Inflation Prediction Model Using Machine Learning

Animesh Kumar Jha Sarthak Chauhan Abhiraj Rana

Introduction

The inflation rate is the rate at which the price of services and commodities increases. The decline in purchasing power over time can cause issues if individuals and businesses do not prepare for the changes. The way inflation is calculated is by estimating the difference in the cost of a basket of selected goods or necessities over time and the percentage change in its value is the inflation percentage. Our model is an Inflation prediction system in which we have compared Machine Learning Models such as Linear Regression, Lasso Regression, Random Forest, XGBoost, and Ridge Regression.

We surveyed three research papers on this topic; [1] the research done is a study on inflation forecast values of numerous Machine Learning models in contrast to newer models such as Random Forest and Quantile Regression. The variables from the dataset are heavily scrutinized and chosen on their predictive values using an out-of-sample empirical analysis to provide forecasts for up to a year. In the paper [2] a search for understanding the link between inflation rates and the growth of inequality, all the variables of the dataset are used to create empirical conclusions. Using deep learning and using historical data the major elements of governing inflation are found along with the predicted inflation rate in the future. Lastly, a comparison of the regularization model and base models is provided where in comparison the base models showed a better output. According to this paper random forest and boosting perform better on unprocessed data[3].

The knowledge gap that we were faced with was that there are numerous Machine Learning Models that have been used to attempt for predicting inflation rates, every algorithm has its pros and cons, and choosing the best algorithm to find the most optimal prediction of inflation is the goal for this project. The questions that we trying to answer are, the most important variables in the dataset chosen that are most suitable for the prediction of inflation values and the algorithm that provides the best performance.

Methods and Materials

The dataset that we chose for this model has been chosen from Kaggle, named - “US Macro-Economic Factors data from 2002-2022”. We chose this dataset as it contains values such as the Unemployment Rate, Consumer Confidence Index, Producer's Purchase Index, Consumer Price Index, Average Mortgage interest rate, Median Household Income, percentage

share of the working population, GDP per capita, and the inflation rate of the particular month for the last twenty years. In Economic theory, these values are the most important factors in inflation prediction. Kaggle link - "https://www.kaggle.com/datasets/sagarvarandekar/macroeconomic-factors-affecting-us-housing-prices".

After loading the dataset, exploratory data analysis was done on the data by skimming it. The numerical values of the dataset were checked and the columns with missing values were found. The data was split into 70-30 percent training and testing sets and then, the KNN imputer function was used to impute the mean values of the columns in place of the missing values in the dataset. We split the dataset before imputing it in order to prevent data leakage. Then we skim the dataset again to find out if any issues are still present in the data. After the dataset is cleaned we use the scale function to scale the dataset values to be used in the final Machine Learning algorithms.

Results

Once the dataset was cleaned and scaled, We used the Forward Selection method to select the most important variables in the dataset that are crucial in predicting the rate of inflation. The variables that were found to be most optimal while inflation prediction is - Producers' Purchase Index, Consumer Price Index, Average Mortgage interest rate, Corporate bond yield, median household income, S&P/Case-Shiller U.S. National Home Price Index given by FRED, monthly home supply, and share of working population.

The values of the train and test set are divided into x and x_test, and y and y_test. The training sets are then used in the cross-validation method by which the optimal lambda values for different Regression models can be found. After this we use the data in Machine Learning models:-
1)Linear Regression –

{r}

```
#Linear Regression Model

set.seed(1902)

linearModel = lm(INFLATION... ~ PPI.CONST.MAT. + CPIALLITEMS+
MORTGAGE.INT..MONTHLY.AVG... + CORP..BOND.YIELD... + MED.HOUSEHOLD.INCOME +
CSUSHPISA + MONTHLY.HOME.SUPPLY + X..SHARE.OF.WORKING.POPULATION, data = train)

summary(linearModel)

linearPred <- predict(linearModel, test)

lm_rmse <- rmse(test$INFLATION..., linearPred) #0.5718129

AIC(linearModel)

BIC(linearModel)
```

```
Call:
lm(formula = INFLATION... ~ PPI.CONST.MAT. + CPIALLITEMS + MORTGAGE.INT..MONTHLY.AVG... +
    CORP..BOND.YIELD... + MED.HOUSEHOLD.INCOME + CSUSHPISA +
    MONTHLY.HOME.SUPPLY + X..SHARE.OF.WORKING.POPULATION, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.38890 -0.29445  0.02902  0.34231  1.30279

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                   -3.733e-15  4.834e-02   0.000 1.000000
PPI.CONST.MAT.                 1.715e+00  1.756e-01   9.769  < 2e-16 ***
CPIALLITEMS                    4.643e-01  3.837e-01   1.210 0.228018
MORTGAGE.INT..MONTHLY.AVG...   1.203e+00  2.084e-01   5.772 3.99e-08 ***
CORP..BOND.YIELD...           -8.827e-01  2.404e-01  -3.671 0.000329 ***
MED.HOUSEHOLD.INCOME          -7.357e-01  3.267e-01  -2.252 0.025673 *
CSUSHPISA                      6.262e-01  2.170e-01   2.886 0.004450 **
MONTHLY.HOME.SUPPLY           -2.471e-01  9.888e-02  -2.499 0.013456 *
X..SHARE.OF.WORKING.POPULATION  6.712e-01  3.127e-01   2.147 0.033346 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6265 on 159 degrees of freedom
Multiple R-squared:  0.6263,    Adjusted R-squared:  0.6075
F-statistic:  33.3 on 8 and 159 DF,  p-value: < 2.2e-16

[1] 330.4117
[1] 361.6513
```

Fig. Output for Linear Regression

2)Ridge Regression –

```
{r}

#Ridge Regression Model


set.seed(1902)

#define response variable

y <- train$INFLATION...

#define matrix of predictor variables

x <- data.matrix(train[, c('PPI.CONST.MAT.','CPIALLITEMS', 'MORTGAGE.INT..MONTHLY.AVG...',
'CORP..BOND.YIELD...','MED.HOUSEHOLD.INCOME', 'CSUSHPISA','MONTHLY.HOME.SUPPLY',
'X..SHARE.OF.WORKING.POPULATION')])

xtest <- data.matrix(test[, c('PPI.CONST.MAT.','CPIALLITEMS', 'MORTGAGE.INT..MONTHLY.AVG...',
'CORP..BOND.YIELD...','MED.HOUSEHOLD.INCOME', 'CSUSHPISA','MONTHLY.HOME.SUPPLY',
'X..SHARE.OF.WORKING.POPULATION')])

ytest <- test$INFLATION...

#perform k-fold cross-validation to find optimal lambda value

crossModelRidge <- cv.glmnet(x, y, alpha = 0)

lambdas <- 10^seq(2, -3, by = -.1)

ridgeRegression <- glmnet(x, y, alpha = 0,family = 'gaussian', lambda = lambdas)

summary(ridgeRegression)

#find optimal lambda value that minimizes test MSE

bestLambda <- crossModelRidge$lambda.min

bestLambda

#produce plot of test MSE by lambda value

plot(crossModelRidge)

#find coefficients of best model

bestModel <- glmnet(x, y, alpha = 0, lambda = bestLambda)

coef(bestModel)

plot(ridgeRegression, xvar = "lambda")

#use fitted best model to make predictions

yPredicted <- predict(ridgeRegression, s = bestLambda, newx = xtest)

ridge_rmse <-rmse(ytest,yPredicted) #0.4914431
```

```
           Length Class     Mode
a0          51    -none-    numeric
beta       408    dgCMatrix S4
df          51    -none-    numeric
dim          2    -none-    numeric
lambda      51    -none-    numeric
dev.ratio   51    -none-    numeric
nulldev      1    -none-    numeric
npasses      1    -none-    numeric
jerr         1    -none-    numeric
offset       1    -none-    logical
call         6    -none-    call
nobs         1    -none-    numeric
[1] 0.03139312
9 x 1 sparse Matrix of class "dgCMatrix"
                                              s0
(Intercept)                       -2.013068e-15
PPI.CONST.MAT.                     1.350516e+00
CPIALLITEMS                        6.493612e-02
MORTGAGE.INT..MONTHLY.AVG...       8.674415e-01
CORP..BOND.YIELD...               -3.103988e-01
MED.HOUSEHOLD.INCOME              -3.238406e-01
CSUSHPISA                          2.282555e-01
MONTHLY.HOME.SUPPLY               -2.236387e-01
X..SHARE.OF.WORKING.POPULATION     3.424238e-01
```

Fig. Output for Ridge Regression

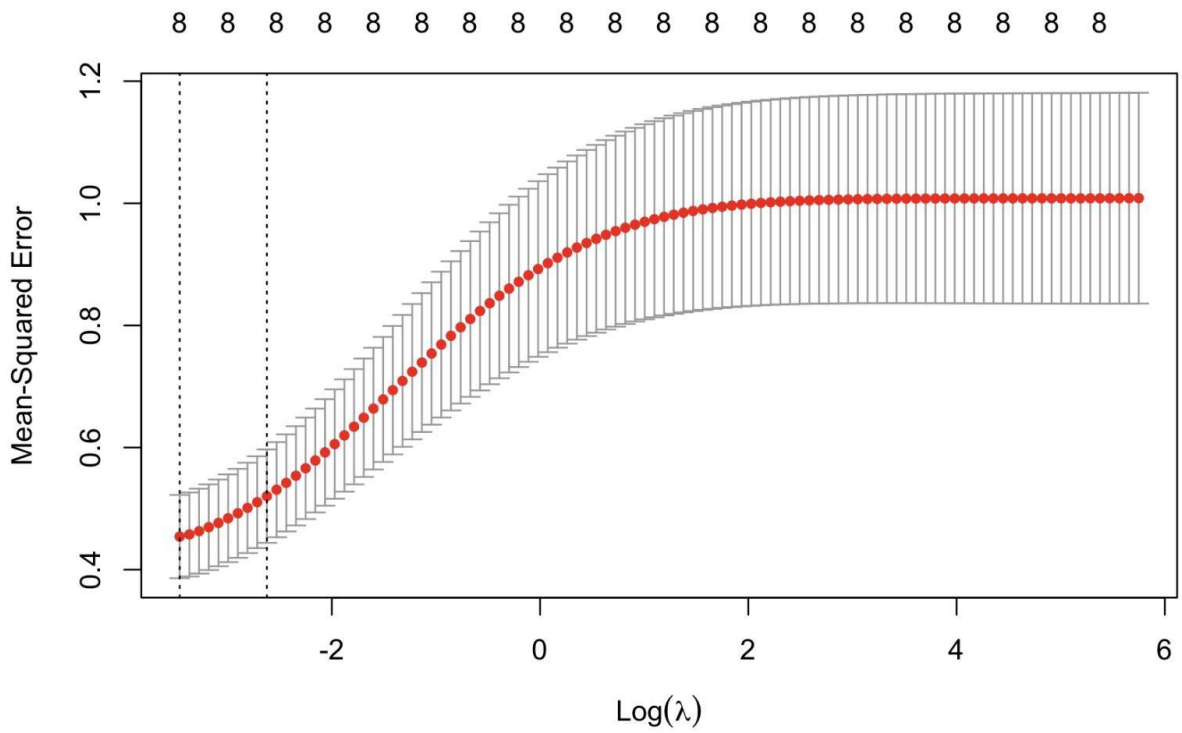

Fig. The graph of MSE vs Log(lambda)

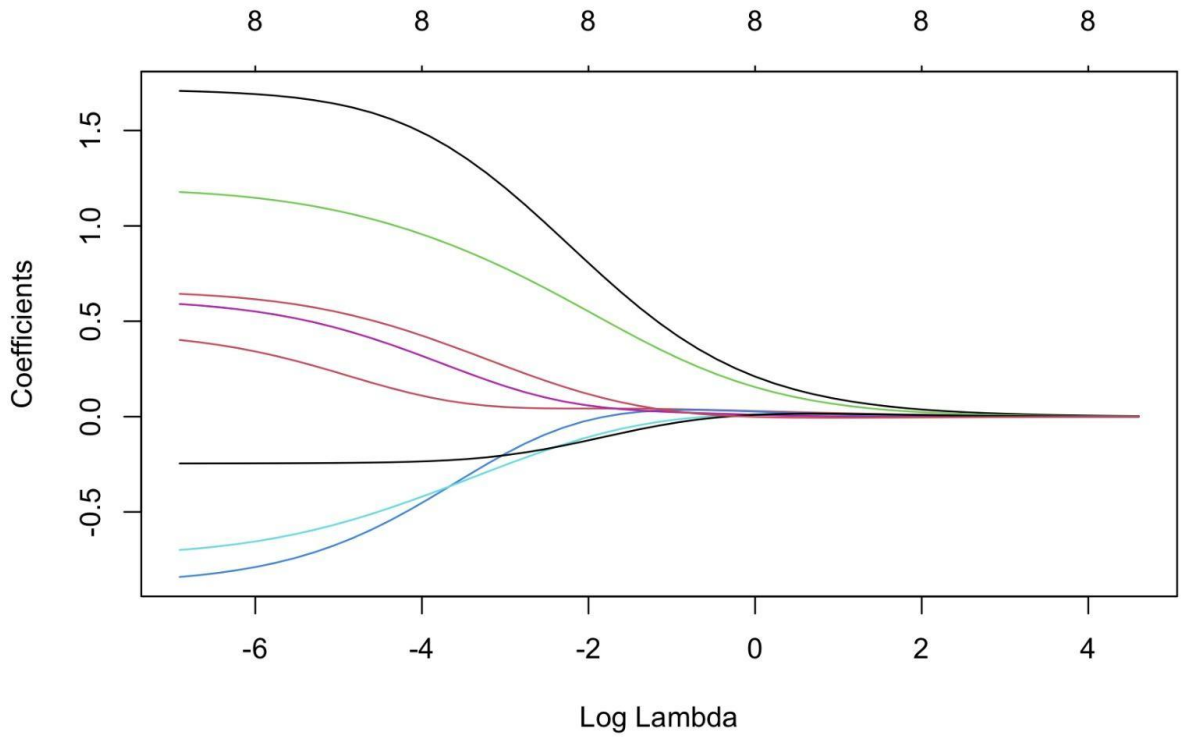

Fig. Graph of Coefficients vs Log(lambda)

3)Lasso Regression –

{r}

```
#Lasso Regression Model

set.seed(1902)

#perform k-fold cross-validation to find optimal lambda value

crossModel2 <- cv.glmnet(x, y, alpha = 1)


#find optimal lambda value that minimizes test MSE

bestLambda2 <- crossModel2$lambda.min

bestLambda2

#produce plot of test MSE by lambda value

plot(crossModel2)

#find coefficients of best model

bestModel2 <- glmnet(x, y, alpha = 1, lambda = bestLambda2)

coef(bestModel2)

#use fitted best model to make predictions

yPredicted2 <- predict(bestModel2, s = bestLambda2, newx = xtest)

lasso_rmse <-rmse(ytest,yPredicted2) #0.5697695
```

```
[1] 0.0002430655
9 x 1 sparse Matrix of class "dgCMatrix"
                                                 s0
(Intercept)                          -3.973851e-15
PPI.CONST.MAT.                        1.712321e+00
CPIALLITEMS                           4.500882e-01
MORTGAGE.INT..MONTHLY.AVG...          1.199470e+00
CORP..BOND.YIELD...                  -8.769051e-01
MED.HOUSEHOLD.INCOME                 -7.299107e-01
CSUSHPISA                             6.174392e-01
MONTHLY.HOME.SUPPLY                  -2.455779e-01
X..SHARE.OF.WORKING.POPULATION        6.626533e-01
```

Fig. The output for Lasso Regression

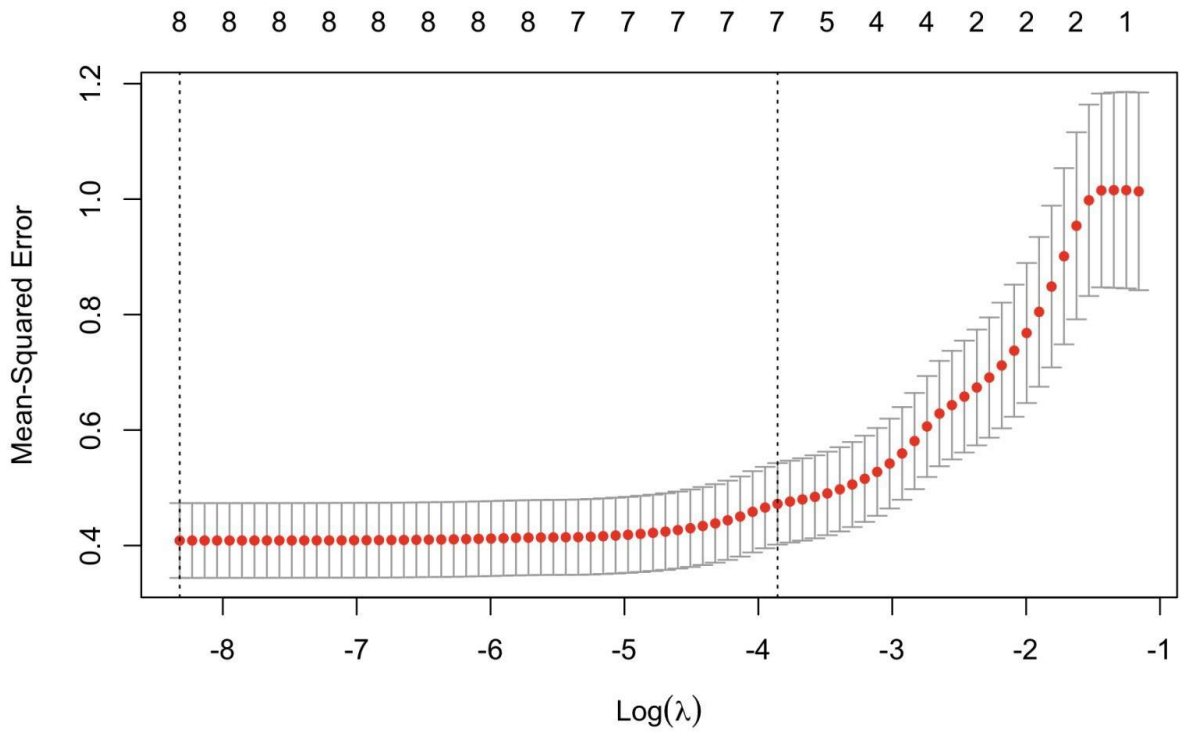

Fig. The graph of MSE vs Log(lambda)

4)Random Forest –

{r}

```
#Random Forest Model

set.seed(1902)

 randomForestModel <- randomForest(INFLATION... ~ PPI.CONST.MAT. + CPIALLITEMS + MORTGAGE.INT..MONTHLY.AVG... + CORP..BOND.YIELD... + MED.HOUSEHOLD.INCOME + CSUSHPISA + MONTHLY.HOME.SUPPLY + X..SHARE.OF.WORKING.POPULATION, data = train, mtry = 3, importance = TRUE, na.action = na.omit)

randomForestModel

plot(randomForestModel)

summary(randomForestModel)

rfPredictions = predict(randomForestModel, test)

mean((rfPredictions - test$INFLATION...)^2)

rf_rmse<-rmse(test$INFLATION..., rfPredictions)#0.6759944
```

```
 Call:
  randomForest(formula = INFLATION... ~ PPI.CONST.MAT. + CPIALLITEMS +
 MORTGAGE.INT..MONTHLY.AVG... + CORP..BOND.YIELD... + MED.HOUSEHOLD.INCOME +      CSUSHPISA +
 MONTHLY.HOME.SUPPLY + X..SHARE.OF.WORKING.POPULATION,      data = train, mtry = 3, importance =
 TRUE, na.action = na.omit)
               Type of random forest: regression
                     Number of trees: 500
 No. of variables tried at each split: 3
```

```
          Mean of squared residuals: 0.1414103
                    % Var explained: 85.77
                Length Class  Mode
call              6    -none- call
type              1    -none- character
predicted       168    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times       168    -none- numeric
importance       16    -none- numeric
importanceSD      8    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y               168    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
[1] 0.4569684
```

Fig. Output for Random Forest

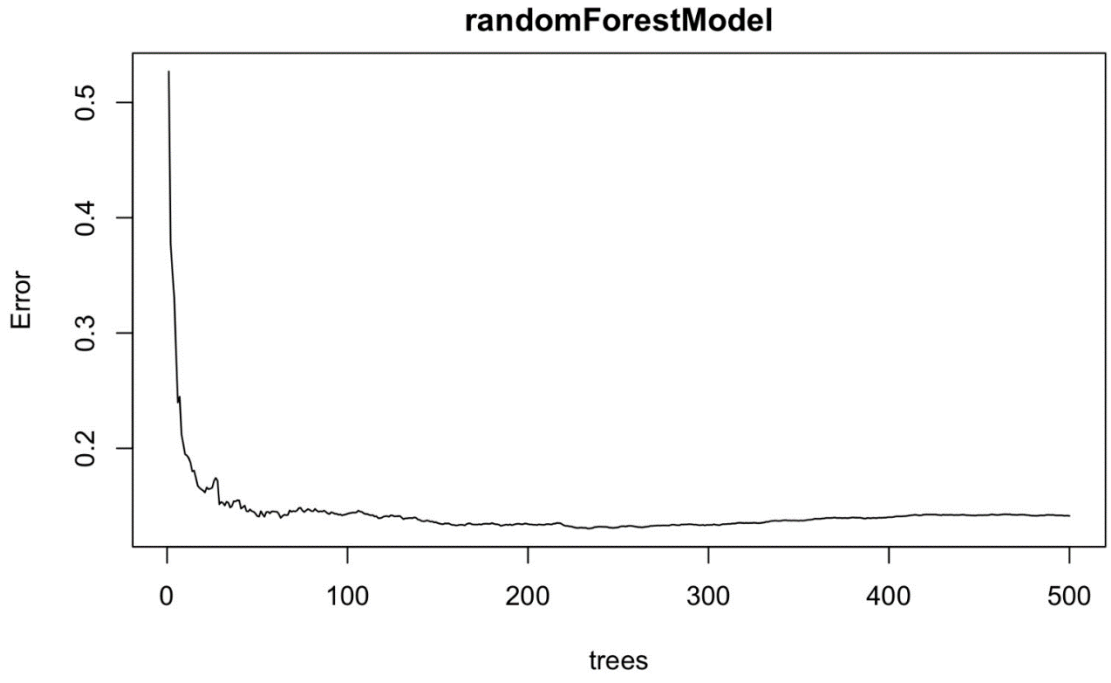

Fig. The graph of Error vs the Trees created

5)XGBoost –

```
#XGBOOST MODEL

set.seed(1902)

train_x = data.matrix(train[, -7])

train_y = train[,7]

test_x = data.matrix(test[, -7])

test_y = test[, 7]

xgb_train = xgb.DMatrix(data = train_x, label = train_y)

xgb_test = xgb.DMatrix(data = test_x, label = test_y)

watchlist = list(train=xgb_train, test=xgb_test)

model = xgb.train(data = xgb_train, max.depth = 3, watchlist=watchlist, nrounds = 70) #From the output we can see that the minimum testing RMSE is achieved at 26 rounds.

final = xgboost(data = xgb_train, max.depth = 3, nrounds = 26, verbose = 0) #we'll define our final XGBoost model to use 26 rounds

xgb.plot.tree(model = final, trees = 1:3)

pred_y = predict(final, test_x)

mean((test_y - pred_y)^2) #mse

MAE(test_y, pred_y) #mae

xg_rmse<-rmse(test_y, pred_y) #rmse 0.5573487


```

```
[1]     train-rmse:0.897246   test-rmse:0.866098
[2]     train-rmse:0.739680   test-rmse:0.767033
[3]     train-rmse:0.640073   test-rmse:0.680113
[4]     train-rmse:0.571949   test-rmse:0.623681
[5]     train-rmse:0.523814   test-rmse:0.590589
[6]     train-rmse:0.475864   test-rmse:0.624883
[7]     train-rmse:0.446896   test-rmse:0.617974
[8]     train-rmse:0.395943   test-rmse:0.566077
[9]     train-rmse:0.360114   test-rmse:0.552384
[10]    train-rmse:0.331065   test-rmse:0.551328
[11]    train-rmse:0.308852   test-rmse:0.560815
[12]    train-rmse:0.286277   test-rmse:0.575717
[13]    train-rmse:0.272002   test-rmse:0.570884
```

[14] train-rmse:0.263641 test-rmse:0.570806
[15] train-rmse:0.255383 test-rmse:0.570363
[16] train-rmse:0.245911 test-rmse:0.568605
[17] train-rmse:0.237166 test-rmse:0.567948
[18] train-rmse:0.225343 test-rmse:0.566148
[19] train-rmse:0.212451 test-rmse:0.568949
[20] train-rmse:0.198938 test-rmse:0.567187
[21] train-rmse:0.194421 test-rmse:0.564668
[22] train-rmse:0.182341 test-rmse:0.559164
[23] train-rmse:0.169826 test-rmse:0.554380
[24] train-rmse:0.164237 test-rmse:0.558524
[25] train-rmse:0.155566 test-rmse:0.556023
[26] train-rmse:0.148065 test-rmse:0.557349
[27] train-rmse:0.146544 test-rmse:0.557649
[28] train-rmse:0.139689 test-rmse:0.560763
[29] train-rmse:0.134906 test-rmse:0.563738
[30] train-rmse:0.131989 test-rmse:0.565147
[31] train-rmse:0.124874 test-rmse:0.562844
[32] train-rmse:0.122828 test-rmse:0.563370
[33] train-rmse:0.121044 test-rmse:0.564114
[34] train-rmse:0.118629 test-rmse:0.564537
[35] train-rmse:0.114708 test-rmse:0.563925
[36] train-rmse:0.112203 test-rmse:0.565329
[37] train-rmse:0.107976 test-rmse:0.565435
[38] train-rmse:0.104595 test-rmse:0.564920
[39] train-rmse:0.100645 test-rmse:0.565461
[40] train-rmse:0.098298 test-rmse:0.567951
[41] train-rmse:0.094117 test-rmse:0.566983
[42] train-rmse:0.092052 test-rmse:0.566956
[43] train-rmse:0.088535 test-rmse:0.566991
[44] train-rmse:0.085178 test-rmse:0.566165
[45] train-rmse:0.082362 test-rmse:0.566716
[46] train-rmse:0.079926 test-rmse:0.566752
[47] train-rmse:0.078685 test-rmse:0.567498
[48] train-rmse:0.074283 test-rmse:0.566310
[49] train-rmse:0.071731 test-rmse:0.566383
[50] train-rmse:0.070841 test-rmse:0.566452
[51] train-rmse:0.070058 test-rmse:0.566555
[52] train-rmse:0.068420 test-rmse:0.566996
[53] train-rmse:0.067011 test-rmse:0.565867
[54] train-rmse:0.066064 test-rmse:0.566169
[55] train-rmse:0.063994 test-rmse:0.564853
[56] train-rmse:0.062709 test-rmse:0.565801
[57] train-rmse:0.061567 test-rmse:0.565276

[58] train-rmse:0.059644 test-rmse:0.567288
[59] train-rmse:0.058105 test-rmse:0.568088
[60] train-rmse:0.056176 test-rmse:0.568228
[61] train-rmse:0.055627 test-rmse:0.568421
[62] train-rmse:0.054124 test-rmse:0.568500
[63] train-rmse:0.053086 test-rmse:0.568598
[64] train-rmse:0.051967 test-rmse:0.569344
[65] train-rmse:0.049996 test-rmse:0.567794
[66] train-rmse:0.049632 test-rmse:0.568462
[67] train-rmse:0.049237 test-rmse:0.567722
[68] train-rmse:0.048476 test-rmse:0.567442
[69] train-rmse:0.047141 test-rmse:0.568092
[70] train-rmse:0.045849 test-rmse:0.566548
[1] 0.3106376
[1] 0.4356466
Fig. Output for XGBoost

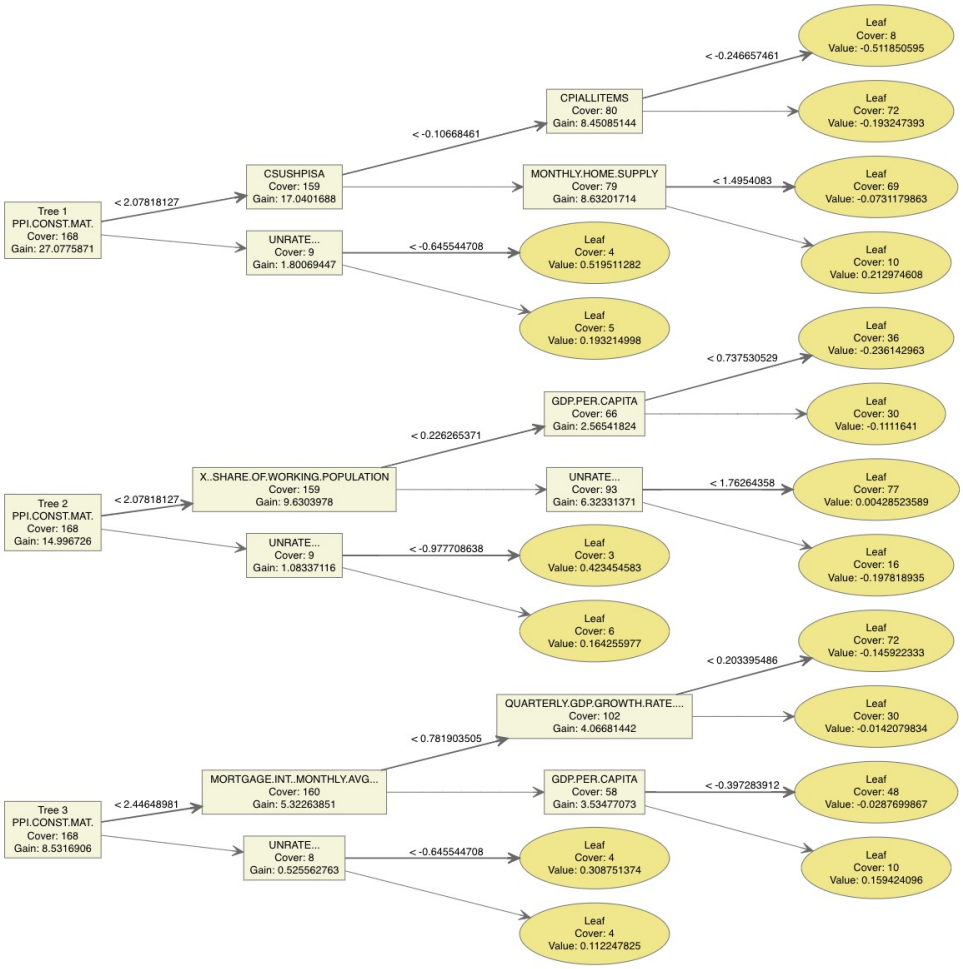

Fig. Graph of Trees created by XGBoost

After all the models were run, we compared the Root Mean Square Error values of each model using bar plots. After creating visualizations and finding that Ridge Regression is the most optimal model for prediction, we plot the values of actual test data in comparison to values of the predicted data by Ridge Regression.

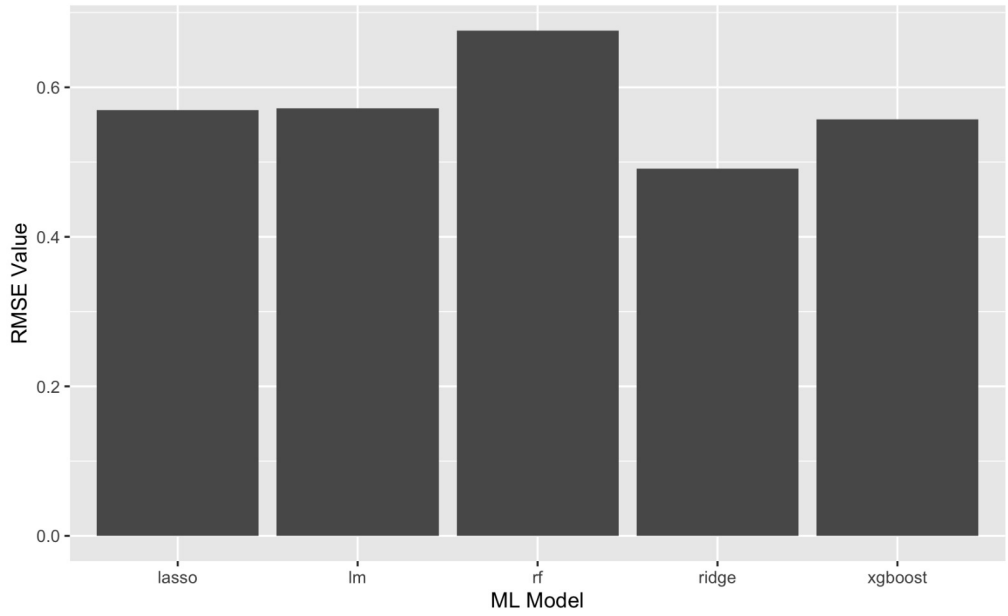

Fig. Bar Graph of the RMSE of models used

Discussion

The model provided an unexpected result with the best performing model being Ridge Regression in comparison to recent more complex algorithms such as Random Forest and XgBoost.

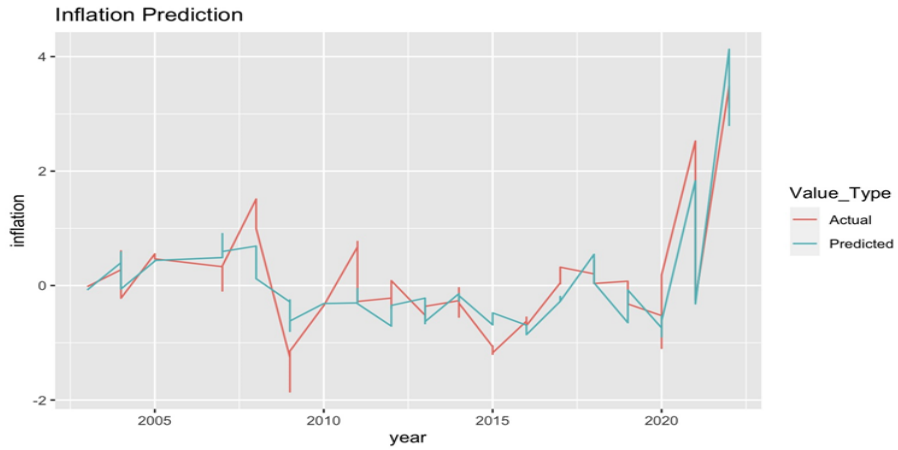

Fig. Actual Inflation data in test set vs the predicted values

As we can see from the figure above the values predicted by Ridge Regression closely match the pattern the actual test data of inflation is providing. The values also closely predict the

recession of 2008 and the anomaly of the COVID outbreak. There may be numerous factors for the discrepancy in the performance measure output such as the small number of data values in the dataset, the variables chosen for the model, etc.

The results for our model are quite close to the study done in paper[3], where the basic functions worked better for the predictive model. The limitations that we faced were that even though there were numerous variables available to us for predicting the inflation rate, inflation also depends on a lot of external factors out of which some are circumstantial and cannot be predicted easily such as the “COVID outbreak” which led to huge economic issues. Also, the future scope for this project would be to incorporate more factors that could help in the more precise and optimal prediction of inflation that could help businesses and individuals to cope with situations such as recessions in a more prepared manner.

Conclusion

We did the predictive analysis using various different models to predict inflation. The model with the best performance was Ridge Regression. There may be various factors affecting the performance measure of the Machine Learning algorithm and the simple fact that recent and more complex algorithms would always provide the most correct output cannot be stated, without experimentation on the chosen dataset and problem statement. Even though the values in the model did not perform with the best performance measure the model was able to very closely predict the pattern of inflation including the two anomalies of recession and the COVID outbreak.

References

[1] Wagner Piazza Gaglianone & Gustavo Silva Araujo, 2022.
"Machine Learning Methods for Inflation Forecasting in Brazil: new contenders versus classical models,"
Working Papers Series 561, Central Bank of Brazil, Research Department.

[2] R. Zhang, "Applying Data Mining Technology on Inflation Prediction in the United States," 2021 8th International Conference on Computational Science/Intelligence and Applied Informatics (CSII), 2021, pp. 34-37, doi: 10.1109/CSII54342.2021.00015.

[3] Inflation Prediction model using Machine Learning
Omprakash Yadav#1 Cynara Gomes#2 Abhishek Kanojiya#3 Abhishek Yadav#4
Department of Computer Engineering Xavier Institute of Engineering, Mumbai University