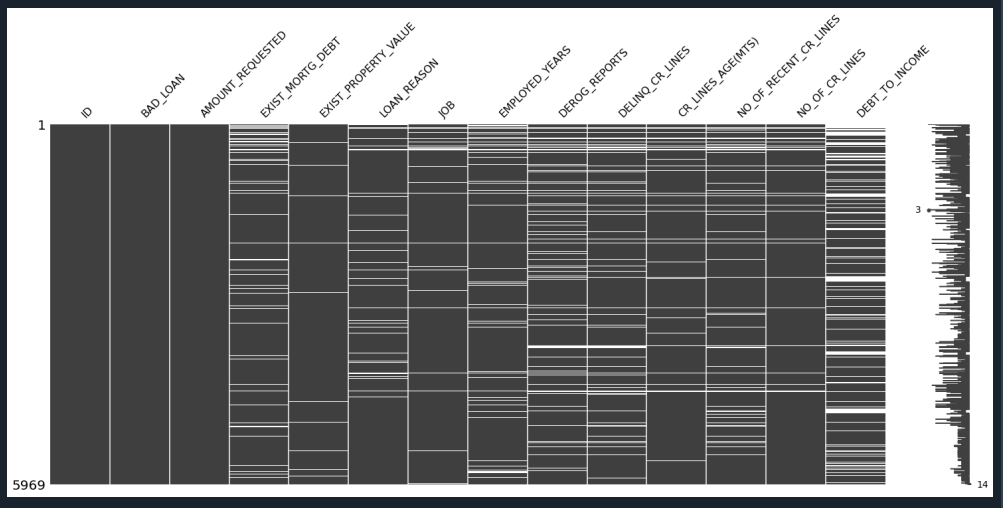
25% 1484.000000 NaN ... 14.000000 29.152041

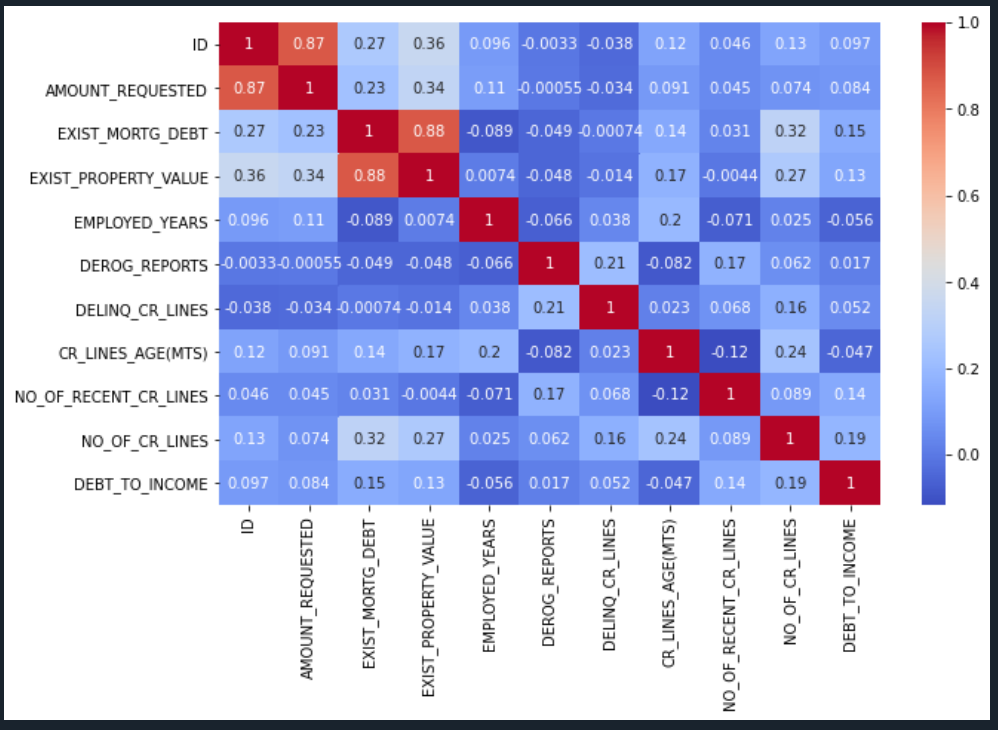
50% 2976.000000 NaN ... 20.000000 34.823546

75% 4468.000000 NaN ... 26.000000 38.999987

max 5960.000000 NaN ... 71.000000 203.312149

1. EDA - Visual (Missing data / heat map - correlation)





findings

1. Missing data

Total Percent

DEBT\_TO\_INCOME 1267 0.212

DEROG\_REPORTS 708 0.119

DELINQ\_CR\_LINES 580 0.097

EXIST\_MORTG\_DEBT 518 0.087

EMPLOYED\_YEARS 515 0.086

NO\_OF\_RECENT\_CR\_LINES 510 0.085

CR\_LINES\_AGE(MTS) 308 0.052

JOB 279 0.047

LOAN\_REASON 252 0.042

NO\_OF\_CR\_LINES 222 0.037

EXIST\_PROPERTY\_VALUE 112 0.019

AMOUNT\_REQUESTED 0 0.000

BAD\_LOAN 0 0.000

ID 0 0.000

If we have very high missing data we will consider for the baseline model deleting the column (> 40%), for rest of columns we will impute basic data for the baseline model the mean of the column or concider the unique data in the column after completing the unique data check.

1. Unique data

Examining feature content that exist not for missing values. We will deal with that separately in missing data

>> Column ID Unique contents for first 20:

[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

>> Column ID Unique contents & count:

27 4

6 4

26 4

665 1

653 1

..

5424 1

1330 1

3379 1

5428 1

2049 1

There should be no repeated ID numbers. We will drop duplicates

>> Column BAD\_LOAN Unique contents for first 20:

['Default' 'Repaid']

>> Column BAD\_LOAN Unique contents & count:

Repaid 4772

Default 1193

paid 2

Dfault 1

default 1

Expected TWO unique values but got FIVE. I will change to column data to Repaid, Default.

We can see 20% of loans appeared not to be repaid

>> Column AMOUNT\_REQUESTED Unique contents for first 20:

[1100 1300 1500 1700 1800 2000 2100 2200 2300 2400 2500 2800 2900 3000

3100 3200 3300 3400 3500 3600]

>> Column AMOUNT\_REQUESTED Unique contents & count:

15000 105

10000 81

20000 74

25000 73

12000 69

50700 1

58900 1

72300 1

47700 1

65500 1

Only appears as numeric data as expected

>> Column EXIST\_MORTG\_DEBT Unique contents for first 20:

[ 25860. 70053. 13500. nan 97800. 30548. 48649. 28502. 32700.

22608. 20627. 45000. 64536. 71000. 24280. 90957. 23030. 28192.

102370. 37626.]

>> Column EXIST\_MORTG\_DEBT Unique contents & count:

42000.0 11

47000.0 10

65000.0 9

45000.0 7

55000.0 7

..

96049.0 1

183615.0 1

52622.0 1

132417.0 1

4000.0 1

Only appears as numeric data as expected

>> Column EXIST\_PROPERTY\_VALUE Unique contents for first 20:

[ 39025. 68400. 16700. nan 112000. 40320. 57037. 43034. 46740.

62250. 29800. 55000. 87400. 83850. 34687. 102600. 40150. 120953.

46200. 73395.]

>> Column EXIST\_PROPERTY\_VALUE Unique contents & count:

60000.0 15

80000.0 14

85000.0 12

65000.0 11

78000.0 10

..

108483.0 1

65054.0 1

84519.0 1

64850.0 1

76207.0 1

Only appears as numeric data as expected

>> Column LOAN\_REASON Unique contents for first 20:

['HomeImp' nan 'DebtCon' 'Homeimp' 'debtCon' 'homeImp' 'debtcon']

>> Column LOAN\_REASON Unique contents & count:

DebtCon 3913

HomeImp 1787

debtcon 14

Homeimp 1

homeImp 1

debtCon 1

Expected TWO unique values but got SIX. will change to DebtCon, HomeImp

>> Column JOB Unique contents for first 20:

['Other' nan 'Office' 'Sales' 'Mgr' 'ProfExe' 'Self']

>> Column JOB Unique contents & count:

Other 2391

ProfExe 1276

Office 951

Mgr 770

Self 193

Sales 109

Expected SIX unique values but got SEVEN. Will investigate empty features and call Unknown

>> Column EMPLOYED\_YEARS Unique contents for first 20:

[10.5 7. 4. nan 3. 9. 5. 11. 16. 18. 2.5 8. 19. 4.5

2. 12. 22. 10. 26. 6. ]

>> Column EMPLOYED\_YEARS Unique contents & count:

0.00 415

1.00 363

2.00 347

5.00 333

4.00 327

4.60 1

17.80 1

0.25 1

5.60 1

6.60 1

Only appears as numeric data as expected

>> Column DEROG\_REPORTS Unique contents for first 20:

[ 0. nan 3. 2. 1. 4. 5. 6. 7. 8. 9. 10.]

>> Column DEROG\_REPORTS Unique contents & count:

0.0 4536

1.0 435

2.0 160

3.0 58

4.0 23

6.0 15

5.0 15

7.0 8

8.0 6

9.0 3

10.0 2

Only appears as numeric data as expected

>> Column DELINQ\_CR\_LINES Unique contents for first 20:

[ 0. 2. nan 1. 6. 15. 4. 3. 5. 7. 8. 10. 12. 11. 13.]

>> Column DELINQ\_CR\_LINES Unique contents & count:

0.0 4188

1.0 654

2.0 250

3.0 129

4.0 78

5.0 38

6.0 27

7.0 3

8.0 5

11.0 2

10.0 2

13.0 1

12.0 1

15.0 1

Only appears as numeric data as expected

>> Column CR\_LINES\_AGE(MTS) Unique contents for first 20:

[ 94.36666667 121.8333333 149.4666667 nan 93.33333333

101.4660019 77.1 88.76602988 216.9333333 115.8

122.5333333 86.06666667 147.1333333 123. 300.8666667

122.9 54.6 90.99253347 122.2666667 67.2 ]

>> Column CR\_LINES\_AGE(MTS) Unique contents & count:

206.966667 7

102.500000 7

123.766667 6

95.366667 6

109.566667 6

..

260.919776 1

237.366667 1

70.337531 1

233.848535 1

123.000000 1

Only appears as numeric data as expected

>> Column NO\_OF\_RECENT\_CR\_LINES Unique contents for first 20:

[ 1. 0. nan 2. 3. 5. 14. 10. 4. 9. 8. 6. 7. 11. 12. 17. 13.]

>> Column NO\_OF\_RECENT\_CR\_LINES Unique contents & count:

0.0 2534

1.0 1345

2.0 780

3.0 392

4.0 156

5.0 75

6.0 56

7.0 44

10.0 28

8.0 22

9.0 11

11.0 10

13.0 2

12.0 2

17.0 1

14.0 1

Only appears as numeric data as expected

>> Column NO\_OF\_CR\_LINES Unique contents for first 20:

[ 9. 14. 10. nan 8. 17. 12. 13. 25. 24. 16. 22. 0. 4. 21. 19. 45. 26.

37. 3.]

>> Column NO\_OF\_CR\_LINES Unique contents & count:

16.0 316

19.0 307

24.0 264

23.0 259

21.0 238

58.0 3

53.0 2

71.0 2

63.0 1

57.0 1

Only appears as numeric data as expected

>> Column DEBT\_TO\_INCOME Unique contents for first 20:

[ nan 37.11361356 36.88489409 3.7113123 31.58850318 38.26360073

29.68182705 30.05113629 29.91585903 36.211348 49.20639579 32.05978327

40.11567728 35.55353879 42.90999735 0.52449921 35.73055919 41.5163897

29.39354338 20.47091551]

>> Column DEBT\_TO\_INCOME Unique contents & count:

38.263601 4

29.681827 4

37.113614 4

21.621195 1

36.647081 1

..

22.169596 1

38.393635 1

27.969138 1

43.519093 1

27.520682 1

Only appears as numeric data as expected

1. Duplicate data

Now that we have an idea of the Missing and Unique data in the columns I will drop duplicated data. After visually reviewing the data I am going to keep the first of the rows. The unique content of the ID column suggests that a number of rows have been duplicated. Also, I am doing this step now as I do not want duplicated rows to affect the imputed data.

print("\nNumber of rows before drop\_duplicated: ",len(df))

df.drop\_duplicates(subset=None, keep='first', inplace=True, ignore\_index=False)

print("\nNumber of rows After drop\_duplicated: ", len(df));pause()

Review the data again to delete any columns we do not need and then we will tackle the unique data before basic impute of the missing data

1. Is there columns we do not need

Drop the ID column

DEBT\_TO\_INCOME Keep

DEROG\_REPORTS Keep

DELINQ\_CR\_LINES Keep

EXIST\_MORTG\_DEBT Keep

EMPLOYED\_YEARS Keep

NO\_OF\_RECENT\_CR\_LINES Keep

CR\_LINES\_AGE(MTS) Keep

JOB Keep

LOAN\_REASON Keep

NO\_OF\_CR\_LINES Keep

EXIST\_PROPERTY\_VALUE Keep

AMOUNT\_REQUESTED Keep

BAD\_LOAN Keep

ID Delete

df.drop('ID',axis=1, inplace=True)

1. Fix problem identified in Unique data

First we will check the uniquness of thes ecolumns

check\_columns = df[['BAD\_LOAN','LOAN\_REASON', ‘JOB’]]

dataframe\_unique\_check(check\_columns);pause()

Make the changes and check again. We can see that the columns are now correct.

>> Column BAD\_LOAN Unique contents & count:

Repaid 4771

Default 1189

>> Column LOAN\_REASON Unique contents & count:

DebtCon 3928

HomeImp 1780

>> Column JOB Unique contents & count:

Other 2667

ProfExe 1276

Office 948

Mgr 767

Self 193

Sales 109

1. Impute missing data

For this stpe I would review the missing data and as this was the baseline model I would impute the required data this is the out come

Total Percent

DEBT\_TO\_INCOME 1267 0.213

DEROG\_REPORTS 708 0.119

DELINQ\_CR\_LINES 580 0.097

EXIST\_MORTG\_DEBT 518 0.087

EMPLOYED\_YEARS 515 0.086

NO\_OF\_RECENT\_CR\_LINES 510 0.086

CR\_LINES\_AGE(MTS) 308 0.052

LOAN\_REASON 252 0.042

NO\_OF\_CR\_LINES 222 0.037

EXIST\_PROPERTY\_VALUE 112 0.019

JOB 0 0.000

AMOUNT\_REQUESTED 0 0.000

BAD\_LOAN 0 0.000

===> Press Return to Continue Program ?

Total Percent

DEBT\_TO\_INCOME 0 0.0

NO\_OF\_CR\_LINES 0 0.0

NO\_OF\_RECENT\_CR\_LINES 0 0.0

CR\_LINES\_AGE(MTS) 0 0.0

DELINQ\_CR\_LINES 0 0.0

DEROG\_REPORTS 0 0.0

EMPLOYED\_YEARS 0 0.0

JOB 0 0.0

LOAN\_REASON 0 0.0

EXIST\_PROPERTY\_VALUE 0 0.0

EXIST\_MORTG\_DEBT 0 0.0

AMOUNT\_REQUESTED 0 0.0

BAD\_LOAN 0 0.0

Using the followuing code

##Impute columns missing data

# DEBT\_TO\_INCOME 0.213

DEBT\_TO\_INCOME = round((df['DEBT\_TO\_INCOME'].mean()),1); df['DEBT\_TO\_INCOME'].fillna(DEBT\_TO\_INCOME, inplace=True)

# DEROG\_REPORTS 0.119

DEROG\_REPORTS = round((df['DEROG\_REPORTS'].mean()),1); df['DEROG\_REPORTS'].fillna(DEROG\_REPORTS, inplace=True)

# DELINQ\_CR\_LINES 0.097

DELINQ\_CR\_LINES = round((df['DELINQ\_CR\_LINES'].mean()),1); df['DELINQ\_CR\_LINES'].fillna(DELINQ\_CR\_LINES, inplace=True)

# EXIST\_MORTG\_DEBT 0.087

EXIST\_MORTG\_DEBT = round((df['EXIST\_MORTG\_DEBT'].mean()),1); df['EXIST\_MORTG\_DEBT'].fillna(EXIST\_MORTG\_DEBT, inplace=True)

# EMPLOYED\_YEARS 0.086

EMPLOYED\_YEARS = round((df['EMPLOYED\_YEARS'].mean()),1); df['EMPLOYED\_YEARS'].fillna(EMPLOYED\_YEARS, inplace=True)

# NO\_OF\_RECENT\_CR\_LINES 0.086

NO\_OF\_RECENT\_CR\_LINES = round((df['NO\_OF\_RECENT\_CR\_LINES'].mean()),1); df['NO\_OF\_RECENT\_CR\_LINES'].fillna(NO\_OF\_RECENT\_CR\_LINES, inplace=True)

# CR\_LINES\_AGE(MTS) 0.052

CR\_LINES\_AGE = round((df['CR\_LINES\_AGE(MTS)'].mean()),1); df['CR\_LINES\_AGE(MTS)'].fillna(CR\_LINES\_AGE, inplace=True)

# LOAN\_REASON 0.042

df['LOAN\_REASON'].fillna('Other', inplace = True)

# NO\_OF\_CR\_LINES 0.037

NO\_OF\_CR\_LINES = round((df['NO\_OF\_CR\_LINES'].mean()),1); df['NO\_OF\_CR\_LINES'].fillna(NO\_OF\_CR\_LINES, inplace=True)

# EXIST\_PROPERTY\_VALUE 112 0.019

EXIST\_PROPERTY\_VALUE = round((df['EXIST\_PROPERTY\_VALUE'].mean()),1); df['EXIST\_PROPERTY\_VALUE'].fillna(EXIST\_PROPERTY\_VALUE, inplace=True)

12 cretaed a test and train data frame

Step. 3

1. run various model on base line data to find baseline model
2. Will also run with and with out feature normalistaion to see what is better
3. The outcome of this step is to get baseline performance data fro a basline ML models

Sorted results for all models .head(20): (with Normalised data)

CV Model Model\_Accuracy Model\_STD Accuracy\_Score

148 12 ExtraTreesClassifier 0.948 0.015 1.000

114 10 ExtraTreesClassifier 0.944 0.013 1.000

80 8 ExtraTreesClassifier 0.942 0.011 1.000

131 11 ExtraTreesClassifier 0.942 0.015 1.000

97 9 ExtraTreesClassifier 0.941 0.011 1.000

46 6 ExtraTreesClassifier 0.940 0.015 1.000

63 7 ExtraTreesClassifier 0.937 0.012 1.000

29 5 ExtraTreesClassifier 0.937 0.011 1.000

12 4 ExtraTreesClassifier 0.934 0.011 1.000

45 6 RandomForestClassifier 0.919 0.010 1.000

28 5 RandomForestClassifier 0.918 0.009 1.000

147 12 RandomForestClassifier 0.918 0.013 1.000

79 8 RandomForestClassifier 0.918 0.011 1.000

62 7 RandomForestClassifier 0.917 0.008 1.000

130 11 RandomForestClassifier 0.917 0.012 1.000

113 10 RandomForestClassifier 0.916 0.012 1.000

96 9 RandomForestClassifier 0.914 0.011 1.000

11 4 RandomForestClassifier 0.913 0.008 1.000

31 5 GradientBoostingClassifier 0.901 0.009 0.919

65 7 GradientBoostingClassifier 0.901 0.011 0.919

Project Report

# GitHub URL

(insert URL here)

# Abstract

(Short overview of the entire project and features)

# Introduction

(Explain why you chose this project use case)

# Dataset

(Provide a description of your dataset and source. Also justify why you chose this source)

# Implementation Process

(Describe your entire process in detail)

# Results

(Include the charts and describe them)

# Insights

(Point out at least 5 insights in bullet points)

# References

(Include any references if required)