Predicting Bankruptcy

Aaron David Valdivia

3/11/2021

Introduction

For this project we will be looking at a data considering compiled from the Taiwan Economic Journal from 1999 to 2009. The table lists 6819 companies, with 95 quantitative attributes as well as whether or not they went bankrupt according to the buisness regulations of the Taiwan Stock Exchange. The data set is included in the submission but can also be found on Kaggle at https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction.

Our goal will be to predict whether a company goes bankrupt given the quantitative attributes of the company. We will attempt to build two different models one using logistic regression and the other using random forests. We will also employ the use of principle component analysis in order to find uncorrelated and highly variable collections of features.

Analysis

First off we will load our data set and clean the variable names. We will label each feature $X1$ through $X95$ and the response variable will be labeled Y. We will also convert the response variable to a factor.

```
data<-read.csv("data.csv")
cdata<-data %>%
  setNames(paste0("X", seq_along(.) - 1)) %>%
  rename(Y = 1)
cdata$Y<-factor(cdata$Y)
```

We can also look at the summary of the data. After doing so a quick look we can see that one variable looks odd.

```
summary(cdata)[,95]
```

```
##
## "Min.   :1  " "1st Qu.:1  " "Median :1  " "Mean   :1  " "3rd Qu.:1  "
##
## "Max.   :1  "
```

This variable has no variance and so is useless for our models. We remove it and then partition our data set into training and test sets.

```
remove<-which(apply(cdata,2,sd)==0)
cdata<-cdata[,-remove]
set.seed(1)
index<-createDataPartition(cdata$Y,times=1, p=.3, list=FALSE)
test_set<-cdata[index,]
train_set<-cdata[-index,]
```

Logistic Regression Being that this is a classification problem we might try naively to train a logistic regression model, on say, the first variable.

```
fit<-train(Y~X1,method="glm",data=train_set)
prediction<-predict(fit,test_set)
confusionMatrix(prediction,test_set$Y)
```

We can see that accuracy is very high but mostly because there are very few bankruptcies. If we instead consider Cohen's Kappa we are not doing so well. We may also try naively to fit a model using all the variables.

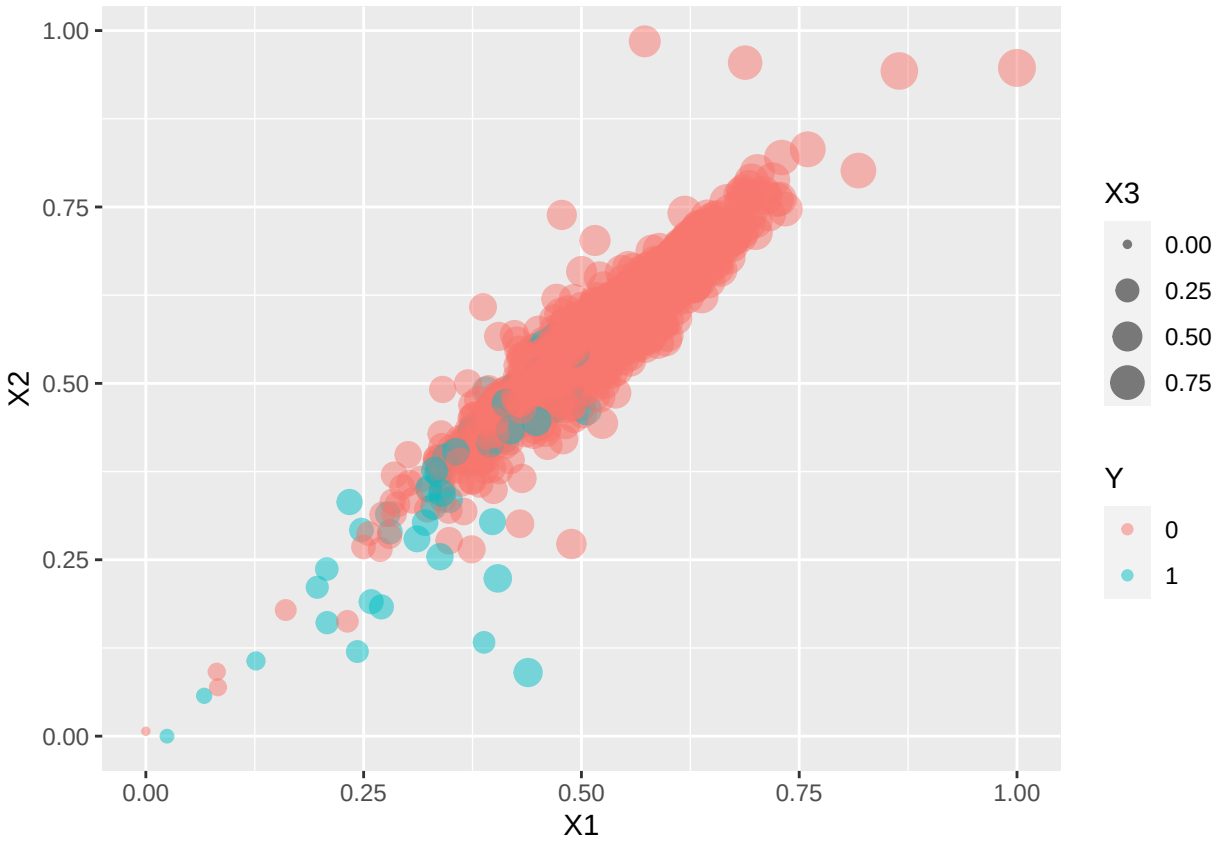
```
fit<-train(Y~.,method="glm",data=train_set, maxit=200,metric="Kappa")
prediction<-predict(fit,test_set)
confusionMatrix(prediction,test_set$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1957   53
##          1   23   13
##
##                Accuracy : 0.9629
##                  95% CI : (0.9537, 0.9706)
##     No Information Rate : 0.9677
##     P-Value [Acc > NIR] : 0.9034119
##
##                   Kappa : 0.2375
##
##  Mcnemar's Test P-Value : 0.0008794
##
##             Sensitivity : 0.9884
##             Specificity : 0.1970
##          Pos Pred Value : 0.9736
##          Neg Pred Value : 0.3611
##              Prevalence : 0.9677
##          Detection Rate : 0.9565
##    Detection Prevalence : 0.9824
##       Balanced Accuracy : 0.5927
##
##        'Positive' Class : 0
##
```

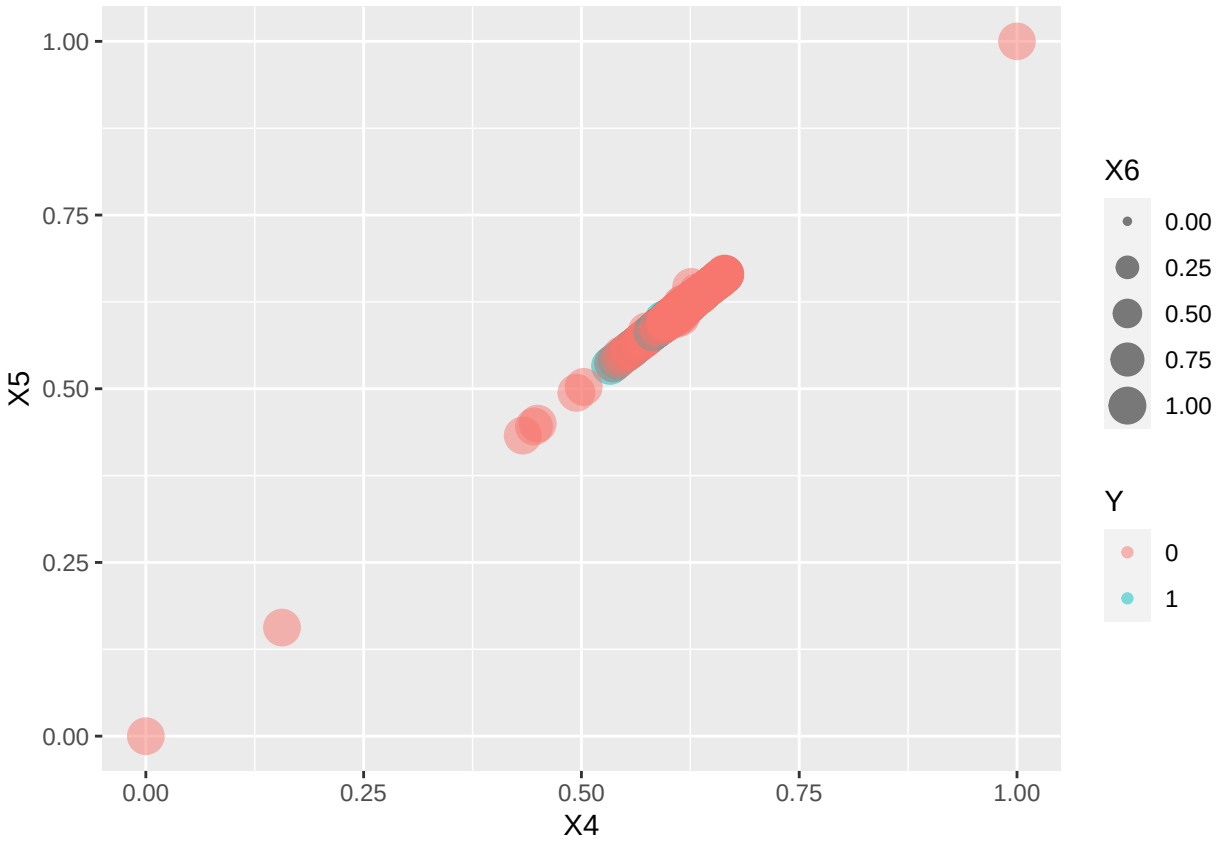
Here we see an increase in Kappa but we would hope to do better. We begin by considering the relationships between a selection of different variables.

We will plot 3 varaibles at a time through the 18th variable. Using 2 spatial dimensions, the size of the point as the third and color to distinguish between bankrupcy statuses.

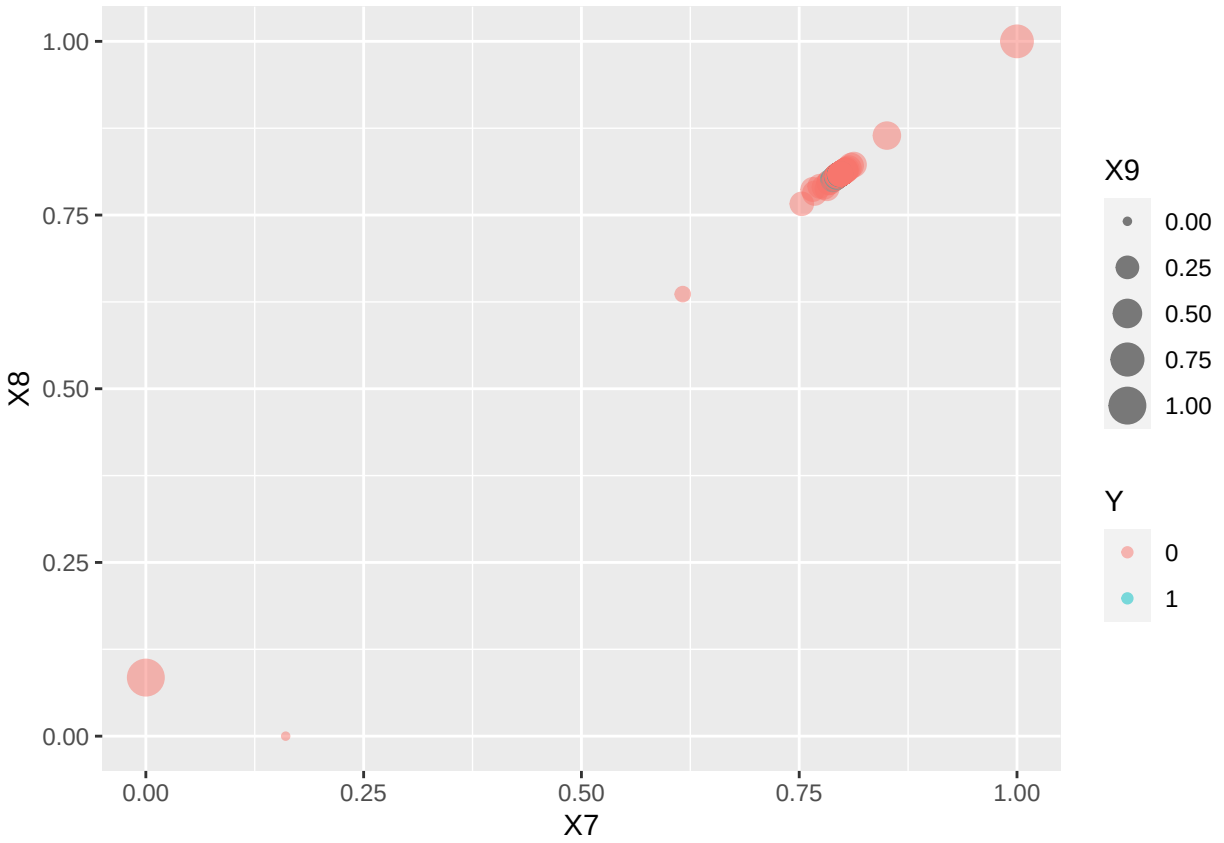
```
train_set %>% ggplot(aes(x=X1,y=X2,color=Y,size=X3))+geom_point(alpha=.5)
```



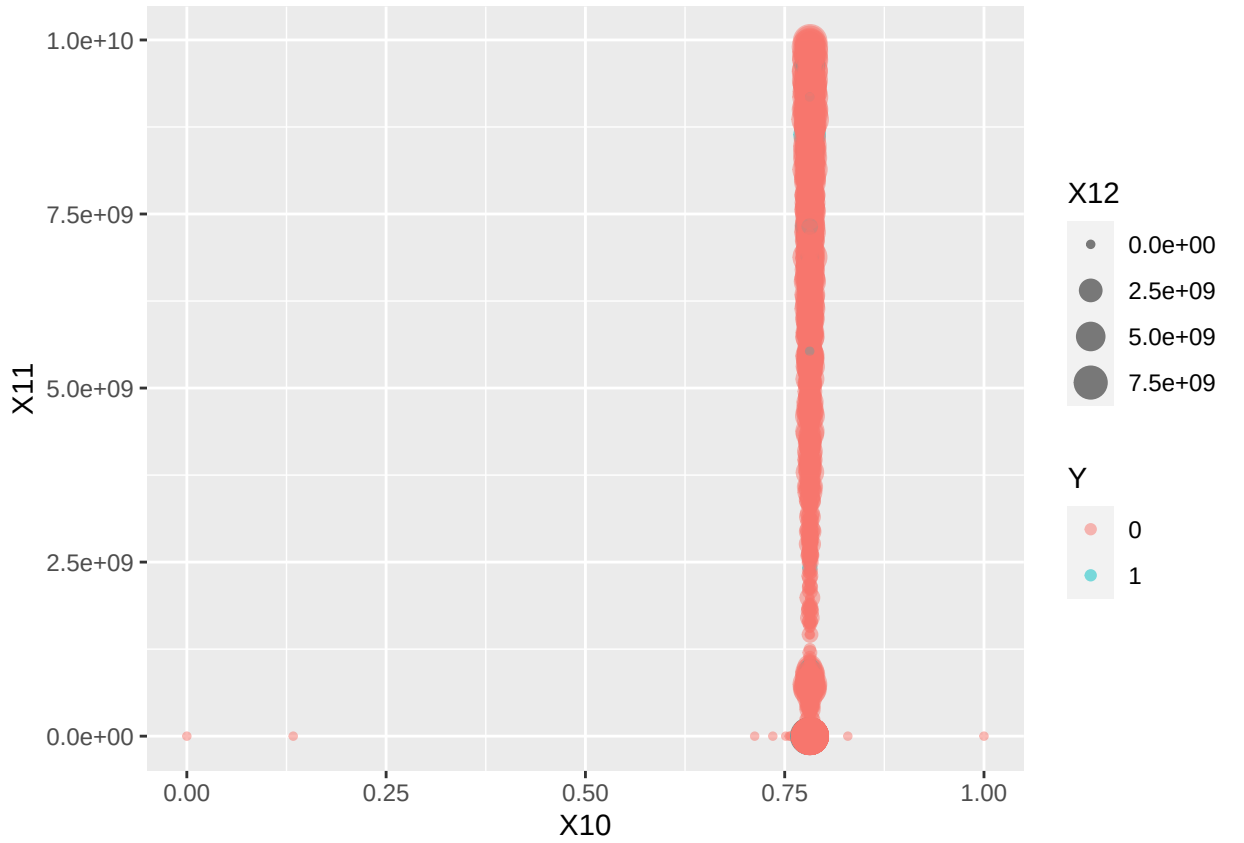

```
train_set %>% ggplot(aes(x=X4,y=X5,color=Y,size=X6))+geom_point(alpha=.5)
```



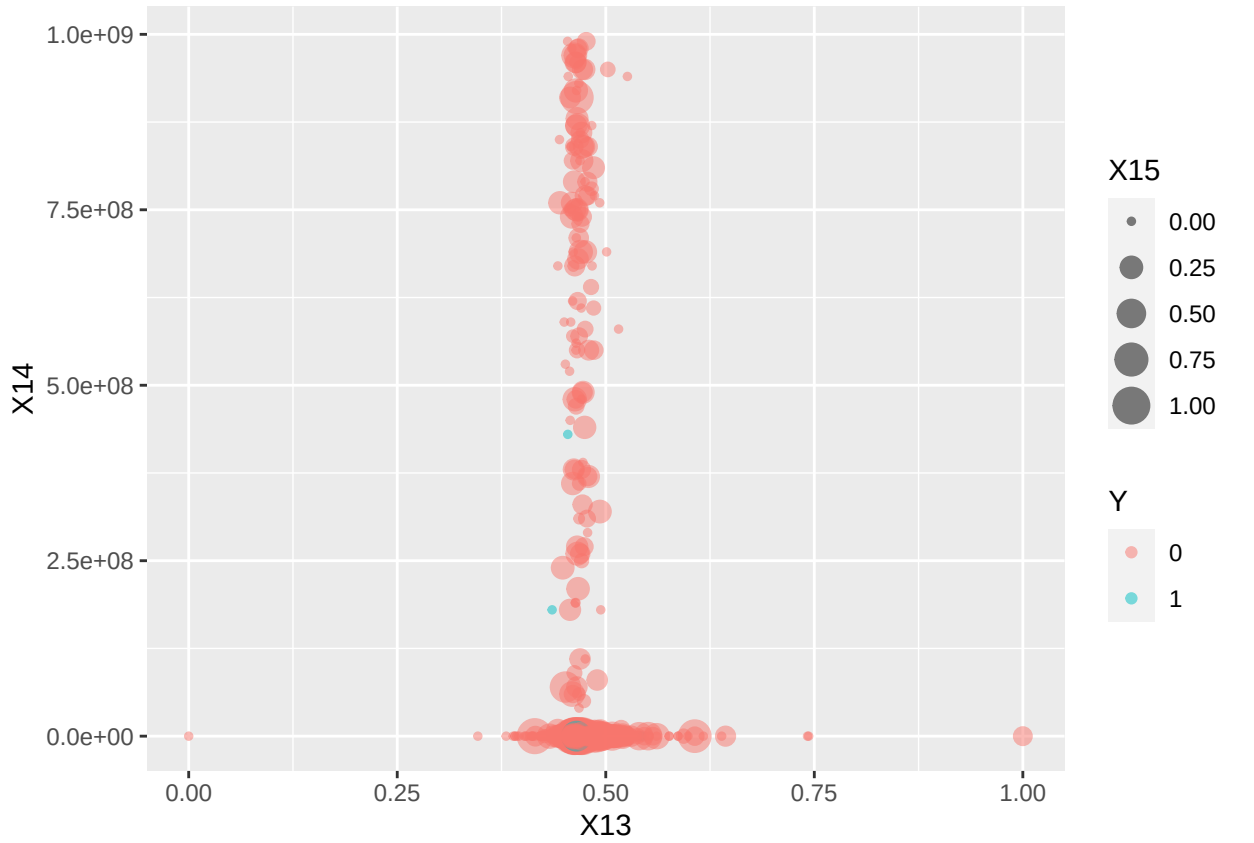

```
train_set %>% ggplot(aes(x=X7,y=X8,color=Y,size=X9))+geom_point(alpha=.5)
```



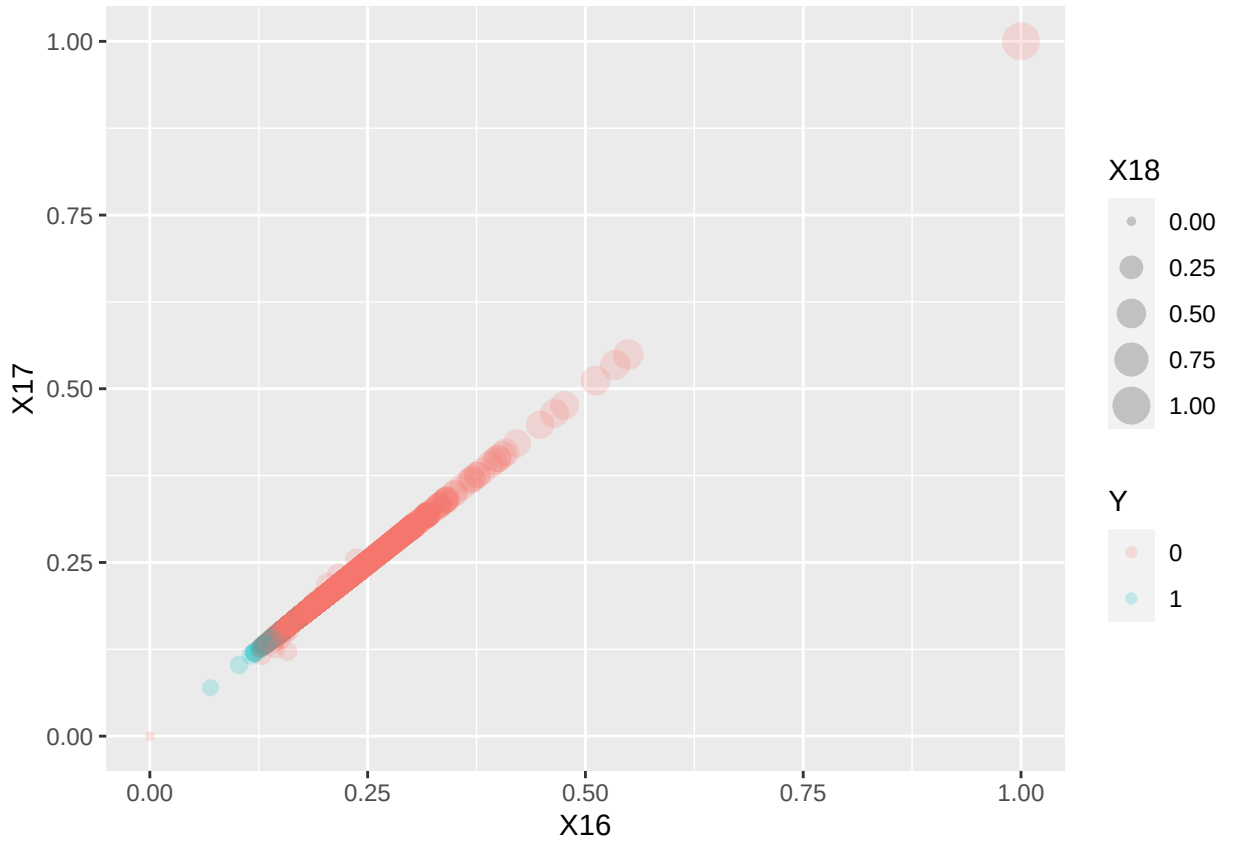

```
train_set %>% ggplot(aes(x=X10,y=X11,color=Y,size=X12))+geom_point(alpha=.5)
```




```
train_set %>% ggplot(aes(x=X13,y=X14,color=Y,size=X15))+geom_point(alpha=.5)
```




```
train_set %>% ggplot(aes(x=X16,y=X17,color=Y,size=X18))+geom_point(alpha=.2)
```



We can see from these graphs that many variables are highly correlated while others are seemingly not correlated much at all. We want to use collections of features that have high variability and are also not correlated, which would lead to over-training. We will use principle component analysis to overcome this issue. We also will tune our model for the number of principle components to use.

Random Forests In addition to logistic regression we will model our data using the random forest algorithm. Using this algorithm we will tune over the the number of predictors used in each tree. The algorithm takes quite a while and does not produce the best results. The code for the algorithm can be seen below but we will not run the results here.

Using a single tree we obtain a kappa value of .2737.

```
levels(train_set$Y)<-c("NB","B")
levels(test_set$Y)<-c("NB","B")
set.seed(1)
fit <- train(Y~ ., method = "rpart",
             tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
             data = train_set,metric="Kappa")
confusionMatrix(predict(fit, test_set), test_set$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NB    B
##         NB 1963   52
##         B    17   14
##
```

```
##                Accuracy : 0.9663
##                  95% CI : (0.9575, 0.9737)
##     No Information Rate : 0.9677
##     P-Value [Acc > NIR] : 0.675
##
##                   Kappa : 0.2737
##
##  Mcnemar's Test P-Value : 4.256e-05
##
##             Sensitivity : 0.9914
##             Specificity : 0.2121
##          Pos Pred Value : 0.9742
##          Neg Pred Value : 0.4516
##              Prevalence : 0.9677
##          Detection Rate : 0.9594
##    Detection Prevalence : 0.9848
##       Balanced Accuracy : 0.6018
##
##        'Positive' Class : NB
##
```

We can try to improve on these results using the random forest algorithm.

```
#fit <- train(Y~ ., method = "Rborist", data = train_set,
#             tuneGrid=data.frame(predFixed=1:10,minNode=2),
#             metric="Kappa")
#confusionMatrix(predict(fit,test_set),test_set$Y)
```

We can also improve our results with random forests by using principle components here as well. We do so with the following code.

```
#PCA<-preProcess(train_set[,-1],method = "pca",pcaComp = 90)
#trainpc<-cbind(train_set$Y,predict(PCA,train_set[,-1]))
#fit<-train(`train_set$Y`~.,method="Rborist",data=trainpc,
#           tuneGrid=data.frame(predFixed=1:10,minNode=2),metric="Kappa")
#testpc<-predict(PCA,test_set[,-1])
#prediction<-predict(fit,testpc)
#confusionMatrix(prediction,test_set$Y)
```

Results

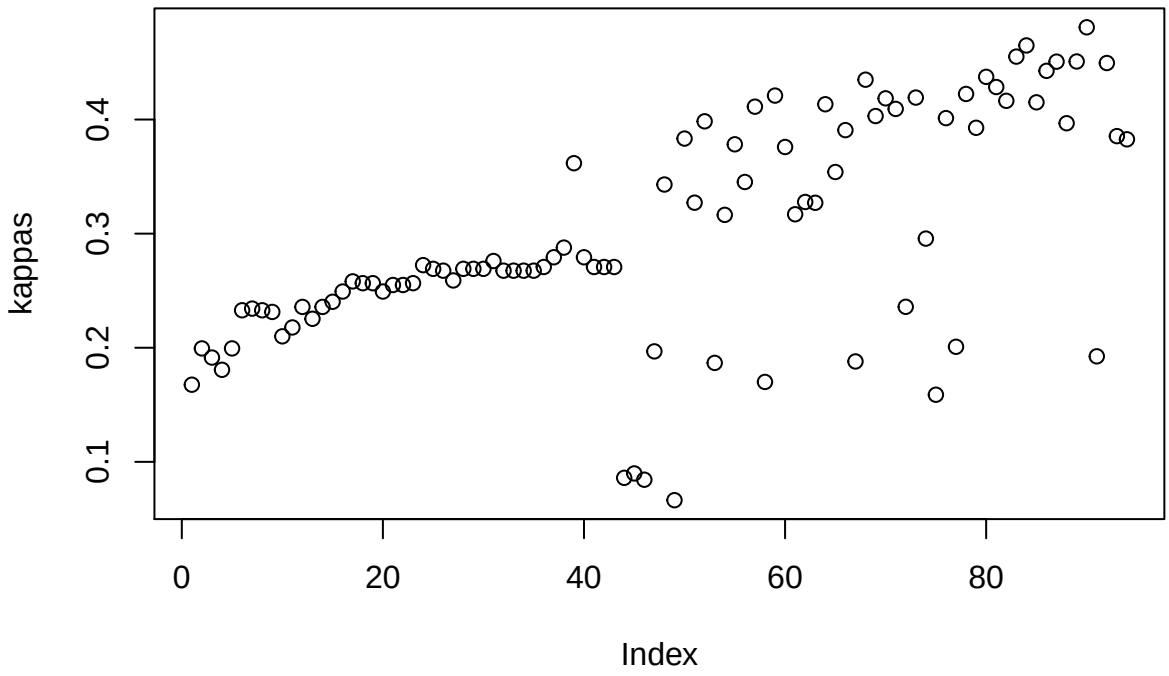
The logistic regression model using all the all the available variables had a Kappa value of .1764 where when using the optimum number of principle components we managed a better value of Kappa. A graph of kappa for all numbers of principle components is shown below.

```
tuner<-function(N){
  PCA<-preProcess(train_set[,-1],method = "pca",pcaComp = N,metric="Kappa")
  trainpc<-cbind(train_set$Y,predict(PCA,train_set[,-1]))
  fit<-train(`train_set$Y`~.,method="glm",data=trainpc)

  prediction<-predict(fit,trainpc)
  CM<-confusionMatrix(prediction,train_set$Y)
  CM$overall[[2]]
}

kappas<-sapply(1:94,tuner)
```

```
plot(kappas)
```



Evaluating on the test set gives the following results which we can see are improved.

```
PCA<-preProcess(train_set[,-1],method = "pca",pcaComp = 90)
testpc<-cbind(test_set$Y,predict(PCA,test_set[,-1]))
fit<-train(`test_set$Y`~.,method="glm",data=testpc)

prediction<-predict(fit,testpc)
confusionMatrix(prediction,test_set$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NB    B
##         NB 1970   46
##         B    10   20
##
##                Accuracy : 0.9726
##                  95% CI : (0.9646, 0.9793)
##     No Information Rate : 0.9677
##     P-Value [Acc > NIR] : 0.1155
##
##                   Kappa : 0.4047
##
##  Mcnemar's Test P-Value : 2.91e-06
```

```
##
##             Sensitivity : 0.9949
##             Specificity : 0.3030
##          Pos Pred Value : 0.9772
##          Neg Pred Value : 0.6667
##              Prevalence : 0.9677
##          Detection Rate : 0.9629
##    Detection Prevalence : 0.9853
##       Balanced Accuracy : 0.6490
##
##        'Positive' Class : NB
##
```

The random forest algorithm received a Kappa value of 0.1563 when tuned for the number of predictors and when using principle component analysis increased as can be seen below.

```
PCA<-preProcess(train_set[,-1],method = "pca",pcaComp = 90)
trainpc<-cbind(train_set$Y,predict(PCA,train_set[,-1]))
fit<-train(`train_set$Y`~.,method="Rborist",data=trainpc,
           tuneGrid=data.frame(predFixed=1:10,minNode=2),metric="Kappa")
testpc<-predict(PCA,test_set[,-1])
prediction<-predict(fit,testpc)
confusionMatrix(prediction,test_set$Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   NB    B
##         NB 1974   63
##         B     6    3
##
##                Accuracy : 0.9663
##                  95% CI : (0.9575, 0.9737)
##     No Information Rate : 0.9677
##     P-Value [Acc > NIR] : 0.675
##
##                   Kappa : 0.0728
##
##  Mcnemar's Test P-Value : 1.566e-11
##
##             Sensitivity : 0.99697
##             Specificity : 0.04545
##          Pos Pred Value : 0.96907
##          Neg Pred Value : 0.33333
##              Prevalence : 0.96774
##          Detection Rate : 0.96481
##    Detection Prevalence : 0.99560
##       Balanced Accuracy : 0.52121
##
##        'Positive' Class : NB
##
```

As a final note a number of warnings may be seen when running the code from the R script. They have been suppressed in the R-Markdown file for formatting's sake. The warnings are all about probabilities converging to 0 or 1 which literature says is expected when a prediction is highly probable.

Conclusion

The models that we developed give us some insight into which companies might go bankrupt. Similar methods could be used to check for poor business practices or companies that are in financial trouble. The best model that we found was the logistic regression model with principal component analysis tuned for the number of principal components used. WE acheived a Kappa Value of .4047 Future work into improving these results might focus on K-nearest-neighbors with some collection of variables possibly with some transformation like principle component analysis done. One might also consider the relationship between the principle components and the response variable; just because the component has high variability does not mean that it differentiates well with respect to the response variable.