Credit Approval Model

Raj Solanki

5/15/2019

Intro

This project will go over the application of various machine learning models to predict whether or not an applicant should be approved for a credit card or not. In today's modern economy, cash is no longer dominating the payment space as it used to. More consumers are turning towards credit cards as a universal form of payment. Credit cards offer many more benefits to the consumer such as

1. Quick and Easy
2. Ability to transactions online and in real time
3. Hassle free especially with carrying around change etc.
4. Basically a line of credit, allow consumers to have flexibility in their payments.

With these massive changes coming into the payment space, more consumers are applying for credit cards. This means that this places immense pressure on credit card companies to look into automated strategies for approval to make decisions within a split second. Later in this project, we will be going over how **K-NN** and **Decision Trees** could be applied to this real life problem.

1. **Business Problem:** We want to be able to accurately predict whether or not an applicant should be approved or not.
2. **Data Preparation** The data has already been gathered and put together in a table format. Here we will load into R and take a look some general attributes and exploratory work related to the data set before we start building predictive models.

```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
data<-read.csv("C:/Users/Raji Roy/Desktop/Udemy/Credit Approval Model/crx.txt",header = F,sep=",", stri
```

```
paste0("The shape of the data set is (",nrow(data),",",ncol(data),")"  )
```

```
## [1] "The shape of the data set is (690,16)"
```

The columns are anonymized since this is actual applicant information. Here we rename the columns of the data set.

```
rename<-paste0("V",1:15)
rename[16]<-"Target"
names(data)<-rename
names(data)
```

```
##  [1] "V1"     "V2"     "V3"     "V4"     "V5"     "V6"     "V7"
##  [8] "V8"     "V9"     "V10"    "V11"    "V12"    "V13"    "V14"
## [15] "V15"    "Target"
```

```
summary(data)
```

```
##       V1                  V2                    V3               V4
##  Length:690          Length:690          Min.   : 0.000   Length:690
##  Class :character   Class :character   1st Qu.: 1.000   Class :character
##  Mode  :character   Mode  :character   Median : 2.750   Mode  :character
##                                         Mean   : 4.759
##                                         3rd Qu.: 7.207
##                                         Max.   :28.000
##       V5                 V6                 V7                  V8
##  Length:690          Length:690          Length:690          Min.   : 0.000
##  Class :character   Class :character   Class :character   1st Qu.: 0.165
##  Mode  :character   Mode  :character   Mode  :character   Median : 1.000
##                                                             Mean   : 2.223
##                                                             3rd Qu.: 2.625
##                                                             Max.   :28.500
##       V9                V10                V11           V12
##  Length:690          Length:690          Min.   : 0.0   Length:690
##  Class :character   Class :character   1st Qu.: 0.0   Class :character
##  Mode  :character   Mode  :character   Median : 0.0   Mode  :character
##                                         Mean   : 2.4
##                                         3rd Qu.: 3.0
##                                         Max.   :67.0
##      V13                V14                V15
##  Length:690          Length:690          Min.   :     0.0
##  Class :character   Class :character   1st Qu.:     0.0
##  Mode  :character   Mode  :character   Median :     5.0
##                                         Mean   :  1017.4
##                                         3rd Qu.:   395.5
##                                         Max.   :100000.0
##     Target
##  Length:690
##  Class :character
##  Mode  :character
##
##
##
```

Now we will convert the data to the appropriate data types and re code the target variable to 0 for not approved and 1 for approved. We then use the `as.numeric` function to convert the characters to the numerical factors.

```
data$V1<-as.factor(data$V1)
data$V1<-as.numeric(data$V1)

data$V4<-as.factor(data$V4)
data$V4<-as.numeric(data$V4)

data$V5<-as.factor(data$V5)
data$V5<-as.numeric(data$V5)

data$V6<-as.factor(data$V6)
data$V6<-as.numeric(data$V6)

data$V7<-as.factor(data$V7)
data$V7<-as.numeric(data$V7)

data$V9<-as.factor(data$V9)
data$V9<-as.numeric(data$V9)
```

```r
data$V10<-as.factor(data$V10)
data$V10<-as.numeric(data$V10)

data$V12<-as.factor(data$V12)
data$V12<-as.numeric(data$V12)

data$V13<-as.factor(data$V13)
data$V13<-as.numeric(data$V13)

data$Target<-ifelse(data$Target=="+",1,0)
```

This data set does have missing values which were imputed beforehand. We will check for column wise just to be on the safe side. Missing values are coded as ?. We have missing values in columns V2 and V14 which are both continuous, and we will impute these with the mean.

```r
  for(i in names(data))

  {

    print(paste0("There are ",sum(data[,i]=="?")," missing values in column ",i))

  }
```

```
## [1] "There are 0 missing values in column V1"
## [1] "There are 12 missing values in column V2"
## [1] "There are 0 missing values in column V3"
## [1] "There are 0 missing values in column V4"
## [1] "There are 0 missing values in column V5"
## [1] "There are 0 missing values in column V6"
## [1] "There are 0 missing values in column V7"
## [1] "There are 0 missing values in column V8"
## [1] "There are 0 missing values in column V9"
## [1] "There are 0 missing values in column V10"
## [1] "There are 0 missing values in column V11"
## [1] "There are 0 missing values in column V12"
## [1] "There are 0 missing values in column V13"
## [1] "There are 13 missing values in column V14"
## [1] "There are 0 missing values in column V15"
## [1] "There are 0 missing values in column Target"
```

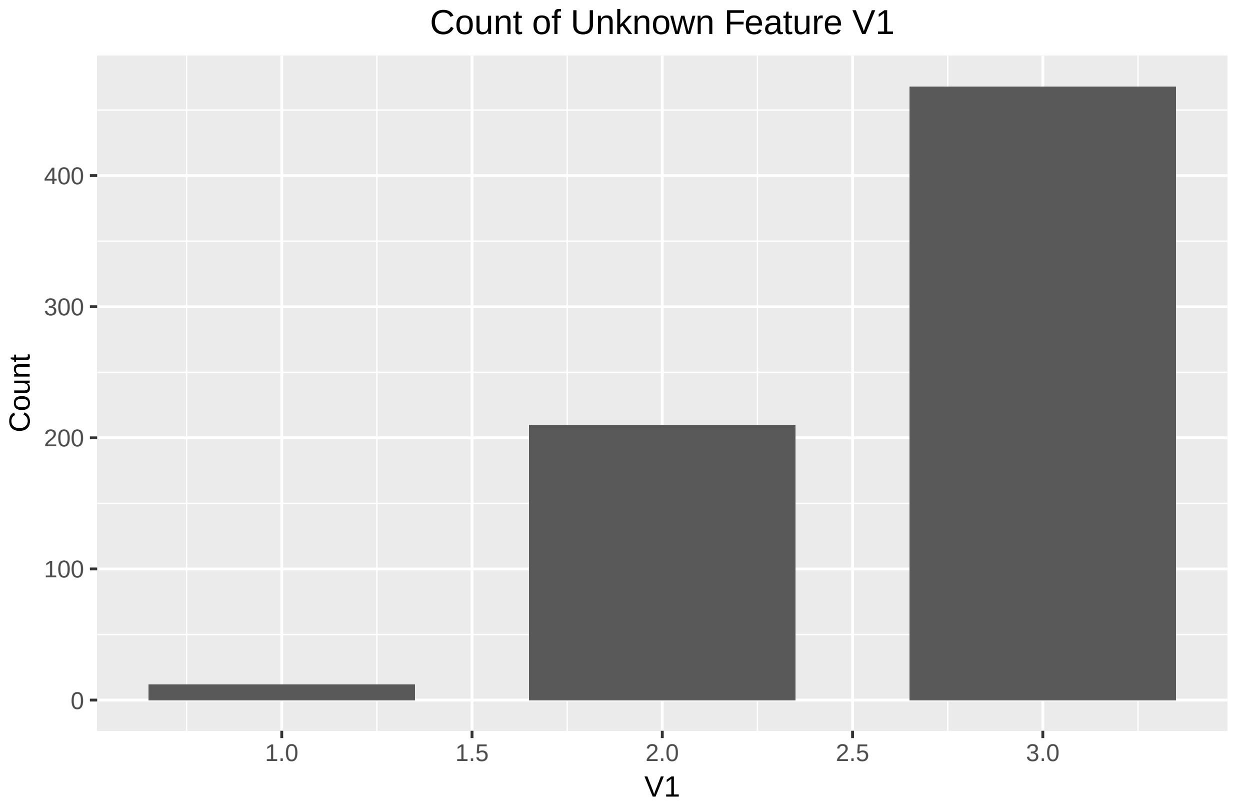
```r
data$V2[data$V2=="?"]<-mean( as.numeric(data$V2[data$V2!="?"]))
data$V2<-as.numeric(data$V2)

data$V14[data$V14=="?"]<-mean( as.numeric(data$V14[data$V14!="?"]))
data$V14<-as.numeric(data$V14)
```

3. **Exploratory Data Analysis (EDA)** - Here we are going to take a look at several plots to see if we can identify any patterns in the data. `ggplot2` will be used

Plot 1 shows the distribution of values in V1 and we see that the majority of the values are b. There doesn't appear to be a large skew of values in 1 category.

```r
p<-ggplot(data=data, aes(x=V1)) +

  geom_bar(stat="count",width=0.7)+labs(caption="Plot 1")
```

```
p<-p+ggtitle("Count of Unknown Feature V1")+

   theme(plot.title = element_text(hjust =   0.5),

         plot.caption = element_text(color="red",size=10))+

   ylab("Count")+xlab("V1")

p
```

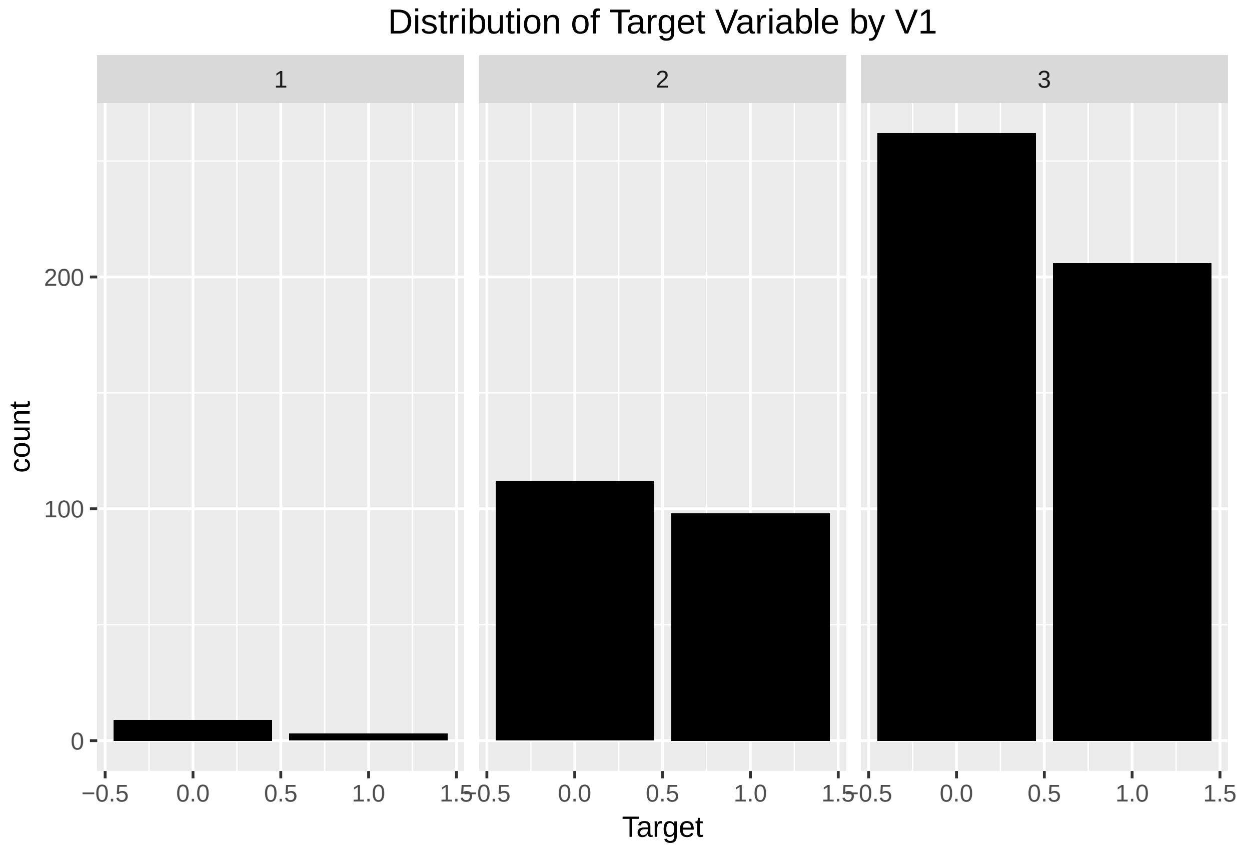


Plot 1

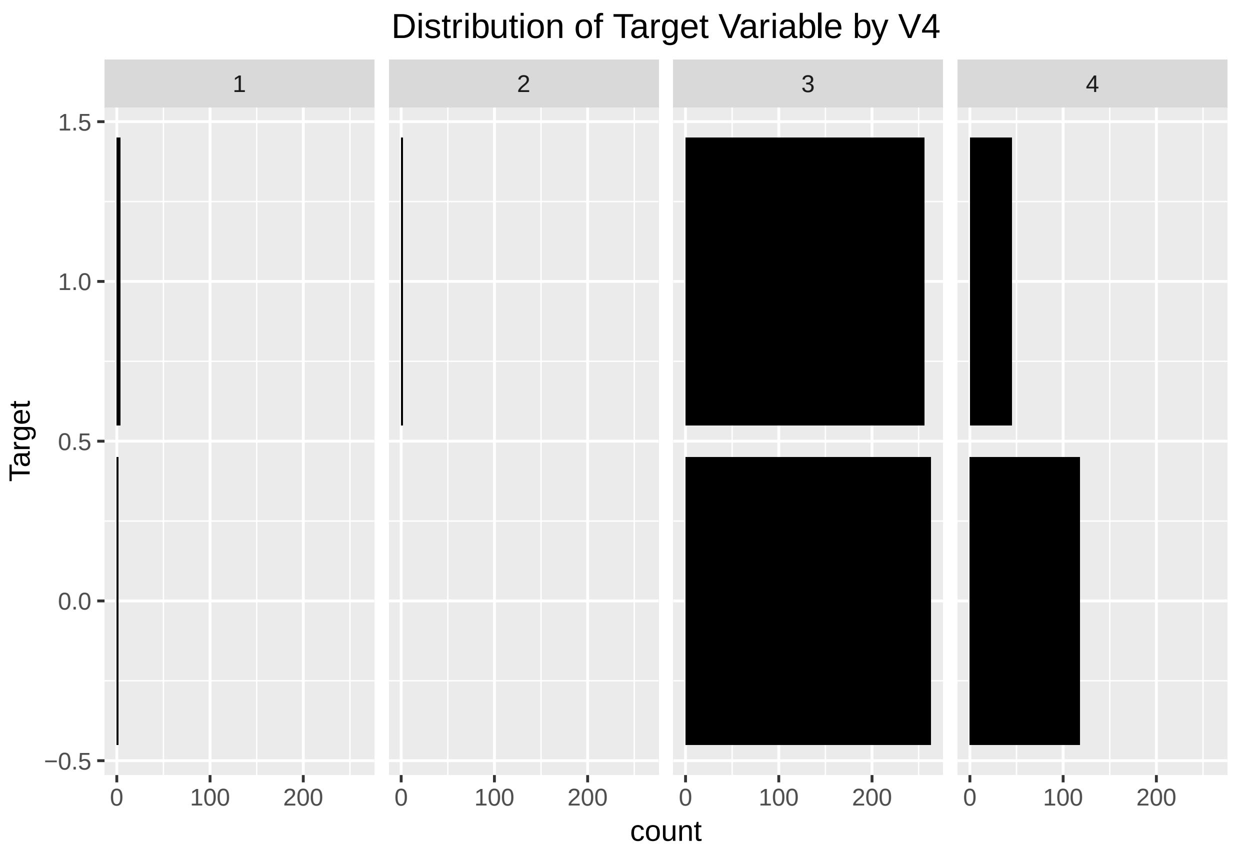
V1, V4 - V7, and V9-V13 are all categorical.

Plot 3 is a factor plot showing the distribution of the target variable by some of the categorical variables such as V1, V4, etc. We are looking for any unbalances that may cause potential issues when building some of the

```
p <- ggplot(data, aes(Target)) + geom_bar(stat = "count",fill="black")
p<-p + facet_grid(cols = vars(V1))
p<-p+ggtitle("Distribution of Target Variable by V1")+ theme(plot.title = element_text(hjust =   0.5))
p
```



```
p <- ggplot(data, aes(Target)) + geom_bar(stat = "count",fill="black")+coord_flip()
p<-p + facet_grid(cols = vars(V4))
p<-p+ggtitle("Distribution of Target Variable by V4")+ theme(plot.title = element_text(hjust =   0.5))
p
```



4. Predictive Model Building

Since this is a classification problem, we are going to be using K-NN, Decision Trees, and SVMs as our choice of models for this. Later ensemble methods such as bagging and boosting will be used to see if we can get higher predictive rate using the models by themselves.

The initial step before the models are initialized with the data for training, we will be splitting the data set into 3 different subsets: Training, Validation, and Test sets.Training Set will be used to build the model, the validation set will be used to fine tune the model, and the test set will be the unseen data to test the model performance. The split for the 3 subsets can be at your discretion but in general a 65/10/25 split is generally used.For this project we will go with a 65/10/25 split and this is validated this below as well.

```
set.seed(123)
ind.train<- sort(sample(1:nrow(data),replace = FALSE, size = floor(.65*nrow(data))))
ind.left<-setdiff(1:nrow(data),ind.train)
ind.val<-sort(sample(ind.left,replace = FALSE, size = floor(.10*nrow(data))))
ind.test<-setdiff(ind.left,ind.val)
print(paste( round(length(ind.train)/nrow(data),2), round(length(ind.val)/nrow(data),2), round(length(in
```

```
## [1] "0.65 0.1 0 2"
```

Now we will split apart the data set into the respective subsets.

```
train.x=data[ind.train, !(names(data) %in% c("Target") )]
train.y=data[ind.train, c("Target")]
val.x=data[ind.val, !(names(data) %in% c("Target") )]
val.y=data[ind.val, c("Target")]

test.x=data[ind.test, !(names(data) %in% c("Target") )]
test.y=data[ind.test, c("Target")]
```

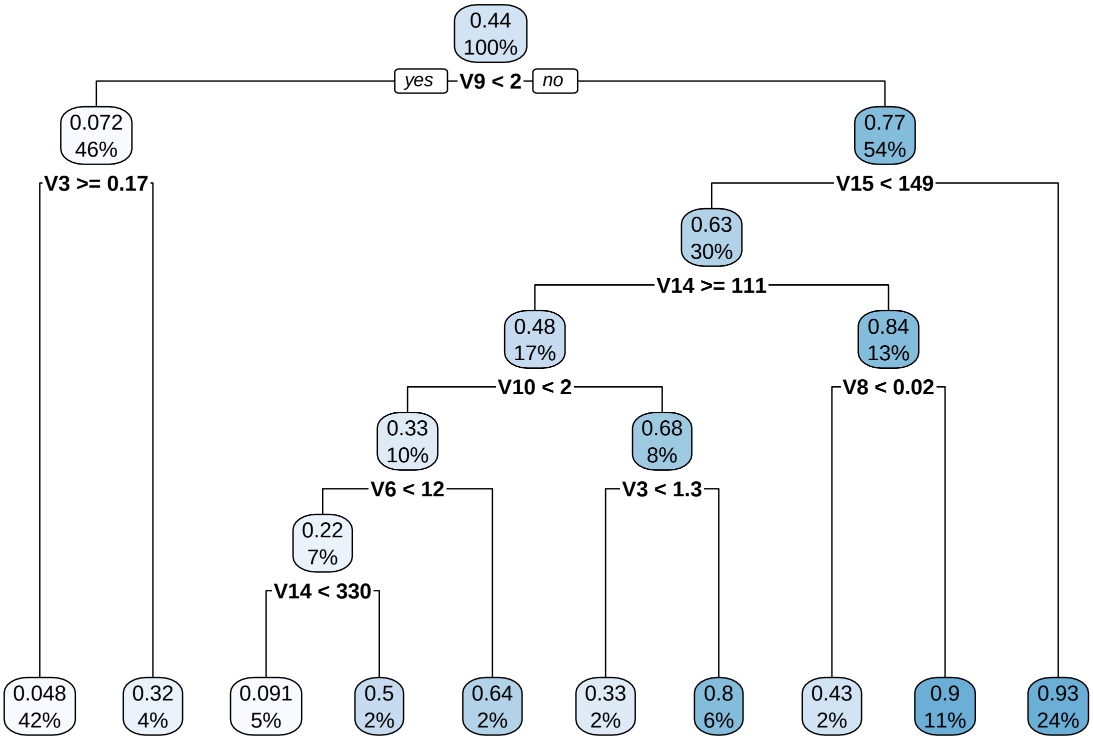
Train K-NN Model first as shown below, and then fine tune the model using the validation set. We will then create the confusion matrix which shows the number of correct/wrong predicted values. It appears k=7 is the best and we don't choose a low number for the clusters since that introduces a large amount of variance into the model.

```
library("class")

misclass.rate=c()

   for (i in 1:50){
      test.pred<-knn(train.x,test.x,cl=train.y, k=i)
      tab1<-table(test.y ,test.pred)
      misclass.rate[i]=round((tab1[2]+tab1[3])/nrow(test.x),2)*100

   }
which(misclass.rate==min(misclass.rate))
```

```
## [1] 7
```

`rpart` will be used to build the decision trees and **`rpart.tool`** will be used to create the decision tree plot.

```
library("rpart")
library("rpart.plot")

train.new<-train.x
train.new["Target"]<-train.y
fit1<-rpart(Target~.,data=train.new)
rpart.plot(fit1)
```



Here the decision tree did better then the K-NN model and has a classification rate of 83%.

```
preds.tree<-predict(fit1,test.x)
preds.tree<-ifelse(preds.tree>=0.5,1,0)
tab1<-table(test.y ,preds.tree)
print(paste0("The misclassification rate on the training set is ", round((tab1[2]+tab1[3])/nrow(test.x)
```

```
## [1] "The misclassification rate on the training set is 13%"
```

By using boosting, we were able to get a classification rate of 84%. The only issue with boosting is that the model itself cannot be easily visualized since there are 5000 trees in this case whose observations are averages. Nonetheless it did pretty well.

```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
boost.myData<-gbm(Target~.,data=train.new,n.trees=5000,interaction.depth=5)
```

```
## Distribution not specified, assuming bernoulli ...
```

```
pred.boost<-predict(boost.myData,newdata=test.x,n.trees=5000,type="response")
pred.boost<-ifelse(pred.boost>=0.5,1,0)
tab1<-table(test.y ,pred.boost)
print(paste0("The misclassification rate on the training set is ", round((tab1[2]+tab1[3])/nrow(test.x)
```

```
## [1] "The misclassification rate on the training set is 14%"
```