The Application of Probit, K-Nearest Neighbors, Tree Methods and Support Vector Machine on Predicting U.S. Recession

Haoyang Lyu

July 2023

1 Introduction

Since the 1980s, economists have argued about the predicting power of the slope of the yield curve - the spread between long and short term interest rate - on the U.S. economic activity. Based on this, many economists and research institution have built models trying to predict the economic activity of the U.S. One famous one is the U.S. Recession Probability Model created by **Federal Reserve Bank of New York** using Probit. This report will combine the idea of the Probit and some tradition machine learning methods on predicting the U.S. Recession.

The slope of yield curve should be an excellent indicator of future economic events for this reason:

- Current monetary policy has a significant influence on the yield curve and the economic activity. A rise of short-term rate tends to flatten the yield curve and the real economic growth.

In this report, a few other financial indicators will be included for analysis. One being stock prices. Finance theory suggests that stock prices are determined by expectations about future dividend streams, which is tightly related to the future state economy. Money supply (M0, M1, M2, M3) will also be included in the analysis since it's tightly connected to credit market which is connected to the expectations for the future economy. A few other financial variables will also be included.

---

2 Methodology

In this report, three methods will be used for analyzing and predicting the future economic state of the U.S.

- Probit
- K Nearnest Neighbors
- Decision Tree Methods

We will going deeper for each one at its corresponding analysis page. Since we are predicting whether the U.S. will have recession in the future, we will be using indicator variable here: 1 being "in recession" and 0 being "not in recession" for specific period of time.

---

3 Data Descriptions

- **spread** : Term spread from the difference between 10-year and 3-month Treasury rates.(Monetary Policy has huge influence on both yield curve and real economic activity)
- **sp500**: S&P 500 index.(Future dividend streams)
- **cpi**: Consumer Price Index.
- **m0**: Monetary Base.(Monetary Base and M1, M2, M3 forms opinions on future credit market)
- **m1**: M1
- **m2**: M2
- **m3**: M3
- **unem_rate**: Unemployment Rate
- **new_private_house**: New Privately Own Housing Units
- **consumer_senti**: University of Michigan Consumer Sentiment
- **real_gdp**: Real GDP Growth
- **vendor_performance**: Vendor Performance

3.1 Manipulate the data

Since some data such as real gdp, money aggregate, cpi etc. are posted in lag or on a quarterly basis. All the data have been adjusted to indicate the most up-to-date information that we have at the analysis at each time period.

3.2 Feature Selection

Since we are using many variables, feature selection is performed in this case to select optimal set of variable to be used in the final analysis:

- Correlation Matrix is being used here to see the correlation coefficients between every two variables.
- Best Subset Selection

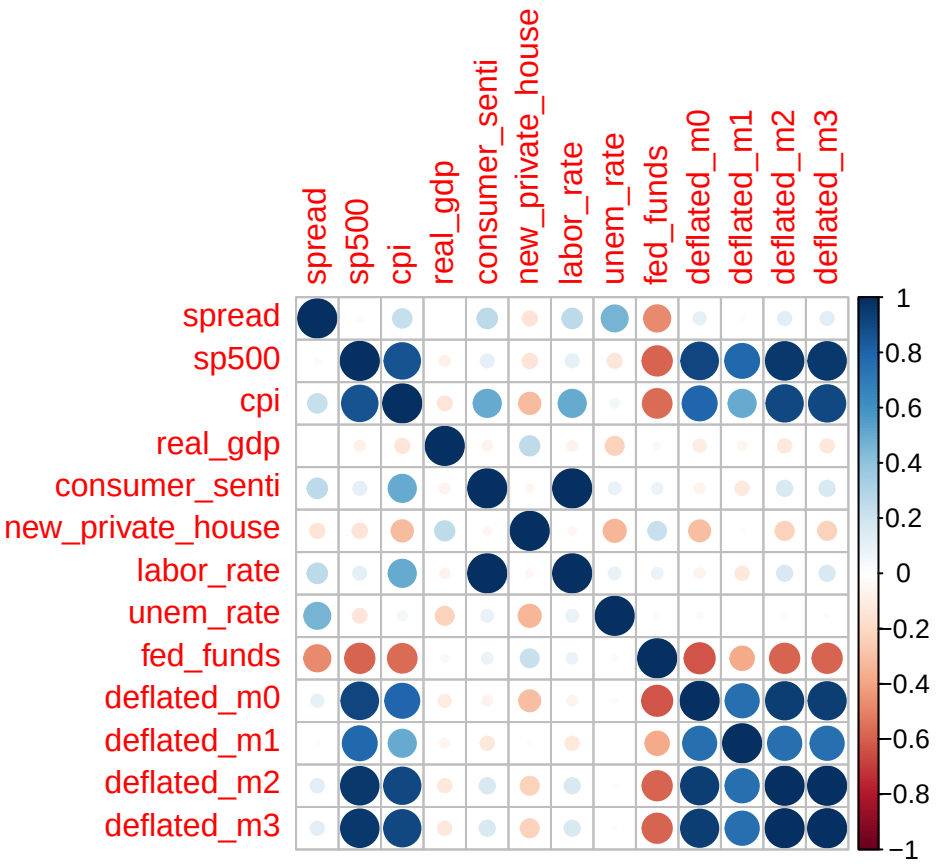
Correlation Matrix:

The following 12 varibales are being used for feature selection: spread, sp500, cpi, consumer_senti, unem_rate, fed_funds, new_private_house, deflated_m0, deflated m1, deflated m2, deflated m3, real_gdp.

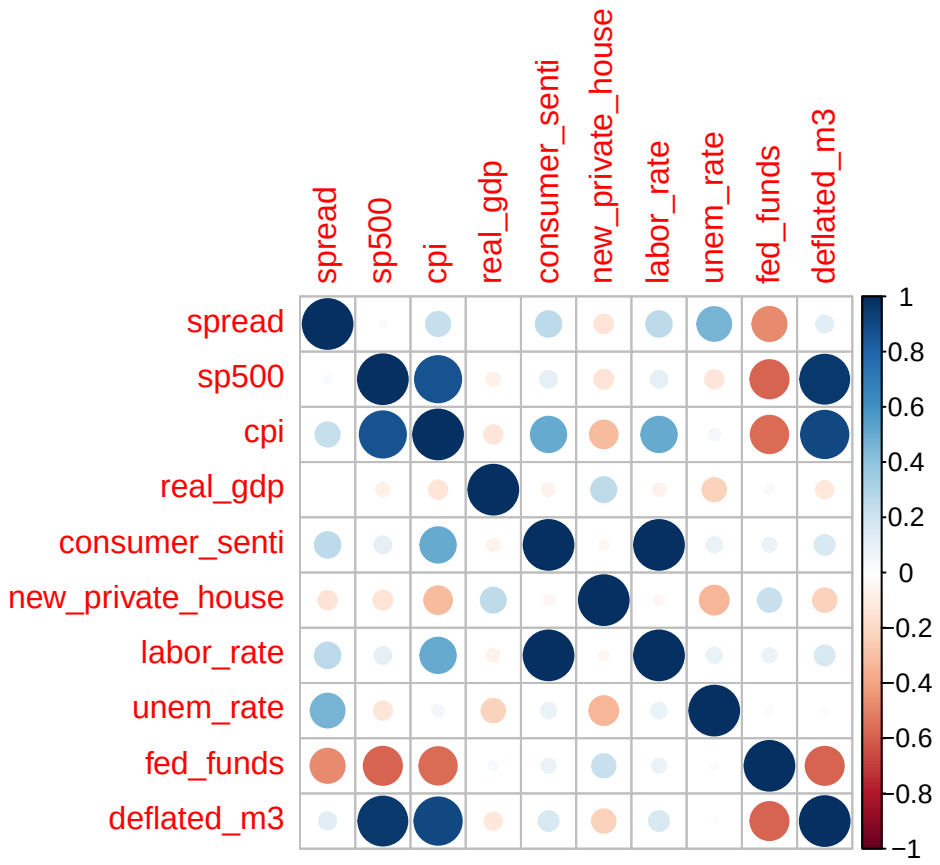
Through the correlation matrix, we notice that deflated m0, deflated m1, deflated m2, deflated m3, cpi and sp500 correlate with each other. By eliminating deflated m0, deflated m1 and deflated m2. We get the updated correlation matrix.

```
correlation_matrix <- cor(
  recession_data%>%
    dplyr::select(c( "spread",
                     "sp500",
                     "cpi",
                     "real_gdp",
                     "consumer_senti",
                     "new_private_house",
                     "labor_rate",
                     "unem_rate",
                     "fed_funds",
                     "deflated_m0",
                     "deflated_m1",
```

```
                      "deflated_m2",
                      "deflated_m3"
                      ))
)

corrplot(correlation_matrix)
```




```
updated_correlation_matrix <- cor(
  recession_data%>%
    dplyr::select(c( "spread",
                      "sp500",
                      "cpi",
                      "real_gdp",
                      "consumer_senti",
                      "new_private_house",
                      "labor_rate",
                      "unem_rate",
                      "fed_funds",
                      "deflated_m3"
                      ))
)

corrplot(updated_correlation_matrix)
```

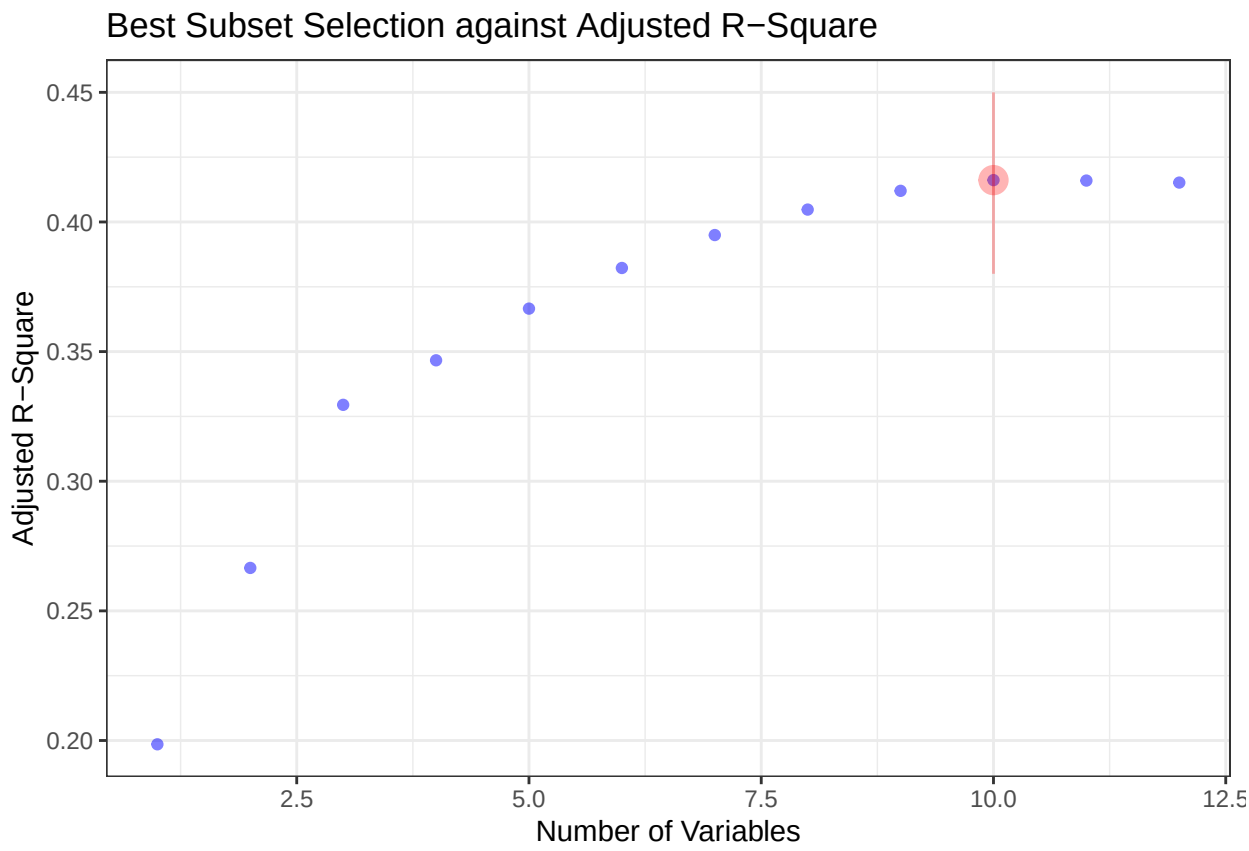


Best Subset Selection:

The following 12 varibales are being used for feature selection: spread, sp500, cpi, consumer_senti, unem_rate, fed_funds, new_private_house, deflated_m0, deflated m1, deflated m2, deflated m3, real_gdp.

By performing best subset selection in this case, we noticed that the best model is the one with 10 variable by using Adjusted R-Square. The following variable is being selected: spread, sp500, consumer_senti, unem_rate, fed_funds, new_private_house, deflated m0, deflated m1, deflated m3, real_gdp

```
subset_full <- regsubsets(one_year_rec ~ spread + sp500 + cpi + consumer_senti + unem_rate + fed_funds +
subset_summary <- summary(subset_full)
subset_result <- cbind(1:12,subset_summary$adjr2)%>%
  as.data.frame()

ggplot(subset_result, aes(x = V1, y = V2)) +
   geom_point(size = 1.5, alpha = 0.5, color = "blue") +
   theme_bw() +
   ggtitle("Best Subset Selection against Adjusted R-Square") +
   xlab("Number of Variables") +
   ylab("Adjusted R-Square") +
   annotate("pointrange", x = 10, y = 0.4161428, ymin = 0.38, ymax = 0.45, size = 1, alpha = 0.3, color
```



4 Probit

4.1 Probit Regression Model Descriptions:

Probit Regression is a traditional econometrics regression model where the dependent variable is binary which only takes two values (indicator variables). In our case, we take two values 1 being "in recession" and 0 being "not in recession". Probit Regression Model is estimated through Maximum Likelihood Procedure.

Probit Regression Model takes the following form:

$P(Y = 1|X) = \phi(X^T\beta)$ where ϕ is the cumulative distribution function.

In the probit regression model, a latent variable is being introduced.

$Y^* = X^T\beta + \epsilon$ and $Y = 1[Y^* \geq 0]$

Based on above,

$P(Y = 1|X) = P(1[Y^* \geq 0]) = P(Y^* \geq 0|X) = P(X^T\beta + \epsilon \geq 0|X)$

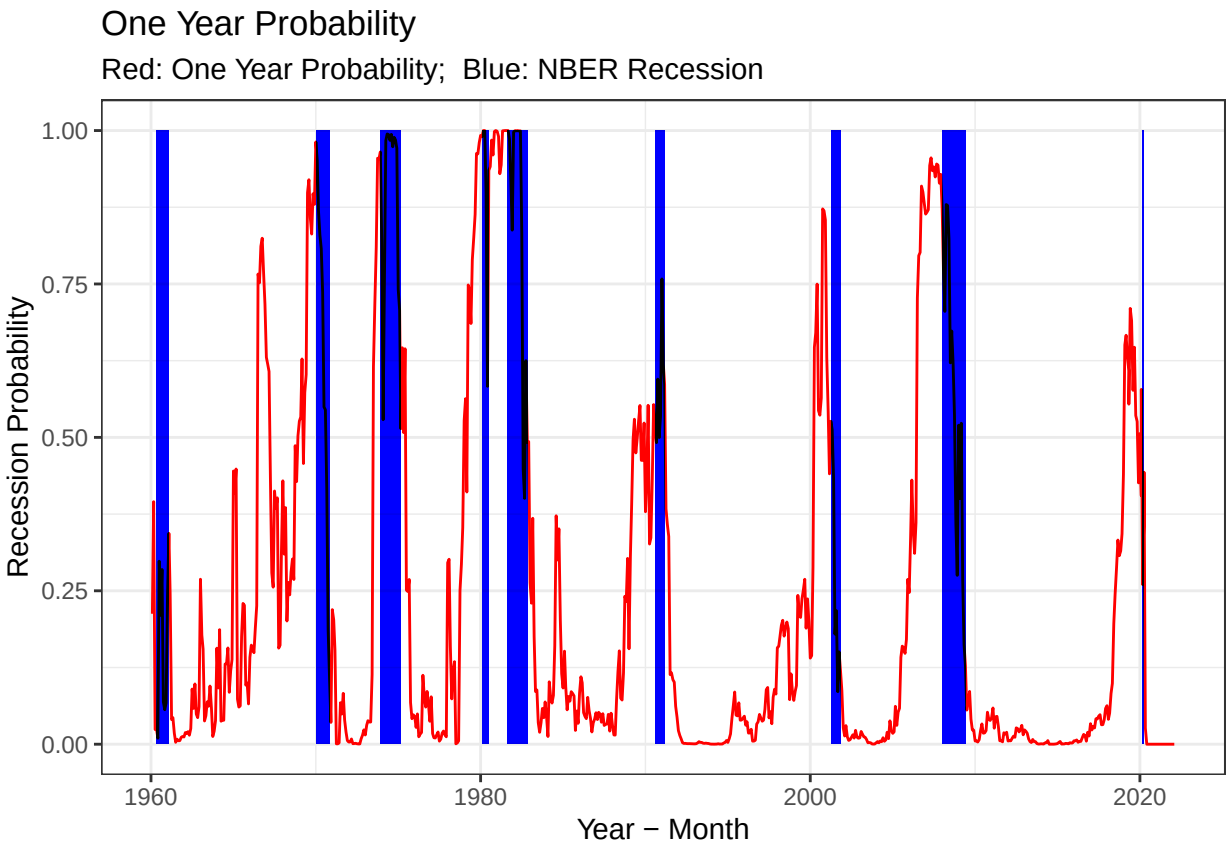
To calculate the Probability of $P(X^T\beta + \epsilon \geq 0|X)$, Assuming the ϵ belongs to a Normal Distribution, then we have the following:

$P(\epsilon \geq -X^T\beta|X)$ which equals to $\phi(X^T\beta)$

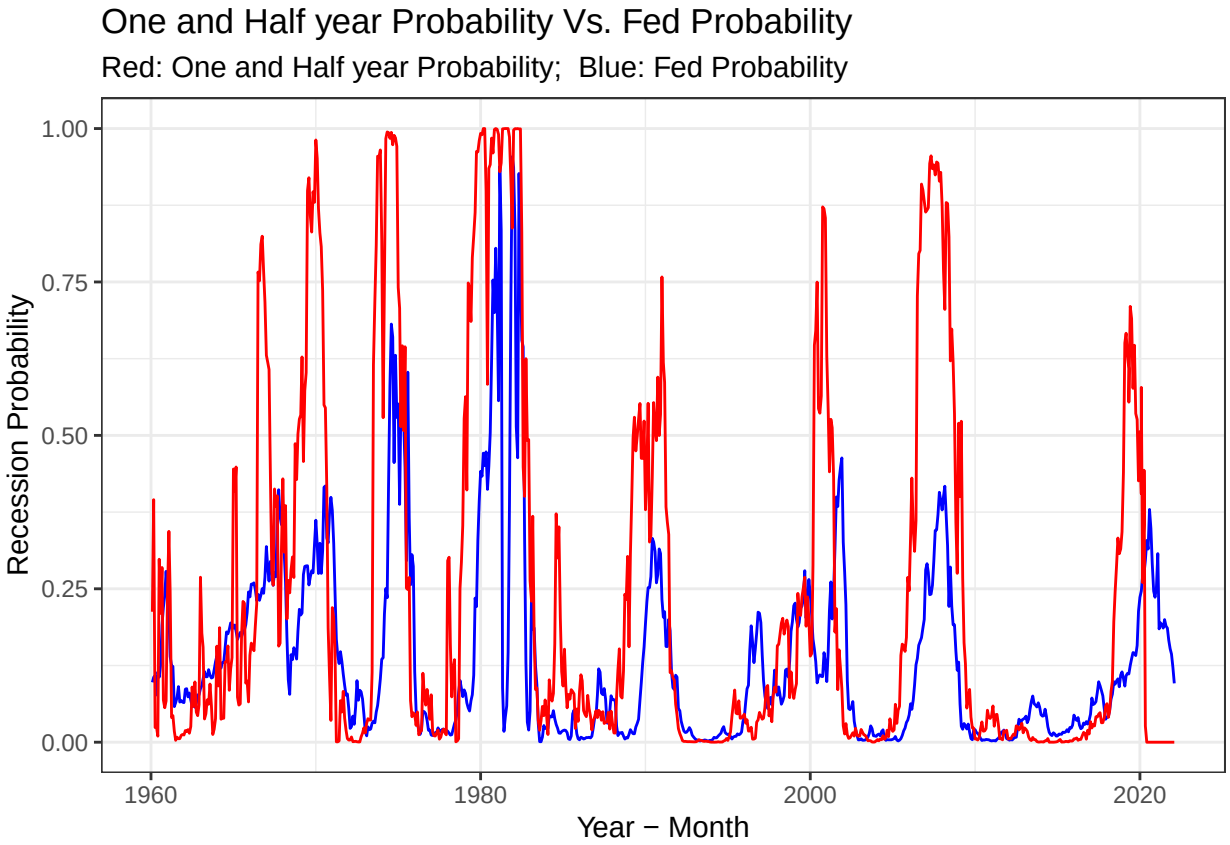
```r
one_year_probit <- glm(one_year_rec ~ spread + sp500 + consumer_senti + unem_rate + fed_funds + new_pri

one_half_year_probit <- glm(one_half_year_rec ~ spread + sp500 + consumer_senti + unem_rate + fed_funds

one_year_rec_prob <- predict(one_year_probit, type = "response")
one_half_year_rec_prob <- predict(one_half_year_probit, type = "response")

rec_prob <- cbind(recession_data$year_month, recession_data$fed_rec_prob) %>%
  cbind(one_year_rec_prob) %>%
  cbind(one_half_year_probit) %>%
  set_colnames(c("year_month", "fed_rec_prob", "one_year_rec_prob", "one_half_year_probit"))%>%
  as.data.frame() %>%
  mutate(year_month = as.Date(paste(year_month, "-01", sep = ""))) %>%
  mutate(fed_rec_prob = as.numeric(fed_rec_prob))%>%
  mutate(one_year_rec_prob = as.numeric(one_year_rec_prob))%>%
  mutate(one_half_year_rec_prob = as.numeric(one_half_year_rec_prob))
```

4.2 Probit Model Visulation

```r
## One Year Probability Vs. Fed Probability
ggplot(data = rec_prob, aes(x = year_month)) +
  geom_line(aes( y = one_year_rec_prob), color = "red")+
  geom_rect(aes(xmin = as.Date("1960-05-01"), xmax = as.Date("1961-02-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("1970-01-01"), xmax = as.Date("1970-11-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("1973-12-01"), xmax = as.Date("1975-03-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("1980-02-01"), xmax = as.Date("1980-07-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("1981-08-01"), xmax = as.Date("1982-11-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("1990-08-01"), xmax = as.Date("1991-03-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("2001-04-01"), xmax = as.Date("2001-11-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("2008-01-01"), xmax = as.Date("2009-06-01"), ymin = 0, ymax = 1), fill =
  geom_rect(aes(xmin = as.Date("2020-03-01"), xmax = as.Date("2020-04-01"), ymin = 0, ymax = 1), fill =
  theme_bw() +
  theme(legend.position = "bottom")+
  ggtitle("One Year Probability", subtitle = "Red: One Year Probability;  Blue: NBER Recession") +
  xlab("Year - Month") +
  ylab("Recession Probability")
```




```
## One and Half Year Probability Vs. Fed Probability
ggplot(data = rec_prob, aes(x = year_month)) +
  geom_line(aes( y = fed_rec_prob), color = "blue") +
  geom_line(aes( y = one_year_rec_prob), color = "red")+
  theme_bw() +
  theme(legend.position = "bottom")+
  ggtitle("One and Half year Probability Vs. Fed Probability", subtitle = "Red: One and Half year Probal
  xlab("Year - Month") +
  ylab("Recession Probability")
```



---

5 K-Nearest Neighbors

5.1 K-Nearest Neighbors Descriptions:

K-Nearest Neighbors is machine learning classifier, which uses proximity to make classification about the groupings of individual data points. The following graphs shows intuitively how KNN works.

When using KNN for classification problem, a classification label is assigned on the basis of a majority vote from the nearest neighbors(The label that is most frequently represented around a given data point(target data point)). Therefore, one goal of the K-nearest neighbors algorithm is to identify the nearest neighbors of a given point. In order to find the nearest neighbors, we need determine the distance metrics to calculate the distance.In this case, we are using **Euclidean Distance**, which is calculated in the following approach:

$Distance(X, Y) = \sqrt{\Sigma(Y_i - X_i)^2}$

By finding the nearest k number of neighbors using the Euclidean Distance, we are able to find the majority vote from them and make the corresponding classification.

Creating Training and Test Set: For machine learning model, we divided data into two parts, one being the training set which will be used to train the model and tune the parameter and the other being the test set which will be used to test the prediction accuracy. In this case, 2/3 of total observations will be used in the training set while the left 1/3 will be used in the test set.

```
training <- sample(1:nrow(recession_data), nrow(recession_data)*(2/3))
training_set <- recession_data[training,]
test_set <- recession_data[-training,]
```

```
features <- c(names(recession_data)[2:5], names(recession_data)[8:23])
train_x <- training_set[,features]
test_x <- test_set[,features]
train_oneyear_y <- training_set[, "one_year_rec"]
train_onehalfyear_y <- training_set[, "one_half_year_rec"]

set.seed(123)

## One year recession prediction
one_year_knn_pred <- knn(train_x, test_x, train_oneyear_y, k = 4)
confusionMatrix(data = factor(one_year_knn_pred),reference = factor(test_set[, "one_year_rec"]))
```

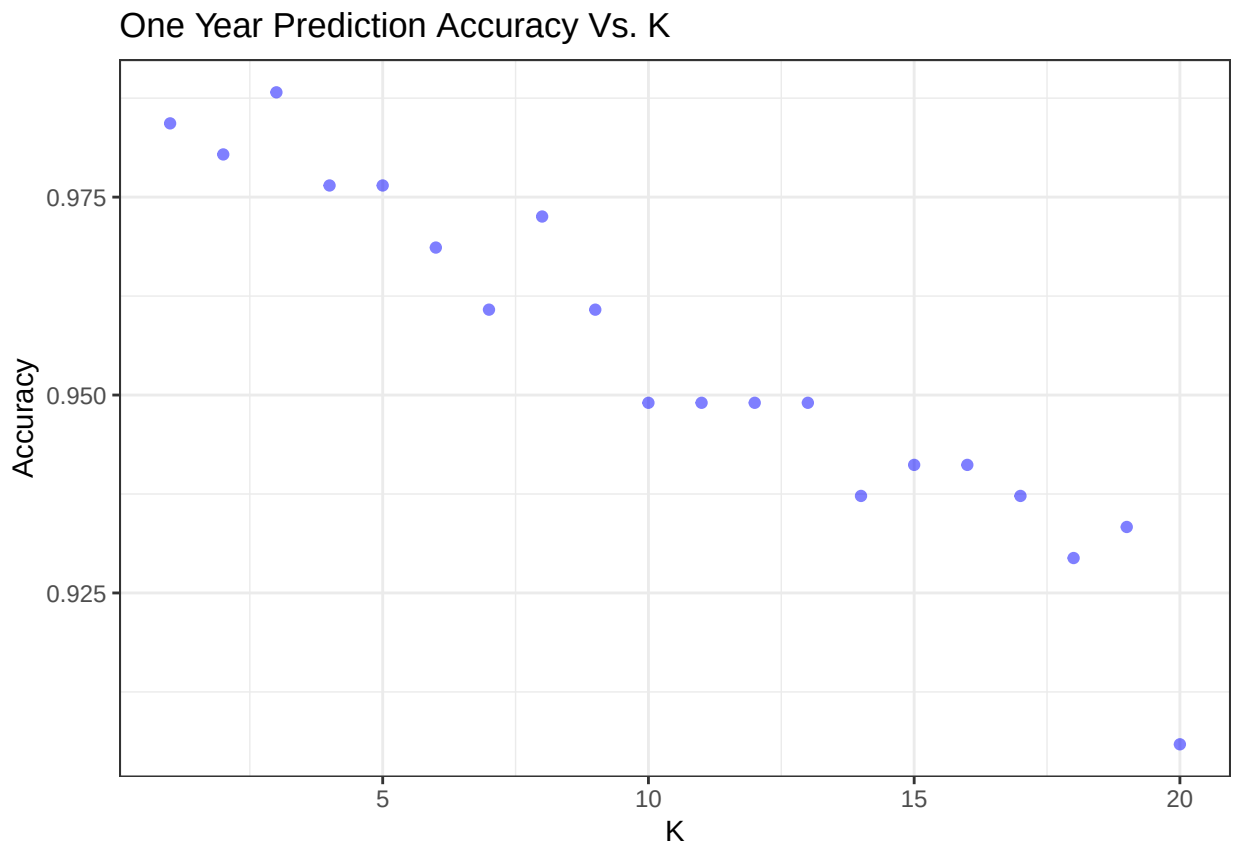
```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 184   2
##          1   3  66
##
##                Accuracy : 0.9804
##                  95% CI : (0.9548, 0.9936)
##     No Information Rate : 0.7333
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9501
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9840
##             Specificity : 0.9706
##          Pos Pred Value : 0.9892
##          Neg Pred Value : 0.9565
##              Prevalence : 0.7333
##          Detection Rate : 0.7216
##    Detection Prevalence : 0.7294
##       Balanced Accuracy : 0.9773
##
##        'Positive' Class : 0
##
```

Cross Validation: In order to gain the optimal prediction accuracy, we need to find what will be the optimal K (# of neighbors to be considered) in this case. In this case, we use number 1 to 20 for K and to see which test accuracy is the highest and choose the the K with highest prediction accuracy. In this case, we are choosing K = 4.

After tuning the parameter K to be 4, The KNN model gives a test (out-of-sample) accuracy of 98.0392%.

```
# Cross Validating K
set.seed(123)
correct_ratio <- rep(0,20)
for (i in 1:20) {
  one_year_knn_pred <- knn(train_x, test_x, train_oneyear_y, k = i)
  correct_ratio[i] <- mean(one_year_knn_pred == test_set[, "one_year_rec"])
}

correct_ratio <-  as.data.frame(t(as_data_frame(rbind(1:20,correct_ratio))))%>%
  set_colnames(c("k", "accuracy"))

ggplot(correct_ratio, aes(x = k, y = accuracy)) +
  geom_point(size = 1.5, alpha = 0.5, color = "blue") +
  theme_bw() +
  ggtitle("One Year Prediction Accuracy Vs. K") +
  xlab("K") +
  ylab("Accuracy")
```




```
## One and half year recession prediction
one_half_year_knn_pred <- knn(train_x, test_x, train_onehalfyear_y, k = 6)
table(one_half_year_knn_pred, test_set[, "one_half_year_rec"])
```

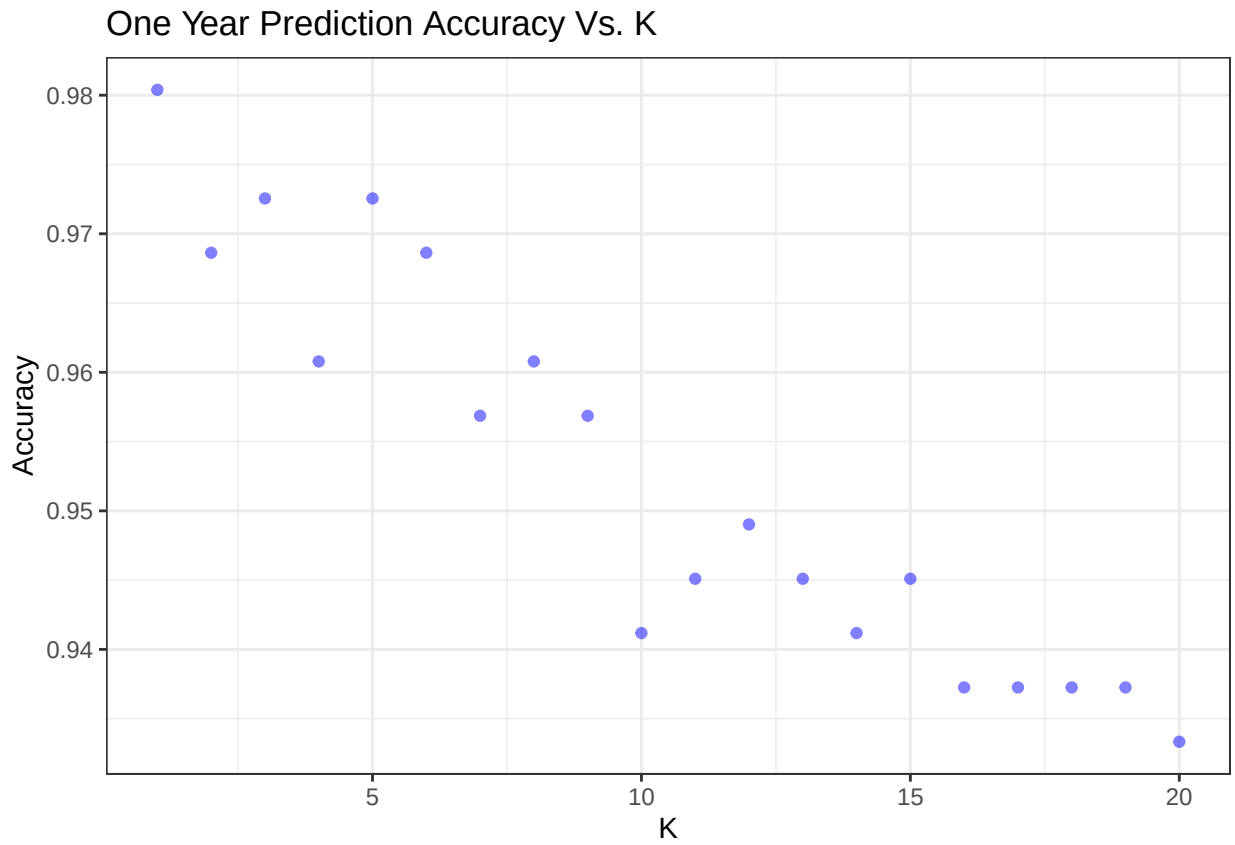
```
##
## one_half_year_knn_pred   0   1
##                      0 166   6
##                      1   5  78
```

```
mean(one_half_year_knn_pred == test_set[, "one_half_year_rec"])
```

```
## [1] 0.9568627
```

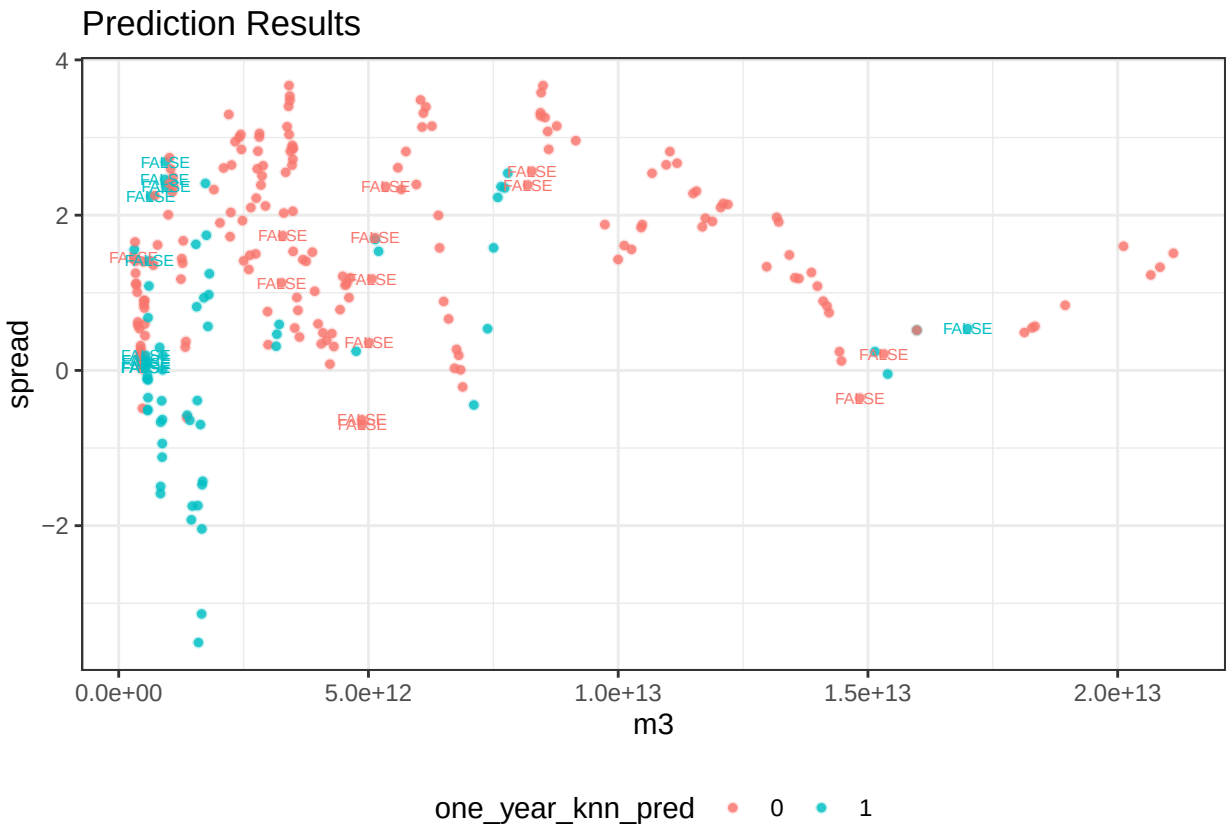
```
set.seed(123)
# Cross Validating K
correct_ratio <- rep(0,20)
for (i in 1:20) {
  one_half_year_knn_pred <- knn(train_x, test_x, train_onehalfyear_y, k = i)
  correct_ratio[i] <- mean(one_half_year_knn_pred == test_set[, "one_half_year_rec"])
}

correct_ratio <-  as.data.frame(t(as_data_frame(rbind(1:20,correct_ratio))))%>%
  set_colnames(c("k", "accuracy"))
```

```r
ggplot(correct_ratio, aes(x = k, y = accuracy)) +
  geom_point(size = 1.5, alpha = 0.5, color = "blue") +
  theme_bw() +
  ggtitle("One Year Prediction Accuracy Vs. K") +
  xlab("K") +
  ylab("Accuracy")
```

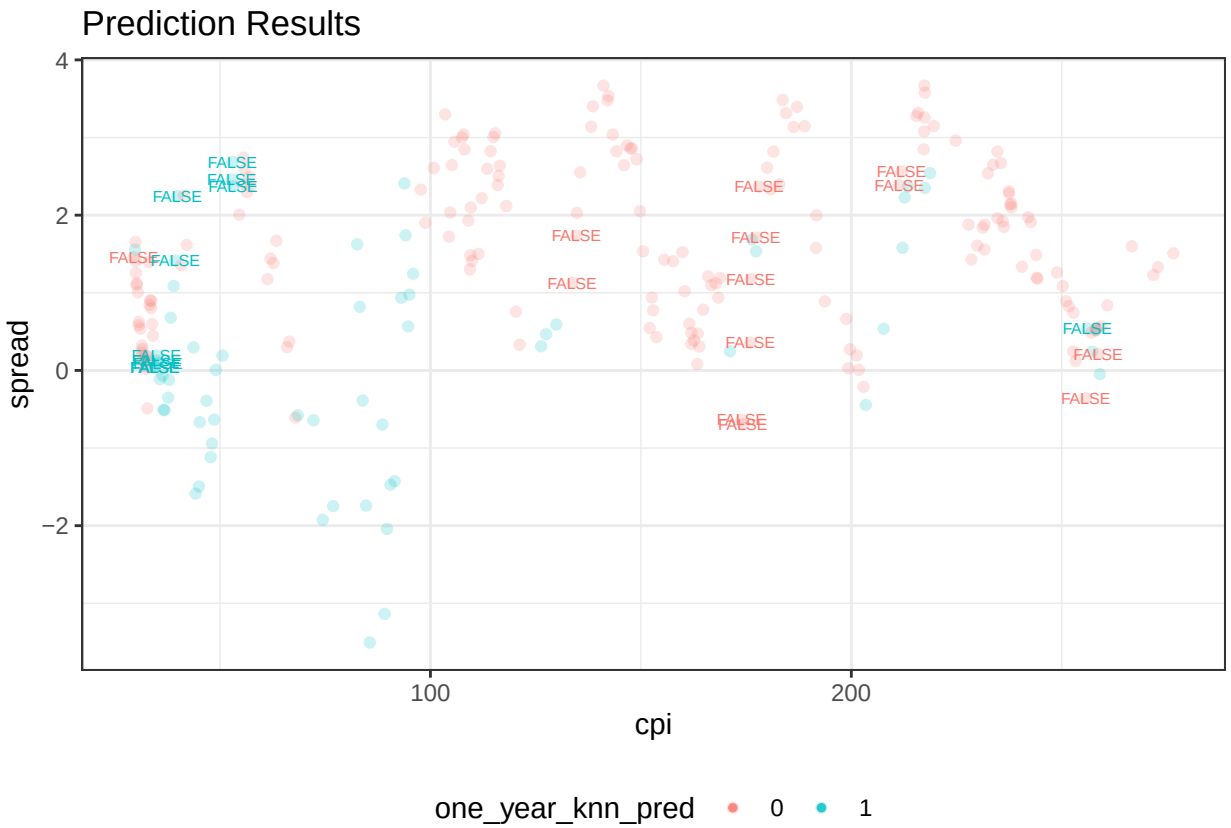


Cross-Validating K for One Year Recession Prediction: This part goes through 1 to 20 in K and compare the out-of-sample prediction accuracy to select optimal K.

```r
## Create KNN Graph
test_graph <- data.frame(test_set, one_year_knn_pred)
test_graph$correct <- test_graph$one_year_knn_pred == as.character(test_graph$one_year_rec)
test_graph$correct[test_graph$correct == "TRUE"] <- ""

ggplot(test_graph, aes(y = spread, x = m3, col = one_year_knn_pred, label = correct)) +
  geom_point(size = 1.5, alpha = 0.2) +
  geom_point(data = subset(test_graph, correct = FALSE),
             aes(y = spread, x = m3, col = one_year_knn_pred, label = correct),
             size = 1, alpha = 0.8)+
  theme_bw() +
  theme(legend.position = "bottom") +
  ggtitle("Prediction Results") +
  geom_text(size = 2, show.legend = FALSE)+
  xlim(min(test_graph$m3), max(test_graph$m3)) +
  ylim(min(test_graph$spread), max(test_graph$spread))
```




```
ggplot(test_graph, aes(y = spread, x = cpi, col = one_year_knn_pred, label = correct)) +
  geom_point(size = 1.5, alpha = 0.2) +
  geom_point(data = subset(test_graph, correct = FALSE),
             aes(y = spread, x = m3, col = one_year_knn_pred, label = correct),
             size = 1, alpha = 0.8)+
  theme_bw() +
  theme(legend.position = "bottom") +
  ggtitle("Prediction Results") +
  geom_text(size = 2, show.legend = FALSE)+
  xlim(min(test_graph$cpi), max(test_graph$cpi)) +
  ylim(min(test_graph$spread), max(test_graph$spread))
```



KNN Visualization: KNN graph is shown here in two dimension chart (might be more insight to see it in three and higher dimension)

---

6 Tree Method

```
tree_fit <- tree(as.factor(one_year_rec) ~ spread + sp500 + consumer_senti + unem_rate + fed_funds + new
tree_pred <- predict(tree_fit,test_set, type = "class")
table(tree_pred, test_set$one_year_rec)
```

```
##
## tree_pred   0   1
##         0 180  11
##         1   7  57
```

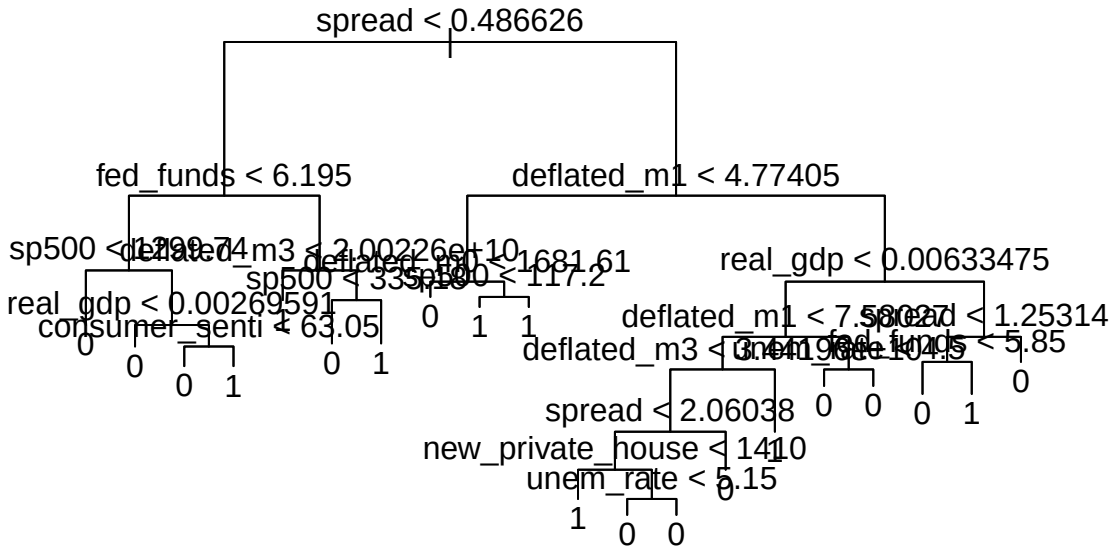
```
mean(tree_pred == test_set$one_year_rec)
```

```
## [1] 0.9294118
```

One and Half Year Recession Prediction Confusion Matrix as Above

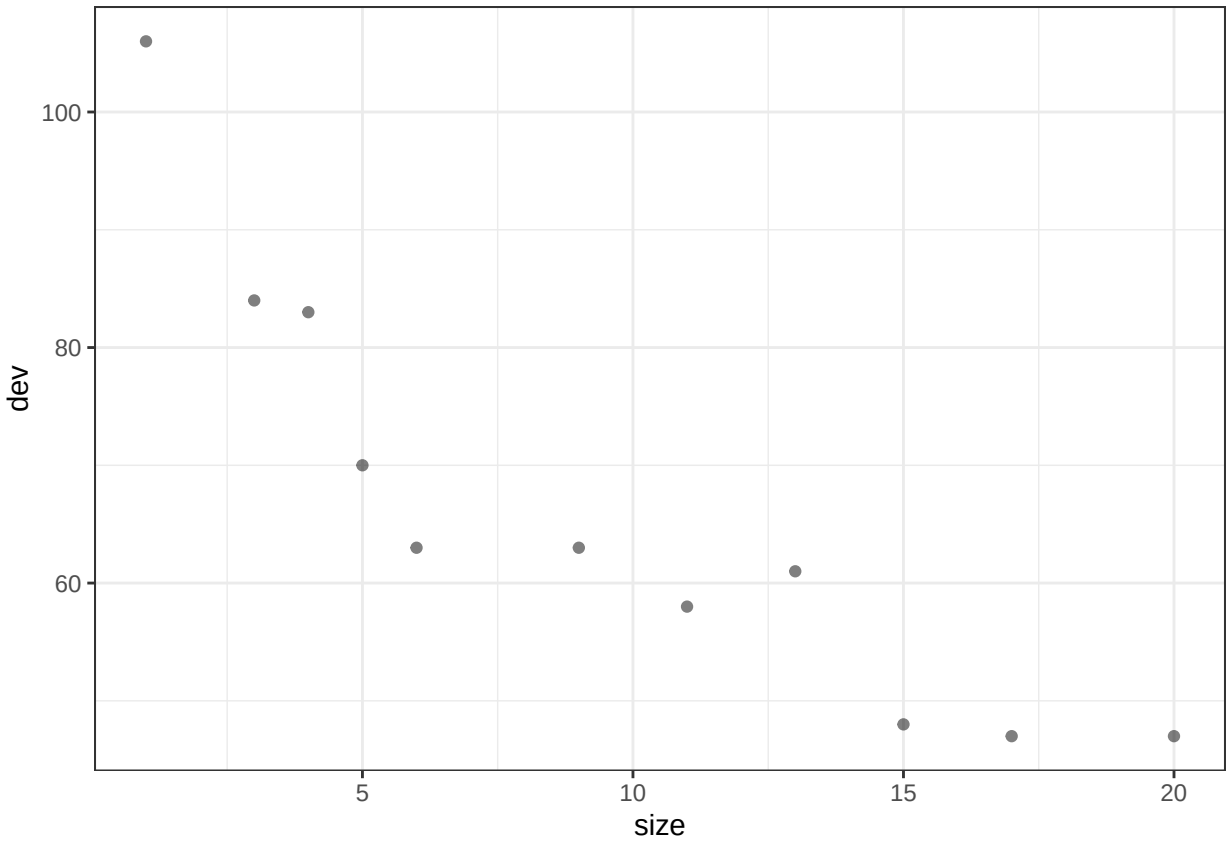
One and Half Year Recession Prediction Accuracy as Above

```
plot(tree_fit)
text(tree_fit, pretty = 0)
```



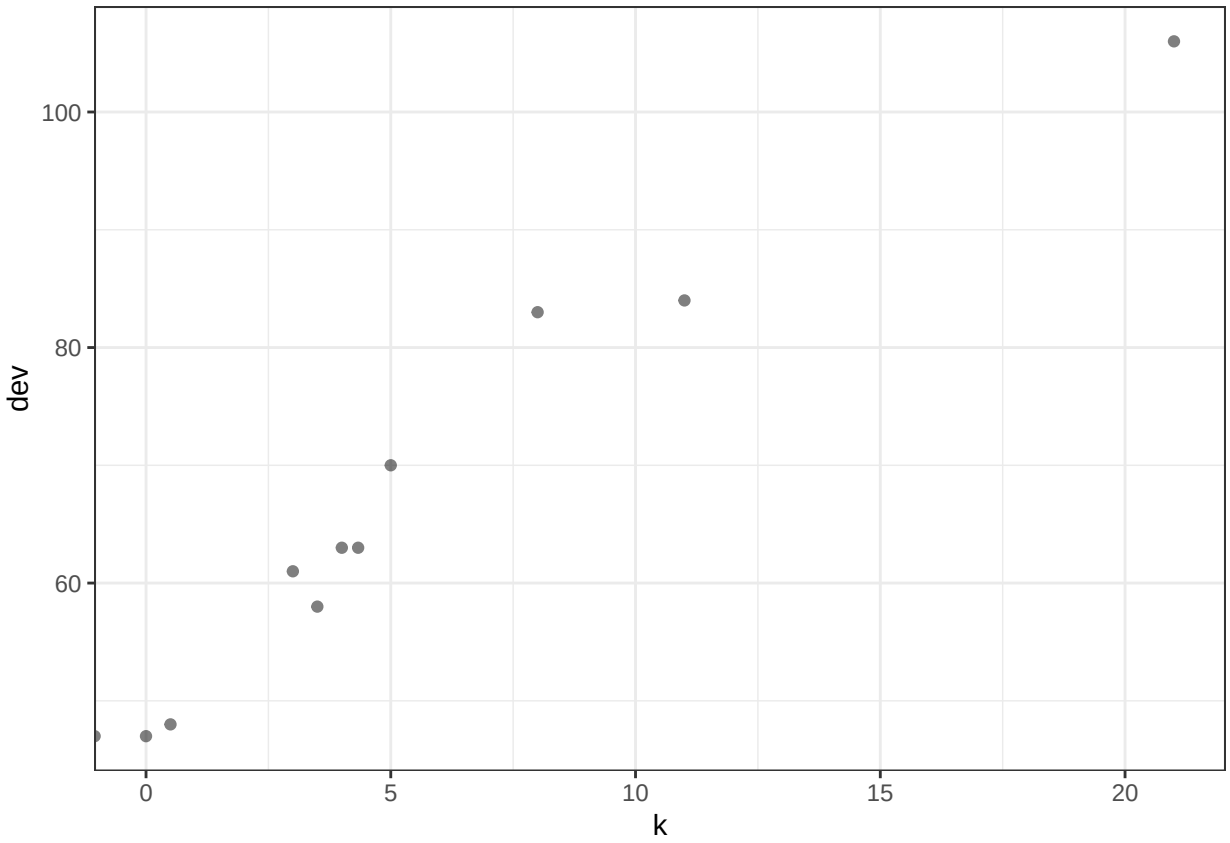
Tree Visualization: Visualization of the decision tree above.

```
# Conducting CV
set.seed(1)
cv_tree_fit <- cv.tree(tree_fit, FUN = prune.misclass)
cv_result <- data.frame(cv_tree_fit$size, cv_tree_fit$k, cv_tree_fit$dev)%>%
  set_colnames(c("size", "k", "dev"))

ggplot(cv_result, aes(x = size, y = dev)) +
  geom_point(size = 1.5, alpha = 0.5) +
  theme_bw()
```



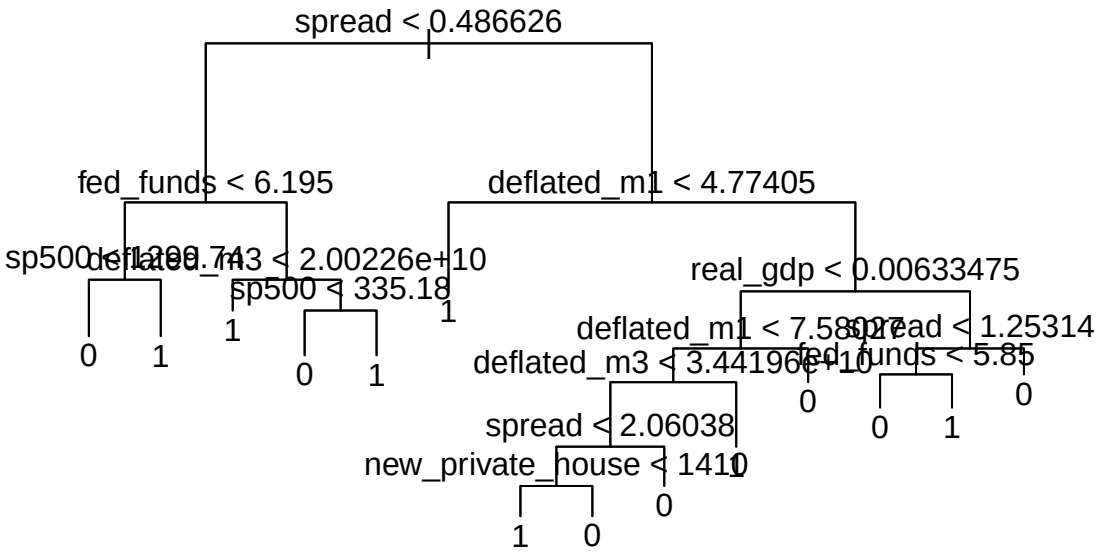
Cross-Validating for Optimal Size of the Tree: This graph go through the tree size against the error rate.

```
ggplot(cv_result, aes(x = k, y = dev)) +
  geom_point(size = 1.5, alpha = 0.5) +
  theme_bw()
```



Cross-Validating for Optimal alpha: This graph go through the alpha (tuning parameter for prune tree) against the error rate.

```
prune_tree <- prune.tree(tree_fit, best = 13)
plot(prune_tree)
text(prune_tree, pretty = 0)
```




```
prune_tree_pred <- predict(prune_tree, test_set, type = "class")
table(prune_tree_pred, test_set$one_year_rec)
```

```
##
## prune_tree_pred   0   1
##               0 178  12
##               1   9  56
```

```
mean(prune_tree_pred == test_set$one_year_rec)
```

```
## [1] 0.9176471
```

6.1 Random Forest

```
rf_fit <- randomForest(as.factor(one_year_rec) ~ spread + sp500 + consumer_senti + unem_rate + fed_funds
rf_pred <- predict(rf_fit, newdata = test_set)
table(rf_pred, test_set$one_year_rec)
```

```
##
## rf_pred   0   1
##       0 185   8
##       1   2  60
```

```r
mean(rf_pred == test_set$one_year_rec)
```

```
## [1] 0.9607843
```

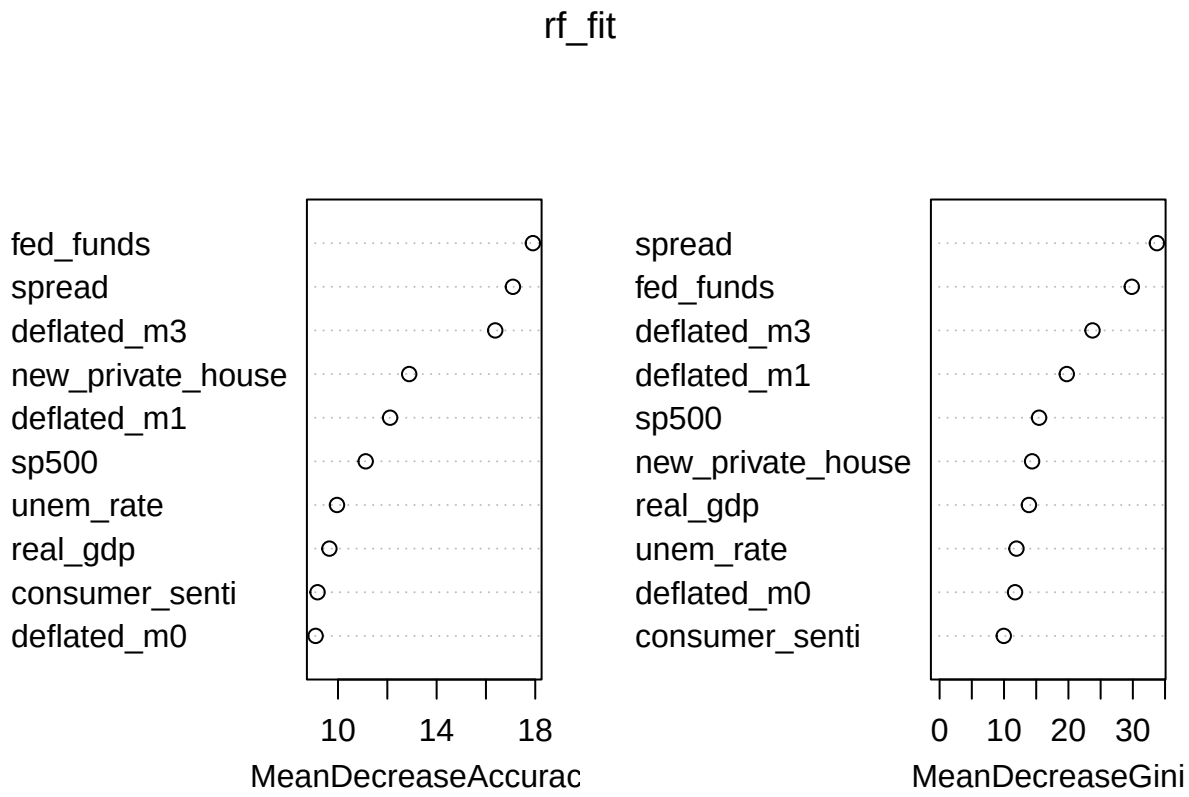
```r
rf_fit <- randomForest(as.factor(one_year_rec) ~ spread + sp500 + consumer_senti + unem_rate + fed_funds
rf_pred <- predict(rf_fit, newdata = test_set)
table(rf_pred, test_set$one_year_rec)
```

```
##
## rf_pred   0   1
##       0 186   3
##       1   1  65
```

```r
mean(rf_pred == test_set$one_year_rec)
```

```
## [1] 0.9843137
```

```r
varImpPlot(rf_fit)
```



____ ## Boosting

```r
set.seed(123)
boost_fit <- gbm(as.character(one_year_rec) ~ spread + sp500 + consumer_senti + unem_rate + fed_funds +
boost_fit
```

```
## gbm(formula = as.character(one_year_rec) ~ spread + sp500 + consumer_senti +
##     unem_rate + fed_funds + new_private_house + deflated_m0 +
##     deflated_m3 + real_gdp + deflated_m1, distribution = "bernoulli",
##     data = training_set, n.trees = 1000, interaction.depth = 4)
## A gradient boosted model with bernoulli loss function.
## 1000 iterations were performed.
## There were 10 predictors of which 10 had non-zero influence.
```

```
boost_pred <- predict(boost_fit, newdata = test_set, n.trees = 1000)
```
