**Machine Learning the Value of Gold Final Report**

Russell Chan

5/18/22

Math448

**Table of Contents**

* Abstract
* Data Entry
* Ordinary Least Squares Regression
* Ridge Regression
* Lasso Regression
* Principal Component Regression (PCR)
* Partial Least Squares Regression (PLS)
* Conclusion
* Appendix A: R Code

**Abstract**

Gold prices are at an all-time high and we want to know how it will increase or decrease. The rate of gold is volatile, and we want to learn its behavior. Our main objective is to predict whether the price of gold will go down, go up, or stay the same. We want to know when the value of gold will go down to buy, and when the value will go up to sell.

Our data starts in 1985 and provides a dollar amount that gold was sold per ounce each day. The entire dataset has 9714 observations of the daily recorded value of gold per ounce. To make the data usable for prediction purposes, lag variables were created for the gold price, and the month and year were extracted from the date. This project will attempt to find the best prediction model to predict the actual value of gold based on the date and lag variables.

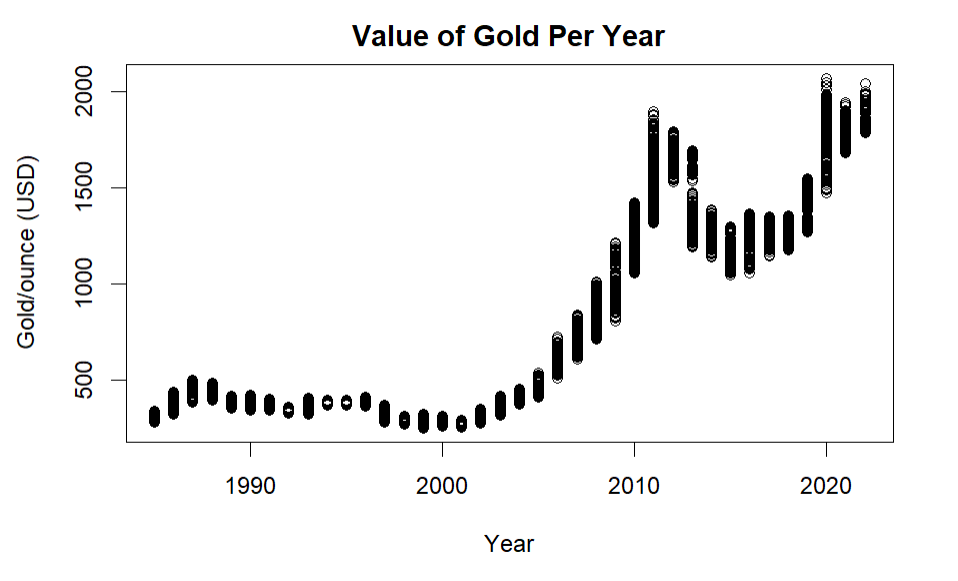
The methods that will predict and analyze our data are ordinary least squares linear regression, ridge regression, lasso regression, principal component regression (PCR), and partial least squares (PLS) regression. For each one we will find each test's mean squared error and compare the errors of the methods. The one with the lowest MSE will prove to be the best prediction method for the gold rate data.

The best method to use for our data set is PLS, but with some limitations. Though it had the best MSE, it was not by a lot compared to others, and it still had a considerable amount of error. Moving forward we will continue to implement more predictive models to our data in pursuit of the best predictive method.

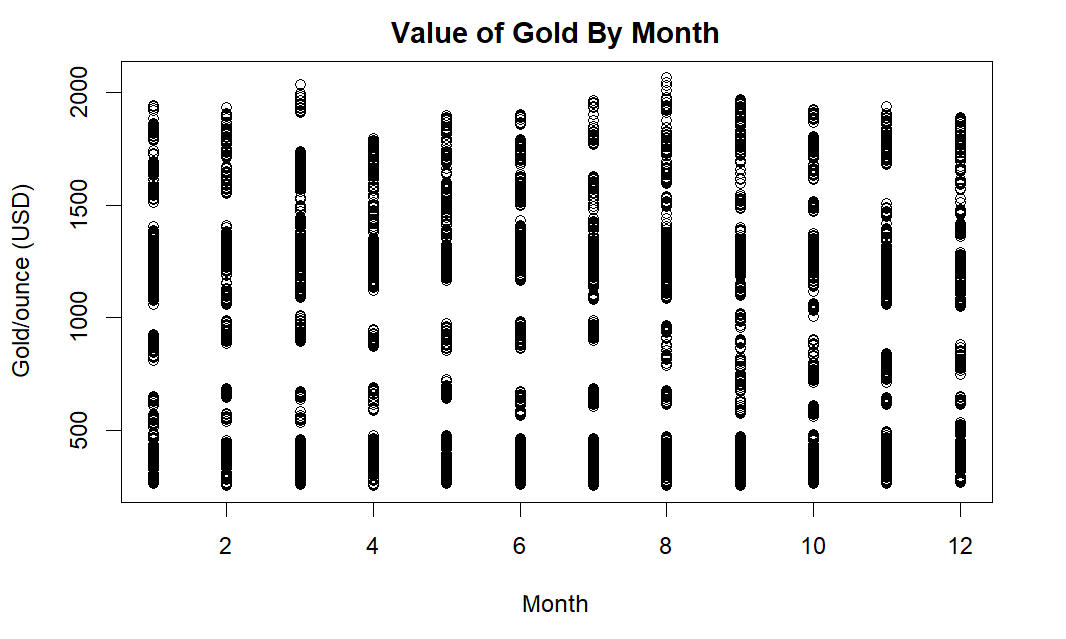
Dataset: <https://www.kaggle.com/hemil26/gold-rates-1985-jan-2022>

**Data Entry**

The variables including year, month, and actual rate, are entered into a data frame. I will add lag predictors to the data to see if the previous values control trends of the current value. Doing this will make the first observation in the data the fifth lag, and the sixth observation in the data the first actual value. The dependent variable we are predicting in the models is the actual rate, and the independent variables are the remaining variables. The month variable is changed to a categorical variable to show if there are trends in certain months. The year variable is changed to a numeric variable to see the pattern of the rates through the years. The data is split into training and validation sets, with the training set being 80% of the data, and the validation set is the remaining 20%. This will help us test the accuracy of the prediction methods.

**Data Visualization**

The plot between year and gold value shows a positive relationship. Between 1985 and 2000, there is insignificant change in price. As the years move forward, the value of gold increases with some exceptions. From 2000 to 2011, gold prices increase. It peaks around 2011, dips, and then increases once again around 2018 until now. The plot shows that the year has a clear relationship with the value of gold.



The relationship between months and gold prices is inconclusive. The value of gold shows a similar range for every month. This means that gold prices are not controlled by the month, and we can’t say the month contributes to the value of gold. Months will still be included to predict because our data initially doesn’t have many predictors and we want as much information as possible.

**Ordinary Least Squares Regression**

Ordinary least squares regression, also known as linear regression, is the prediction model that uses a linear model where  are the k explanatory variables and Y is a dependent variable. The linear model:  predicts Y based on the variables and the coefficients with the lowest error. The least squares in the name refers to the minimum square error which is needed to produce the best coefficients with the least error. The coefficients obtained from OLS that have significance in predicting the actual gold rate are the lag4, lag5, and year variables. The test MSE found through the OLS method is 79.00529.

**Ridge Regression**

            Ridge regression is like OLS, but instead of estimating by minimizing the residual sum of squares, it adds a shrinkage penalty and minimizes the where is a tuning parameter to estimate , known as a ridge estimator. λ is found in ridge regression from the training and testing sets from data entry. Cross validation is used to choose a λ that returns the minimum error rate. The size of λ controls the relative impact of the RSS and shrinkage penalty; if λ is large, the shrinkage penalty will be the dominant criterion, if λ is small, the model will behave like OLS, where RSS is the dominant criterion. Ridge regression shrinks all coefficients towards 0 to find the lowest test MSE. The best lambda found is 51.16, and the test MSE obtained from ridge regression is 431.408 which is very high.

**Lasso Regression**

Lasso regression works similarly like OLS and ridge regression, except it introduces a different shrinkage penalty known as . The lasso estimates by minimizing where remains a tuning parameter. λ is found the same way as it is found in ridge; like ridge regression, the lasso model will be heavily influenced by the size of λ; a large λ will lead to the shrinkage penalty being the dominant criterion, and a small λ will result in a model close to the normal OLS regression. The lasso penalty has the tendency to force some coefficients to be zero when the λ is sufficiently large. This provides a sparse model, which only includes necessary estimators and is easier to interpret. The best lambda found is 12.3833, and the test MSE calculated is 239.5951 which is high, but not as high as the ridge error.

**Principal Component Regression (PCR)**

Principal component regression involves transforming predictors and then fitting a least square model with the transformed predictors.  First, we perform a principal component analysis on the x variables to reduce the dimensionality of the data. This is done by selecting several principal components by using cross validation until it reaches the lowest error. In my data, the number of components used were 17. Dimensionality remains the same because we originally had 17 predictors. We should expect this model to behave like linear regression. The test MSE produced is 79.00529 which is the exact same as linear regression.

**Partial Least Squares Regression (PLS)**

            Partial least squares regression works the same way PCR works, except one major difference. In PCR, the dependent variable Y does not have any effect on the principal component direction. In PLS, the component direction is affected by both the dependent variable Y and independent variables. PLS is a better predictor because it considers both dependent and independent variables when reducing dimension, whereas PCR only considers dependent variables when identifying components. PLS functions like PCR by selecting components until it reaches one that has a satisfactory error rate. The PLS component with the lowest error was 12 components, and the test MSE from the PLS prediction is 78.88178 which is the lowest MSE of all the prediction methods. This is to be expected as PLS is using a smaller number of predictors to produce a model with less error.

**Conclusion**

|  |  |  |
| --- | --- | --- |
| Method | Test MSE | RMSE |
| Linear Regression | 79.00529 | 8.8884 |
| Ridge Regression | 430.4718 | 20.7478 |
| Lasso Regression | 239.5952 | 15.4789 |
| Principal Component Regression | 79.00529 | 8.8884 |
| Partial Least Squares Regression | 78.88178 | 8.881 |

The predictive model with the lowest error is PLS. Linear regression and PCR have very similar but slightly higher error rates. The worst fitting models are ridge and LASSO regression. After considering the five model results, the best prediction model found is PLS, but not by much. It’s test MSE is 78.88178 and its RMSE is 8.881 which means on average the prediction model is off by $8.88. This error rate could be too high for practical purposes, so implementing more predictive models to find a lower error is necessary for future work. The predictors in the data are somewhat limited as they only consist of the date and lag predictors. Integrating other data sets that occur around the same time period could be beneficial to find important predictor variables that contribute to the value of gold.

**Appendix A: R Code**

#data entry

setwd("C:/Users/macho/Desktop/sfsu/spr22/math448")

gold=read.csv("daily\_gold\_rate.csv",header=T,na.strings="?")

library(glmnet)

attach(gold)

gold$year=substr(gold$Date,1,4)

gold$month=substr(gold$Date,6,7)

set=c("USD","year","month")

data=gold[,set]

attach(data)

actual=data$USD[6:nrow(data)]

month=data$month[6:nrow(data)]

year=data$year[6:nrow(data)]

lag1=data$USD[5:(nrow(data)-1)]

lag2=data$USD[4:(nrow(data)-2)]

lag3=data$USD[3:(nrow(data)-3)]

lag4=data$USD[2:(nrow(data)-4)]

lag5=data$USD[1:(nrow(data)-5)]

df = data.frame(actual, lag1, lag2, lag3, lag4, lag5, month, year)

df$month=as.factor(as.character(df$month))

df$year=as.numeric(as.character(df$year))

#train and validation setup

x=model.matrix(actual~.,df)[,-1]

y=df$actual

set.seed(1)

train=sample(1:nrow(df),0.8\*nrow(df))

df.train=df[train,]

df.test=df[-train,]

x.train=x[train,]

x.test=x[-train,]

y.train=y[train]

y.test=y[-train]

#Linear Regression

ols = lm(actual~., data = df.train)# all predictors

ols = lm(actual~lag1+ lag4+ lag5+ month +year, data = df.train) #significant predictors

summary(ols)

y.pred=predict(ols,newdata=df.test)

mean((y.test-y.pred)^2) #test MSE

#Ridge Regression

grid=10^seq(10,-2,length=100)

ridge.mod=glmnet(x.train,y.train,alpha=0,lambda=grid, thresh=1e-12)

cv.out=cv.glmnet(x.train,y.train,alpha=0)

plot(cv.out)

bestlam=cv.out$lambda.min

bestlam

ridge.pred=predict(ridge.mod,s=bestlam,newx=x.test)

mean((ridge.pred-y.test)^2) #test MSE

#Lasso Regression

lasso.mod=glmnet(x.train,y.train,alpha=1,lambda=grid)

cv.out=cv.glmnet(x.train,y.train,alpha=1)

plot(cv.out)#CV error plot

bestlam=cv.out$lambda.min

bestlam

lasso.pred=predict(lasso.mod,s=bestlam,newx=x.test)

mean((lasso.pred-y.test)^2) #test MSE

#PCR

library(pls)

pcr.fit=pcr(actual~., data=df.train,scale=TRUE,validation="CV")

summary(pcr.fit)

validationplot(pcr.fit,val.type="MSEP") #MSE plotted

pcr.pred=predict(pcr.fit,df.test,ncomp=17)

mean((pcr.pred-y.test)^2) #test MSE

#PLS

pls.fit=plsr(actual~., data=df.train,scale=TRUE, validation="CV")

summary(pls.fit)

validationplot(pls.fit,val.type="MSEP") #plot MSE

pls.pred=predict(pls.fit,df.test,ncomp=12)

mean((pls.pred-y.test)^2) #test MSE