Edwisor Project 1

BANK LOAN DEFAULT CASE

By Dilip Kumar G

Content

1. **Introduction**

1.1 Problem Statement
1.2 Data set and Approach
1.3 EDA - Exploratory Data Analysis

2. **Data Preprocessing**

    2.1 Missing value analysis
    2.2 Outlier Analysis
    2.3 Feature Selection
    2.4 Feature Scaling

3. **Model development**

    3.1 Train Test Split
    3.2 Decision Tree
    3.3 Decision Tree Evaluation
    3.4 Random Forest
    3.5 Random Forest Evaluation
    3.6 Logistic Regression
    3.7 Logistic Regression Evaluation

4. **Conclusion**

5. **Coding**

Introduction

1.1 Problem Statement -

The objective of this case study is to classify new applicants into “Defaulter” or “Not-Defaulter” by using most accurate modeling.

1.2 Data Set and Approach

The dataset has 9 variables and 850 records, each record being loan default status for each customer. Each Applicant was rated as “Defaulted” or “Not-Defaulted”. According to the problem statement this is a binary classification problem.

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

Attribute information

Variable	Description	Variable Type
Age	Age of each customer	Numerical
Education	Education Categories	Categorical
Address	Geographic area	Numerical
Income	Gross income of each customer	Numerical
Debtinc	Individual's debt payment to his or her	Numerical
Creddebt	debt-to-credit ratio is a measurement of how much you owe your creditors as a percentage of your available credit (credit limits)	Numerical
Otherdebt	Any other debt	Numerical
Employment	Employment status	Numerical
Default	Whether the person has defaulted or not	categorical

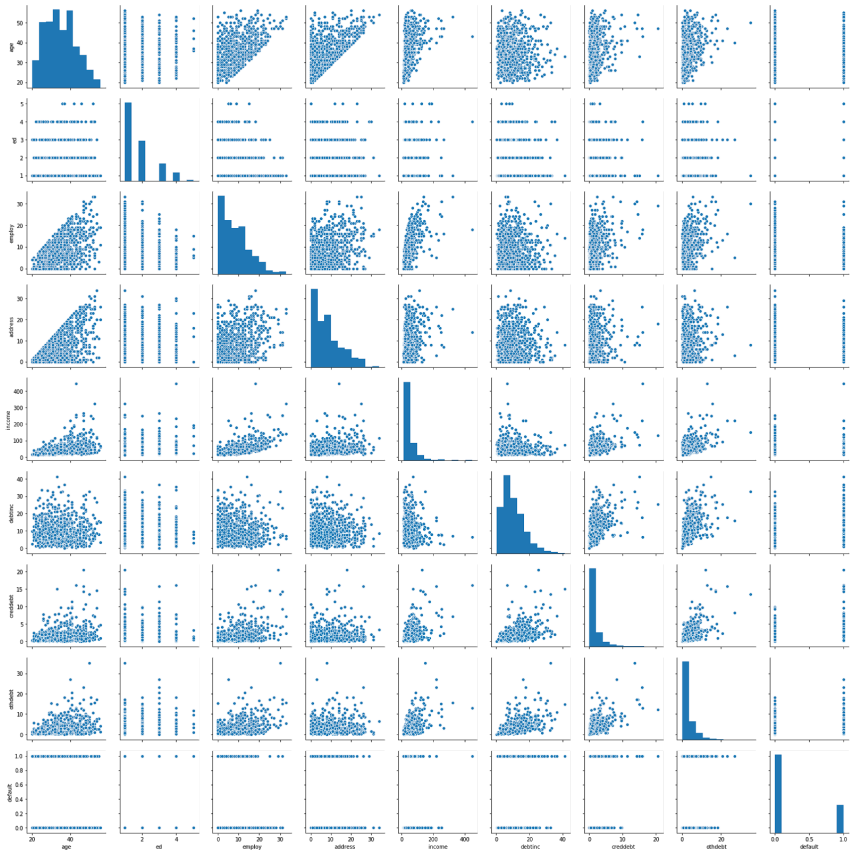
Approach

This is a classification model and we have to predict between the person who might become defaulted or not defaulted. For classification problems like this we can apply a number of machine learning algorithms such as Logistic regression, Random Forest, SVM, Decision tree etc.. Once these machine learning is applied we will evaluate these methods using different techniques and choose the model that gives the best accuracy rate.

1.3 Exploratory Data Analysis (EDA)

Any predictive modelling requires that we look at the data before we start modelling. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed

Let's create some simple plots to check out the data!



Checking the distribution of the data

```
df.describe()
```

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
count	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	700.000000
mean	35.029412	1.710588	8.565882	8.371765	46.675294	10.171647	1.576805	3.078789	0.261429
std	8.041432	0.927784	6.777884	6.895016	38.543054	6.719441	2.125840	3.398803	0.439727
min	20.000000	1.000000	0.000000	0.000000	13.000000	0.100000	0.011696	0.045584	0.000000
25%	29.000000	1.000000	3.000000	3.000000	24.000000	5.100000	0.382176	1.045942	0.000000
50%	34.000000	1.000000	7.000000	7.000000	35.000000	8.700000	0.885091	2.003243	0.000000
75%	41.000000	2.000000	13.000000	12.000000	55.750000	13.800000	1.898440	3.903001	1.000000
max	56.000000	5.000000	33.000000	34.000000	446.000000	41.300000	20.561310	35.197500	1.000000

If we see the above image we can see that variables such as “income”, “debtinc”, “creddebt”, “othdebt” has outliers. Will check and understand more about these outliers in coming steps.

2.1 Missing value analysis

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. If a column has more than 30% of data as missing value either we ignore the entire column or we ignore those observations. There are many ways to handle these missing values and it depends upon on variable type. Usually, if we find missing values in continuous variable we use Mean, median or KNN method to fill the values and in categorical variable we use following methods

1. Replace with most frequently repeated values
2. Apply classification model to predict the variable
3. Create Clustering
4. Ignore missing values

I checked for the missing values in the data and by looking at the below image we can see 150 missing values in the dependent variable. i.e Default variable. Missing values percentage is

17.3% and we can use these missing values to test the model after choosing the best model and once we get the prediction we can fill them in by default missing values.

```
df.isnull().sum()
```

```
age            0
ed             0
employ         0
address        0
income         0
debtinc        0
creddebt       0
othdebt        0
default      150
dtype: int64
```

Proposed Action

I created a new variable and stored missing values there. I will predict these missing values after getting the best model.

Separating 150 missing values and storing them in a new variable.

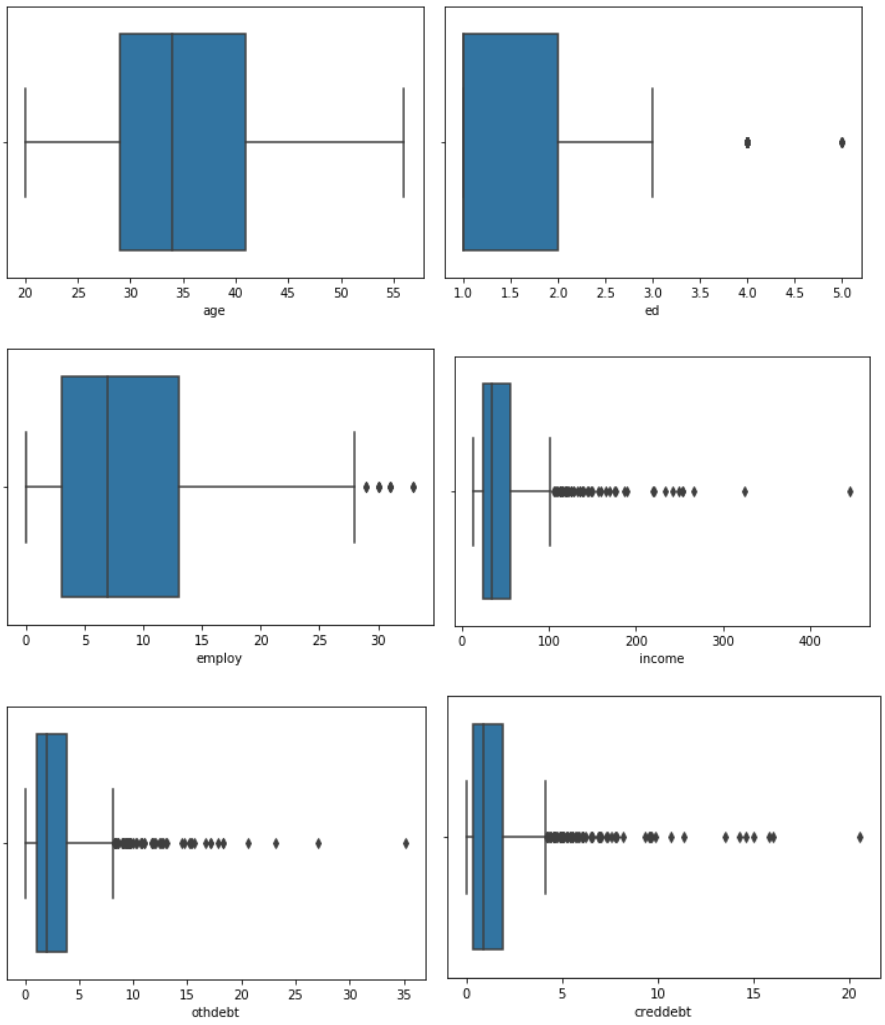
```
## separating missing values

dfmissingvalue = df[df.isnull().values.any(axis=1)]
```

```
dfmissingvalue.describe()
```

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
count	150.00000	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000	150.000000	0.0
mean	35.82000	1.653333	9.393333	8.806667	51.686667	9.756667	1.685314	3.174833	NaN
std	8.22508	0.926759	7.277710	7.221051	45.563310	6.196215	2.169602	3.886865	NaN
min	21.00000	1.000000	0.000000	0.000000	13.000000	0.100000	0.022050	0.052950	NaN
25%	29.25000	1.000000	4.000000	3.000000	25.250000	5.400000	0.533265	1.085211	NaN
50%	35.00000	1.000000	8.000000	7.000000	39.000000	8.900000	1.013213	2.111303	NaN
75%	43.00000	2.000000	13.750000	12.000000	61.750000	12.900000	1.898235	3.850151	NaN
max	56.00000	4.000000	33.000000	30.000000	324.000000	32.500000	13.552500	35.197500	NaN

2.2 Outlier Analysis

The other step of pre-processing is to check for the outliers. Outliers are values which are present in the dataset with an abnormal distance from most parts of values. These outliers can affect the quality of the modeling so it's very important to fix these outliers. The issue of outlier occurs only in continuous variables. Here to check the outlier in our dataset, we used a classic approach to visualize outliers, that is the Boxplot Method.



Proposed Action:

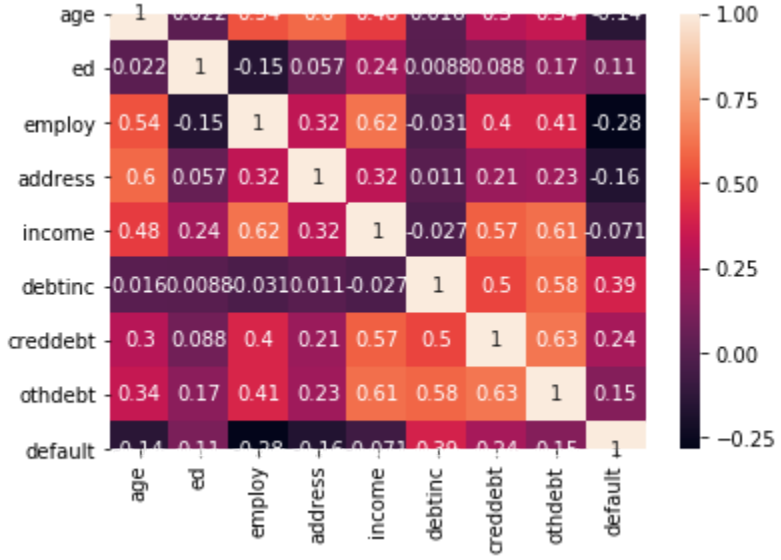
Though I see outliers in income, otherdebt and creddebts I have decided to keep these variables because we can make out that the income and debt values can't be outliers. Those

values will be either machine generated or based on the work experience. Also, we can understand that income and spending will be based on the individual age and education.

2.3 Feature Selection

Feature selection is a very important step that affects our model quality. In feature selection we analyse the importance of each predictor variable which will have a good impact on the outcome. There are various methods in feature selection like correlation matrices, chi square test, Anova method etc..

Here I will check for correlation matrix



Proposed Action

From the above image I did not see any high-correlation in any variable expect for slightly correlation between employ and income. I will continue with all the variables.

2.4 Feature Scaling

Feature scaling is a method to adjust the data that has different scales to avoid biases from big outliers. The most common techniques of feature scaling are Normalization and Standardization. Normalization is used when we want to bound our values between two numbers, typically, between [0,1] or [-1,1] And Standardization transforms the data so as to have zero mean and a variance of 1.

In the given data set we can see that in income we have high values whereas in othdebt we have low values so this could create an issue so I applied feature scaling to regression but only for logistic regression since decision tree and random forest won't require any feature scaling.

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

3 Model Development

After Data pre-processing the next step is to develop a model using a train or historical data Which can perform to predict accurate results on test data or new data. Here we have tried with different models and will choose the model which will provide the most accurate values.

3.1 Selecting Train and Test Data

We create a model to predict on the test data. So, we use the training data to fit the model and testing data to test it. The models generated are to predict the results unknown which is named as the test set. The dataset is divided into train and test set in order to check accuracies, precisions by training and testing it on it.

I am using the sklearn train test split method to separate the data set. In this, I have taken a 70-30 ratio. 70% of data will taken for training and rest 30% is for testing the model.

Steps

Train Test data splitting

```
X = train[['age', 'ed', 'employ', 'address', 'income', 'debtinc', 'creddebt',
       'othdebt']]
y = train['default']
```

```
##importing train_test library
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Now, the splitting of the data is complete and I will build a few models using these data and pick the most accurate model.

3.1 Decision Tree

The Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

Building Decision tree in python

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
dtree = DecisionTreeClassifier()
```

```
dtree.fit(X_train,y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, 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, presort=False,
                       random_state=None, splitter='best')
```

```
predictions = dtree.predict(X_test)
```

Evaluation

We are using confusion matrix to evaluate the models here.

What is confusion matrix?

Confusion matrix is a performance measurement for machine learning classification problems where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values. Below is the image that represents it.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	**TP**	**FN**
Class 2 Actual	**FP**	**TN**

Definition of terms

- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3.2 Decision Tree Evaluation

Decision tree evaluation

```
]: from sklearn.metrics import classification_report,confusion_matrix
```

```
]: print(classification_report(y_test,decision_predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.78      0.82       161
         1.0       0.44      0.55      0.49        49

    accuracy                           0.73       210
   macro avg       0.64      0.67      0.65       210
weighted avg       0.75      0.73      0.74       210
```

```
]: print(confusion_matrix(y_test,decision_predictions))
```

```
[[126  35]
 [ 22  27]]
```

```
]: accuracy_score(y_test, decision_predictions)
```

```
]: 0.7285714285714285
```

In the decision tree I have got a 72% accuracy rate. It's not that bad but still we will have to check other models.

3.3 Random Forest

Random forest is a tree-based algorithm which involves building several trees (decision trees), combining their output to improve the ability of the model to generalize. Random forest avoids overfitting problem.

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, 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=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
random = rfc.predict(X_test)
```

Random Forest Evaluation

```
]: print(confusion_matrix(y_test,random))
```

```
[[150  11]
 [ 27  22]]
```

```
]: print(classification_report(y_test,random))
```

```
              precision    recall  f1-score   support

         0.0       0.85      0.93      0.89       161
         1.0       0.67      0.45      0.54        49

    accuracy                           0.82       210
   macro avg       0.76      0.69      0.71       210
weighted avg       0.81      0.82      0.81       210
```

```
]: accuracy_score(y_test, random)
```

```
]: 0.819047619047619
```

In Random forest, I have got an 81% accuracy rate. This is better than decision tree but Random forest is a very slow.

3.5 Logistic regression

The logistic regression model is performed with some assumptions that data has no outliers and there are two classes to be predicted, and that no two independent variables are highly correlated to each other.

Building and Testing Logistic regression in python using sklearn method.

```
from sklearn.linear_model import LogisticRegression
```

```
logmodel = LogisticRegression()
logmodel.fit(X_train,y_train)
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.py:432: FutureWarning
e changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
predictions = logmodel.predict(X_test)
```

3.6 Logistic Regression Evaluation

For the models to be successful it has to have certain accuracy levels. We can evaluate the model using a confusion matrix, classification report or just by checking the accuracy levels. I have evaluated using all three methods here.

```
from sklearn.metrics import classification_report
```

```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

         0.0       0.89      0.93      0.91       161
         1.0       0.74      0.63      0.68        49

    accuracy                           0.86       210
   macro avg       0.82      0.78      0.80       210
weighted avg       0.86      0.86      0.86       210
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(y_test, predictions)
```

```
0.861904761904762
```

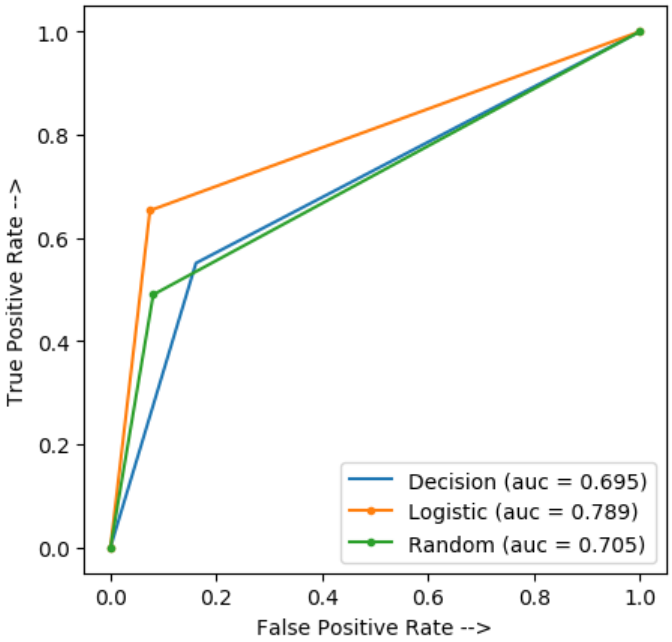
```
from sklearn.metrics import confusion_matrix
```

```
confusion_matrix(y_test, predictions)
```

```
array([[150,  11],
       [ 18,  31]])
```

In the logistic model, I have got an accuracy score of 86%. That's a good accuracy score

ROC Curve:



4. Conclusion

The decision tree area under the score is 0.69 and random forest area under score is 0.70 and logistic regression area under score is better than both decision tree and random forest.

Conclusion:

Logistic regression is performing better for this particular problem.

5. Coding

Python

```python
#!/usr/bin/env python
# coding: utf-8

# ## Import Libraries

# In[1]:


import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


os.getcwd()


# In[3]:


os.chdir("/Users/dilip/desktop/python/")


# ## Data Upload

# In[4]:
```

```
32
33
34 df = pd.read_csv("bankloan.csv")
35
36
37 # ## EDA (Exploratory Data Analysis)
38
39 # In[5]:
40
41
42 df.head()
43
44
45 # In[6]:
46
47
48 df.info()
49
50
51 # In[7]:
52
53
54 df.describe()
55
56
57 # In[8]:
58
59
60 df.shape
61
62
63 # In[9]:
64
65
66 sns.pairplot(df)
67
```

```
68
69 # ## Missing Value Analysis
70
71 # In[10]:
72
73
74 df.isnull().sum()
75
76
77 # We can see 150 missing values in the dependent variable
   i.e default. We can separate this data and use it for
   testing the model.
78
79 # In[11]:
80
81
82 ## separating missing values
83
84 dfmissingvalue = df[df.isnull().values.any(axis=1)]
85
86
87 # In[12]:
88
89
90 dfmissingvalue.describe()
91
92
93 # In[13]:
94
95
96 #dropping all the missing values
97 df.dropna()
98
99
100 # In[14]:
101
```

```
102
103 #creating new variable without missing values
104 newdata = df.dropna()
105
106
107 # In[15]:
108
109
110 newdata.describe()
111
112
113 # ## Data understanding
114
115 # In[16]:
116
117
118 #checking how default and non-default ratio
119 sns.countplot(x="default", data=df)
120
121
122 # In[17]:
123
124
125 #default with respect to employment status
126 sns.countplot(x="employ" , hue="default", data=df)
127
128
129 # employment status 0-30 has more number of defaut cases.
    I will check whether this group is from lower income
130
131 # In[18]:
132
133
134 ##default with respect to income
135 sns.countplot(x="income" , hue="default", data=df)
136
```

```
137
138 # ## Outlier Analysis
139
140 # In[19]:
141
142
143 sns.boxplot(df["age"])
144
145
146 # In[20]:
147
148
149 sns.boxplot(df["ed"])
150
151
152 # In[21]:
153
154
155 sns.boxplot(df["employ"])
156
157
158 # In[22]:
159
160
161 sns.boxplot(df["income"])
162
163
164 # In[23]:
165
166
167 sns.boxplot(df["othdebt"])
168
169
170 # In[24]:
171
172
```

```
sns.boxplot(df["creddebt"])


# I can see more number of outliers in "income", "creddebt", and "othdebt" and it's unlikely that these data are outliers so I have decided to keep all the variables for analysis.

# ## Feature Selection

# In[25]:


##ploting correlation matrix
sns.heatmap(newdata.corr(),annot=True)


# There's no high correlation observed in the above matrix.

# ## Train Test data splitting

# In[26]:


X = newdata[['age', 'ed', 'employ', 'address', 'income', 'debtinc', 'creddebt',
       'othdebt']]
y = newdata['default']


# In[27]:


##importing train_test library
from sklearn.model_selection import train_test_split
```

```
204
205
206 # In[58]:
207
208
209 X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3, random_state=42)
210
211
212 # ## Decision Tree
213
214 # In[59]:
215
216
217 from sklearn.tree import DecisionTreeClassifier
218
219
220 # In[ ]:
221
222
223 dtree = DecisionTreeClassifier()
224
225
226 # In[60]:
227
228
229 dtree.fit(X_train,y_train)
230
231
232 # In[61]:
233
234
235 decision_predictions = dtree.predict(X_test)
236
237
238 # ## Decision tree evaluation
```

```
239
240 # In[62]:
241
242
243 from sklearn.metrics import
    classification_report,confusion_matrix
244
245
246 # In[63]:
247
248
249 print(classification_report(y_test,decision_predictions))
250
251
252 # In[64]:
253
254
255 print(confusion_matrix(y_test,decision_predictions))
256
257
258 # In[65]:
259
260
261 accuracy_score(y_test, decision_predictions)
262
263
264 # ## Random Forest
265
266 # In[66]:
267
268
269 from sklearn.ensemble import RandomForestClassifier
270 rfc = RandomForestClassifier(n_estimators=100)
271 rfc.fit(X_train, y_train)
272
273
```

```
# In[67]:


random = rfc.predict(X_test)


# ## Random Forest Evaluation

# In[68]:


print(confusion_matrix(y_test,random))


# In[69]:


print(classification_report(y_test,random))


# In[72]:


accuracy_score(y_test, random)


# ## Feature Scaling

# In[29]:


#importing the libraries
from sklearn.preprocessing import StandardScaler


# In[30]:
```

```
310
311
312 stndrd=StandardScaler()
313 X_train=stndrd.fit_transform(X_train)
314
315
316 # In[31]:
317
318
319 X_test=stndrd.fit_transform(X_test)
320
321
322 # In[32]:
323
324
325 X_train
326
327
328 # In[33]:
329
330
331 X_test
332
333
334 # ## Model Development
335
336 # ## Logistic regression
337
338 # In[34]:
339
340
341 from sklearn.linear_model import LogisticRegression
342
343
344 # In[35]:
345
```

```
346
347 logmodel = LogisticRegression()
348 logmodel.fit(X_train,y_train)
349
350
351 # In[36]:
352
353
354 predictions = logmodel.predict(X_test)
355
356
357 # In[70]:
358
359
360 accuracy_score(y_test, predictions)
361
362
363 # ## Logistic regression Evaluation
364
365 # In[37]:
366
367
368 from sklearn.metrics import classification_report
369
370
371 # In[38]:
372
373
374 print(classification_report(y_test,predictions))
375
376
377 # In[39]:
378
379
380 from sklearn.metrics import accuracy_score
381
```

```
382
383 # In[40]:
384
385
386 accuracy_score(y_test, predictions)
387
388
389 # In[41]:
390
391
392 from sklearn.metrics import confusion_matrix
393
394
395 # In[42]:
396
397
398 confusion_matrix(y_test, predictions)
399
400
401 # In[57]:
402
403
404 from sklearn.metrics import roc_curve, auc
405
406 logistic_fpr, logistic_tpr, threshold = roc_curve(y_test,
    predictions)
407 auc_logistic = auc(logistic_fpr, logistic_tpr)
408
409 decision_fpr, decision_tpr, threshold = roc_curve(y_test,
    decision_predictions)
410 auc_decision = auc(decision_fpr, decision_tpr)
411
412 random_fpr, random_tpr, threshold = roc_curve(y_test,
    random)
413 auc_random = auc(random_fpr, random_tpr)
414
```

```
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(decision_fpr, decision_tpr, linestyle='-',
  label='Decision (auc = %0.3f)' % auc_decision)
plt.plot(logistic_fpr, logistic_tpr, marker='.',
  label='Logistic (auc = %0.3f)' % auc_logistic)
plt.plot(random_fpr, random_tpr, marker='.', label='Random
  (auc = %0.3f)' % auc_random)


plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()


# Compared Deccision Treee and Random forest accuracy
  level in logistic regression is suitable here since the
  accuracy level in logistic regression is high

```

R

```
rm(list=ls(all=T))


# check the current working directory
getwd()

# setting the working directory
```

```r
setwd("/Users/dilip/desktop/python")


# loading important required libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees','fastDummies', 'psych')

#install.packages(x)
lapply(x, require, character.only = TRUE)

library("dplyr")
library("plyr")
library("ggplot2")
library("data.table")
library("GGally")

#load the data
bank = read.csv("bankloan.csv", header = T, na.strings = c(" ","","NA"))
# Summarizing  data
# dataset contains 850 obs. of 9 variables
dim(bank)

### this will help us to see status
table(bank$default)


# brining first 5 rows of the datasets
head(bank, 5)

tail(bank, 5)

# getting the column names of the dataset.
colnames(bank)

```

```r
# str of the data
str(bank)

bank$default = as.factor(bank$default)

str(bank)

describe(bank)

############## distribution of the age ###
#### age is normally distributed.
plot(density(bank$age))
hist(bank$age, main = " age histogram" , xlab = 'age', ylab = "freq")

#####

ggplot(bank) +
  geom_bar(aes(x=ed),fill="grey")

################ missing value analysis ###################

sum(is.na(bank$default))
missing_val = data.frame(apply(bank, 2, function(x){sum(is.na(x))}))

missing_val$columns = row.names(missing_val)

row.names(missing_val) = NULL

names(missing_val)[1] = "missing_percentage"

missing_val$missing_percentage =
(missing_val$missing_percentage/nrow(bank))*100.

missing_val = missing_val[order(-missing_val$missing_percentage),]

missing_val = missing_val[,c(2,1)]
```

```r

###################### outlier chech ####################
##### All the given variables has outliers. I assume that the income and related
# debt values are dependent on the observer. more the education more the values
### might be.

boxplot(bank$age, main = " outlier check @ age", ylab = " age", col = 5)

boxplot(bank$income, main = " outlier check @ income", ylab = "income", col = 5)

boxplot(bank$creddebt, main = " outlier check @ creddebt", ylab = "creddebt", col = 5)

boxplot(bank$othdebt, main = " outlier check @ othdebt", ylab = "othdebt", col = 5)

### or
ggplot(data = bank, aes(x = "", y = age)) +
 geom_boxplot()

ggplot(data = bank, aes(x = "", y = income)) +
 geom_boxplot()

ggplot(data = bank, aes(x = "", y = creddebt)) +
 geom_boxplot()

 ggplot(data = bank, aes(x = "", y = othdebt)) +
  geom_boxplot()


## Correlation Plot
numeric_index = sapply(bank, is.numeric)
corrgram(bank[,numeric_index], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

###### standardisation###########
```

```
111
112 cnames = colnames(bank)
113 for (i in cnames) {
114   print(i)
115   bank[,i] = (bank[,i]-mean(bank[,i]))/sd(bank[,i])
116 }
117
118 str(bank)
119 ######################## separating the labeled and not labeled
    observations.
120 ##### last 150 observations are not labled. Those observations to be
121 ### predicted after the model building. we will predict those values after
122 ### chossing best model.
123
124 bank_train = bank[1:700,1:9]
125 dim(bank_train)
126 sum(is.na(bank$default))
127 bank_test  = bank[701:850,1:8]
128 dim(bank_test)
129
130 ############# Model buildig##############
131 library(caTools)
132
133 #Splitting into training and testing data
134 set.seed(123)
135 sample = sample.split(bank_train, SplitRatio = 0.8)
136 sample
137 training = subset(bank_train, sample==TRUE)
138 str(training)
139 testing = subset(bank_train, sample==FALSE)
140 str(testing)
141
142
143 ######################logistic regression
    ########################
144
```

```
145 model = glm(default~.,training, family = "binomial")
146
147 summary(model)
148
149 model1 = glm(default~age,training, family = "binomial")
150
151 summary(model1)
152
153 model2 = glm(default~ed,training, family = "binomial")
154
155 summary(model2)
156
157 model3 = glm(default~employ,training, family = "binomial")
158
159 summary(model3)
160
161 model4 = glm(default~address,training, family = "binomial")
162
163 summary(model4)
164
165 model5 = glm(default~creddebt,training, family = "binomial")
166
167 summary(model5)
168
169 ##########################model with high important
    variables############
170
171 model6 = glm(default~creddebt+debtinc+address+employ,training, family =
    "binomial")
172
173 summary(model6)
174
175 res = predict(model6, testing, type = "response")
176
177 range(res)
178
```

```r
confusion_matric = table(Actualvalue=testing$default, predictedvalue=res>0.5)

print(confusion_matric)

accuracy = (104+20)/(104+20+24+7)

print(accuracy)

# precision = 0.74
# recall = 0.45

####### threshold evaluation ################
### ROC CURVE ##########
######AUC####
library(ROCR)
pred_log = prediction(res,testing$default)
acc = performance(pred_log,"acc")
plot(acc)

roc_curve = performance(pred_log, "tpr" , "fpr")
plot(roc_curve)
plot(roc_curve , colorize = T, print.cutoffs.at=seq(0.1,by=0.1))

###### using threshold value of 0.4 we can incraese the true positive rate

confusion_matric = table(Actualvalue=testing$default, predictedvalue=res>0.4)

print(confusion_matric)

accuracy = (107+18)/(107+18+4+26)

print(accuracy)

# accuracy = 0.80
# precision = 0.58
# recall =  0.56
```

```
215
216 ############ AUC##############
217
218 auc = performance(pred_log, "auc")
219 auc
220
221 # AUC = 0.82
222
223 #############Precision recall curve ##############
224 library(PRROC)
225 PRC_curve = performance(pred_log, "prec" , "rec")
226 plot(PRC_curve, colorize = T)
227
228 ############################## DEcision tree###############
229 library(tree)
230 deci_model = tree(default~., data = training)
231 summary(deci_model)
232
233 ### plotting
234 plot(deci_model)
235 text(deci_model,pretty = 0)
236
237 #### prediction
238
239 deci_pred = predict(deci_model, testing, type = "class")
240
241 confusion_matric = table(Actualvalue=testing$default,
    predictedvalue=deci_pred)
242
243 print(confusion_matric)
244
245 #### cross validation
246 cv.deci_model = cv.tree(deci_model, FUN = prune.misclass)
247 cv.deci_model
248 plot(cv.deci_model)
249
```

```r
####pruning
prune.deci_model = prune.misclass(deci_model, best = 10)
plot(prune.deci_model)
text(prune.deci_model)

#### prediction of values again
deci_pred_1 = predict(prune.deci_model, testing, type = "class")

Confusion_matrix_1= table(testing$default, deci_pred_1)

print(Confusion_matrix_1)
# accuracy = 0.74 # precision = 0.54 # recall = 0.43

##################################Random Forest#################

#####random forest 1

library(randomForest)
rf = randomForest(default~., data = training)
print(rf)

## prediction

rf_pred = predict(rf,testing)

confusion_matric = table(Actualvalue=testing$default, predictedvalue=rf_pred)

print(confusion_matric)

## tune mtry

tuneRF(training[,-9], training[,9],stepfactor = 0.5,
       plot = TRUE , ntreeTry = 1000,
       trace = TRUE ,
       improve = 0.05)
```

```
285
286 rf1 = randomForest(default~.,data = training, ntree = 1000, mtry = 2)
287
288 rf1
289
290 # predict
291
292 rf_pred1 = predict(rf1,testing)
293 confusion_matric1 = table(Actualvalue=testing$default,
    predictedvalue=rf_pred1)
294
295 print(confusion_matric1)
296
297 # no. of nodes for the trees
298 hist(treesize(rf1),main = " no. of nodes for the trees", col = "green")
299
300 # variable importance
301 varImpPlot(rf1,
302          sort = T,
303          main = "variable importance")
304
305 importance(rf1)
306 varUsed(rf1)
307
308 ###############  building  random forest by taking only max
    meandecreaseGini
309 ### considering debtinc, employ, creddebt, othdeb, income.
310 ### build model
311 rf_final = randomForest(default~debtinc+employ+creddebt+othdebt+income ,
312                         data = training,
313                         ntree = 1000, mtry = 2)
314 rf_final
315
316 # prediction
317 rf_pred_final = predict(rf_final,testing)
318
319 confusion_matric_f = table(Actualvalue=testing$default,
```

```
    predictedvalue=rf_pred_final)
320
321 print(confusion_matric_f)
322
323 # accuracy = 0.735
324 # precision = 0.55
325 # recall = 0.340
326
327
328 ##### Conclusion ======= logistic model is the best suited model on this
    dataset.
329
330 #### predicting for the test data.
331 res = predict(model6, bank_test, type = "response")
332 range(res)
```