DAFTAR PUSTAKA

Amanda, R., Yasin, H., & Prahutama, A. (2014). *Analisis Support Vector Regression (SVR) Dalam Memprediksi Kurs Rupiah terhadap Dollar Amerika Serikat*. *3*(4), 849–857. http://ejournal-s1.undip.ac.id/index.php/gaussian

Arumsari, M., Wahyuningsih, S., & Siringoringo, M. (2021). Inflation Forecasting for East Kalimantan Province Using *Hybrid* Singular Spectrum Analysis- Autoregressive Integrated Moving Average Model. *Jurnal Matematika, Statistika Dan Komputasi*, *18*(1), 78–92. https://doi.org/10.20956/j.v18i1.14284

Aryani, F., & Maisyitah, R. A. D. (2015). Nilai Eigen dan Vektor Eigen dari Matriks Kompleks Bujursangkar Ajaib. *Jurnal Sains Matematika Dan Statistika*, *1*(2).

Aswi, & Sukarna. (2006). *Analisis Deret Waktu: Teori dan Aplikasinya*. Andira Publisher.

Athoillah, I., Wigena, A. H., & Wijayanto, H. (2021). *Hybrid* Modeling of Singular Spectrum Analysis and Support Vector Regression for Rainfall Prediction. *Journal of Physics: Conference Series*, *1863*(1). https://doi.org/10.1088/1742-6596/1863/1/012054

Awad, M., & Khanna, R. (2015). *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress Open.

Badan Pusat Statistik. (2022). *Indeks Harga Konsumen 90 Kota di Indonesia (2018=100) 2021*. https://www.bps.go.id/id/publication/2022/04/12/94319f7cdb3fdad9ad3ed742/indeks-harga-konsumen-90-kota-di-indonesia--2018-100--2021.html

Bank Indonesia. (2020). *Inflasi*. https://www.bi.go.id/id/fungsi-utama/moneter/inflasi/default.aspx

Bank Indonesia. (2021, February 12). *Pemerintah dan Bank Indonesia Sepakati Lima Langkah Strategis Menjaga Inflasi 2021* [Broadcast]. Bank Indonesia. https://www.bi.go.id/id/publikasi/ruang-media/news-release/Pages/sp_247622.aspx

Basari, M. S. N., & Achmad, A. I. (2021). Metode Singular Spectrum Analysis untuk Meramalkan Indeks Harga Konsumen Indonesia Tahun 2019. *Prosiding Statistika*, *7*(2). https://doi.org/10.29313/.v0i0.28777

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis Forecasting and Control*. Holden-Day.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (2008). *Time Series Analysis : Forecasting and Control* (4th Edition). J. Wiley & Sons.

Cahyono, R. E., Sugiono, J. P., & Tjandra, S. (2019). Analisis Kinerja Metode Support Vector Regression (SVR) dalam Memprediksi Indeks Harga Konsumen. *Jurnal Teknologi Informasi Dan Multimedia*, *1*(2), 106–116. www.siskaperbapo.com

Chatfield, Christopher. (2001). *Time-series forecasting*. Chapman & Hall/CRC.

Danandeh Mehr, A., Nourani, V., Karimi Khosrowshahi, V., & Ghorbani, M. A. (2019). A *Hybrid* Support Vector Regression–Firefly Model for Monthly Rainfall Forecasting. *International Journal of Environmental Science and Technology*, *16*(1), 335–346. https://doi.org/10.1007/s13762-018-1674-2

Darmawan, G. (2016). Identifikasi Pola Data Curah Hujan pada Proses Grouping dalam Metode Singular Spectrum Analysis. *Seminar Nasional Pendidikan Matematika*, 1–7.

Darmawan, G., Rosadi, D., & Ruchjana, B. N. (2022). *Hybrid* Model of Singular Spectrum Analysis and ARIMA for Seasonal Time Series Data. *CAUCHY – Jurnal Matematika Murni Dan Aplikasi*, *7*(2), 302–315. https://doi.org/10.18860/ca.v7i1.14136

Fajar, M., & Rachmad, S. H. (2020). Comparison of ARIMA, SSA, and ARIMA-SSA *Hybrid* Model Performance in Indonesian Economic Growth Forecasting. *Asia-Pasific Statistics Week*.

Golyandina, N., Korobeynikov, A., & Zhigljavsky, A. (2018). *Singular Spectrum Analysis with R*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57380-8

Golyandina, N., Nekrutkin, V., & Zhigljavsky, A. (2001). *Analysis of Time Series Structure: SSA and Related Techniques*. Chapman & Hall.

Hassani, H. (2021). Singular Spectrum Analysis: Methodology and Comparison. *Journal of Data Science*, *5*(2), 239–257. https://doi.org/10.6339/jds.2007.05(2).396

Hong, W. C. (2009). Electric Load Forecasting by Support Vector Model. *Applied Mathematical Modelling*, *33*(5), 2444–2454. https://doi.org/10.1016/j.apm.2008.07.010

Khaeri, H., Yulian, E., & Darmawan, G. (2018). Penerapan Metode Singular Spectrum Analysis (SSA) pada Peramalan Jumlah Penumpang Kereta Api di Indonesia Tahun 2017. *Jurnal Euclid*, *5*(1).

Lahmiri, S. (2018). Minute-ahead Stock Price Forecasting Based on Singular Spectrum Analysis and Support Vector Regression. *Applied Mathematics and Computation*, *320*, 444–451. https://doi.org/10.1016/j.amc.2017.09.049

Makridakis, S., Whellwright, S. C., & McGee, V. E. (1983). *Forecasting : Methods and Application*. John Wiley and Sons.

Mankiw, N. G. (2018). *Pengantar Ekonomi Makro* (7th ed.). Salemba Empat.

Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2008). *Introduction to Time Series Analysis and Forecasting*. Wiley.

Muslim, A. (2017). A *Hybrid* of ARIMA-ANFIS for Export Forecasting. *Kajian Ekonomi & Keuangan*, *1*(2). http://fiskal.kemenkeu.go.id/ejournal

Nugroho, A. S. (2003). Support Vector Machine Teori dan Aplikasinya dalam Bioinformatika. In *Kuliah Umum Ilmu Komputer*. http://asnugroho.net

Sakinah, A. M. (2012). *Perbandingan Stabilitas Hasil Peramalan Suhu dengan R-Forecasting dan V-Forecasting SSA untuk Long-Horizon*. Universitas Padjajaran. Bandung.

Santosa, B. (2007). *Data Mining (Teori dan Aplikasi)*. Graha Ilmu.

Smola, A. J., & Scholkopf, B. (2004). A Tutorial on Support Vector Regression. *Statistics and Computing*, *14*, 199–222.

Suseno, & Astiyah, S. (2009). *Inflasi*. Pusat Pendidikan dan Studi Kebanksentralan (PPSK). http://www.bi.go.id

Vahabie, A. H., Yousefi, M. M. R., Araabi, B. N., Lucas, C., & Barghinia, S. (2007). Combination of Singular Spectrum Analysis and Autoregressive Model for Short Term Load Forecasting. *IEEE Lausanne Power Tech*, 1090–1093. https://doi.org/10.1109/PCT.2007.4538467

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory* (1st ed.). Springer. https://doi.org/10.1007/978-1-4757-2440-0

Wei, W. W. S. (2006). *Time Series Analysis: Univariate and Multivariate Methods* (Second Edition). Pearson Education, Inc.

Xiao, J., Zhu, X., Huang, C., Yang, X., Wen, F., & Zhong, M. (2019). A New Approach for Stock Price Analysis and Prediction Based on SSA and SVM. *International Journal of Information Technology and Decision Making*, *18*(1), 35–63. https://doi.org/10.1142/S021962201841002X

Yasin, H., Prahutama, A., & Utami, T. W. (2014). Prediksi Harga Saham Menggunakan Support Vector Regression dengan Algoritma Grid Search. *Media Statistika*, *7*(No 1), 29–35.

Zhang, Q., Wang, B. De, He, B., Peng, Y., & Ren, M. L. (2011). Singular Spectrum Analysis and ARIMA *Hybrid* Model for Annual Runoff Forecasting. *Water Resources Management*, *25*(11), 2683–2703. https://doi.org/10.1007/s11269-011-9833-y

LAMPIRAN

Lampiran 1 Data Inflasi Indonesia

Bulan	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011
Januari	1,32	0,33	1,99	0,8	0,57	1,43	1,36	1,04	1,77	-0,07	0,84	0,89
Februari	0,07	0,87	1,5	0,2	-0,02	-0,17	0,58	0,62	0,65	0,21	-0,08	0,13
Maret	-0,45	0,89	-0,02	-0,23	0,36	1,91	0,03	0,24	0,95	0,22	-0,14	-0,32
April	0,56	0,46	-0,24	0,15	0,97	0,34	0,05	-0,16	0,57	-0,31	0,15	-0,31
Mei	0,84	1,13	0,8	0,21	0,88	0,21	0,37	0,1	1,41	0,04	0,29	0,12
Juni	0,5	1,67	0,36	0,09	0,48	0,5	0,45	0,23	2,46	0,11	0,97	0,55
Juli	1,28	2,12	0,82	0,03	0,39	0,78	0,45	0,72	1,37	0,45	1,57	0,67
Agustus	0,51	-0,21	0,29	0,84	0,09	0,55	0,33	0,75	0,51	0,56	0,76	0,93
September	-0,06	0,64	0,53	0,36	0,02	0,69	0,38	0,8	0,97	1,05	0,44	0,27
Oktober	1,16	0,68	0,54	0,55	0,56	8,7	0,86	0,79	0,45	0,19	0,06	-0,12
November	1,32	1,71	1,85	1,01	0,89	1,31	0,34	0,18	0,12	-0,03	0,6	0,34
Desember	1,94	1,62	1,2	0,94	1,04	-0,04	1,21	1,1	-0,04	0,33	0,92	0,57

Bulan	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022
Januari	0,76	1,03	1,07	-0,24	0,51	0,97	0,62	0,32	0,39	0,26	0,56
Februari	0,05	0,75	0,26	-0,36	-0,09	0,23	0,17	-0,08	0,28	0,1	-0,02
Maret	0,07	0,63	0,08	0,17	0,19	-0,02	0,2	0,11	0,1	0,08	0,66
April	0,21	-0,1	-0,02	0,36	-0,45	0,09	0,1	0,44	0,08	0,13	0,95
Mei	0,07	-0,03	0,16	0,5	0,24	0,39	0,21	0,68	0,07	0,32	0,4
Juni	0,62	1,03	0,43	0,54	0,66	0,69	0,59	0,55	0,18	-0,16	0,61
Juli	0,7	3,29	0,93	0,93	0,69	0,22	0,28	0,31	-0,1	0,08	0,64
Agustus	0,95	1,12	0,47	0,39	-0,02	-0,07	-0,05	0,12	-0,05	0,03	-0,21
September	0,01	-0,35	0,27	-0,05	0,22	0,13	-0,18	-0,27	-0,05	-0,04	1,17
Oktober	0,16	0,09	0,47	-0,08	0,14	0,01	0,28	0,02	0,07	0,12	-0,11
November	0,07	0,12	1,5	0,21	0,47	0,2	0,27	0,14	0,28	0,37	0,09
Desember	0,54	0,55	2,46	0,96	0,42	0,71	0,62	0,34	0,45	0,57	0,66

Lampiran 2 Syntax SSA-SVR

#SSASVR

```
library(readxl)
original.data <- read_excel("D://Thesis/Inflasi BPS.xlsx",
                            col_types = c("date", "numeric"))
tail(original.data)
## # A tibble: 6 × 2
##   Bulan               Inflasi
##   <dttm>                <dbl>
## 1 2022-07-01 00:00:00    0.64
## 2 2022-08-01 00:00:00   -0.21
## 3 2022-09-01 00:00:00    1.17
## 4 2022-10-01 00:00:00   -0.11
## 5 2022-11-01 00:00:00    0.09
## 6 2022-12-01 00:00:00    0.66
```

#Eksplorasi Data
Pola data inflasi

```
original.data$Bulan <- lubridate::as_date(original.data$Bulan)
head(original.data)
## # A tibble: 6 × 2
##   Bulan      Inflasi
##   <date>       <dbl>
## 1 2000-01-01    1.32
## 2 2000-02-01    0.07
## 3 2000-03-01   -0.45
## 4 2000-04-01    0.56
## 5 2000-05-01    0.84
## 6 2000-06-01    0.5
library(ggplot2)
## Warning: package 'ggplot2' was built under R version 4.2.2
ggplot(data = original.data, aes(x = Bulan, y = Inflasi))+
  geom_line(color = "black", size = 1) + xlab("")+
  #scale_x_date(date_breaks = "3 years") +
  theme_classic() +
  theme(axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"))
```

Split Data
Data training

```
data.training <- ts(original.data$Inflasi, start=c(2000, 1),
                    end = c(2020,12), frequency=12)
plot(data.training, las=1, bty='l', ylab="Inflasi", xlab="")
length(data.training)
## [1] 252
tail(data.training)
## [1] -0.10 -0.05 -0.05  0.07  0.28  0.45
```

Data testing

```
data.testing <- ts(original.data$Inflasi[253:276],
                   start = c(2021,1), end = c(2022,12), frequency=12)
data.testing
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   De
c
## 2021  0.26  0.10  0.08  0.13  0.32 -0.16  0.08  0.03 -0.04  0.12  0.37  0.5
7
```

```
## 2022  0.56 -0.02  0.66  0.95  0.40  0.61  0.64 -0.21  1.17 -0.11  0.09  0.6
6
```

SSA

Periodesasi: pemilihan L

Embed

```
L <- 100
K <- (252-L)+1
Z <- as.matrix(data.training)
X <- embed(Z,L)
LK <- t(X)
# write.csv2(LK,"D://Embedding100.csv")
```

SVD

Hanckel Matrix

```
id <- 1:L
Y <- rbind(id,X)
sort <- Y[,order(-Y[1,])]
THankel <- as.matrix(sort[-1,(1:L)])
Hankel <- t(THankel)
dim(Hankel)
## [1] 100 153
A=(X.svd<-svd(Hankel))
Hankel
```

Trajectory

```
traject <- Hankel %*% t(Hankel)
eval <- eigen(traject)$value #SVD
total_ragam <- sum(eval)
prop_ragam <- eval/total_ragam
kum_ragam <- cumsum(prop_ragam)
kum_ragam
evec <- eigen(traject)$vectors
library(Rssa)
s12 <- ssa(data.training, L= 100)
plot(s12)
plot(s12, type = "vectors", idx = 1:31)
plot(wcor(s12))
```

Rekonstruksi

Grouping

```
recon<-reconstruct(s12, groups =list(trend=c(1),
                                     seasonal=c(2:9,12:15), noise=c(10,11,16:50)
),
                   drop.attributes = F)
```

Plot hasil rekonstruksi

```
plot(wcor(s12, groups = list(trend=c(1),
                             seasonal=c(2:9,12:15))))
```

```
komponen <- cbind(recon$trend, recon$seasonal)
diag.a <- rowSums(komponen)
```

Diagonal Averaging

```
komponen <- cbind(recon$trend, recon$seasonal)
diag.a <- rowSums(komponen)
```

Prediksi data testing (LRF)

```
komp_pred <- rforecast(s12, groups = list(trend=c(1),
                               seasonal=c(2:9,12:15)), len = length(data.testing
))
prediksi <- komp_pred$trend + komp_pred$seasonal
prediksi
##              Jan         Feb         Mar         Apr         May          Ju
n
## 2021  0.29759764  0.06236956  0.04744284  0.28491384  0.49074325  0.4189591
2
## 2022  0.21951963  0.13036674  0.17738322  0.29921529  0.35573223  0.3004658
3
##              Jul         Aug         Sep         Oct         Nov          De
c
## 2021  0.14966542 -0.03155128  0.03747288  0.25075503  0.39082993  0.3562954
8
## 2022  0.22044574  0.19302611  0.19529813  0.19337367  0.21538047  0.2679633
5
rmse <- function(errval)
{
  val = sqrt(mean(errval^2))
  return(val)
}
rmse.SSA <- sqrt(mean((data.testing - prediksi)^2))
rmse.SSA
## [1] 0.3483757
trend <- recon$trend
plot(trend)
```

```
seasonality <- recon$seasonal

plot(seasonality)
```

```
noise <- residuals(recon)
noise
plot(noise)
```

#Penyiapan variabel bebas untuk SVR ##Ekstrak komponen ET

```
T.train <- recon$trend
S.train <- recon$seasonal
N.train <- residuals(recon)
```

#Indentifikasi lag signifikan

```
library(tseries)
library(data.table)
pacf(T.train, lag.max = 50, main='PACF Trend Inflasi Indonesia Metode SSA')

pacf(S.train, lag.max = 50, main='PACF Seasonal Inflasi Indonesia Metode SSA')

pacf(N.train, lag.max = 50, main='PACF Noise Inflasi Indonesia Metode SSA')

Box.test(N.train,type = "Ljung-Box")##whitenoise
##
##  Box-Ljung test
##
```

```
## data:  N.train
## X-squared = 21.097, df = 1, p-value = 4.366e-06
```

##Komponen tren Seasonal dan noise

```
lag.T.train <- data.frame(T.train)
colnames(lag.T.train) <-"x_Ttrain"
nTt <- 1
x_T.train <- setDT(lag.T.train)[,paste0("x_Ttrain",1:nTt) :=shift(x_Ttrain,1:n
Tt)][]
T.t <- as.matrix.data.frame(x_T.train[,-2])
lag.S.train <- data.frame(S.train)
colnames(lag.S.train) <-"x_Strain"
nSt <- 18
x_S.train <- setDT(lag.S.train)[,paste0("x_Strain",1:nSt) :=shift(x_Strain,1:n
St)][]
S.t <- as.matrix.data.frame(x_S.train[,-19])
lag.N.train <- data.frame(N.train)
colnames(lag.N.train) <-"x_Ntrain"
nNt <- 13
x_N.train <- setDT(lag.N.train)[,paste0("x_Ntrain",1:nNt) :=shift(x_Ntrain,1:n
Nt)][]
N.t <- as.matrix.data.frame(x_N.train[,-14])
training.svr <- cbind(original.data[c(1:252),2], T.t, S.t, N.t)
colnames(training.svr) <- c('Y','T1','S1','S2','S3','S4','S5','S6','S7','S8','
S9','S10','S11','S12','S13','S14','S15',
                            'S16' ,'S17','S18','N1','N2','N3','N4','N5','N6','
N7','N8','N9','N10','N11','N12','N13')

train.svr <- training.svr[-1:-18,]
dim(train.svr)
## [1] 234  33
# Kebutuhan dekomposisi variabel X
original.ts <- ts(original.data$Inflasi, start=c(2000, 1), frequency=12)
##decomposition stage on validation set sequentially
N <- length(original.ts)
K.sliding <- 252 #panjang data training
L <- 100
ck <- list()
for (i in 1:24){
  F.train <- original.ts[seq(1, len = K.sliding+i)]
  s1 <- ssa(F.train, L = L)
  r1 <- reconstruct(s1, groups = list(c(1),c(2:9,12:15)))
  pred <- r1$F1+r1$F2
  T1<- r1$F1
  S1<-r1$F2
  N1 <-residuals(r1)

  library(data.table)
  lag.T1 <- data.frame(T1)
  colnames(lag.T1) <-"x"

   n <- 1
  data.trend.temp<-setDT(lag.T1)[,paste0("x",1:n) :=shift(x,1:n)][]
  val.t.12<-tail(data.trend.temp[c(-1:-1),-2],1)##
  colnames(val.t.12) <-c('T1')##
  lag.S1 <- data.frame(S1)
  colnames(lag.S1) <-"x"
```

```
  n <- 18
  data.sea.temp<-setDT(lag.S1)[,paste0("x",1:n) :=shift(x,1:n)][]
  val.s.12<-tail(data.sea.temp[c(-1:-18),-19],1)##
  colnames(val.s.12) <- c('S1','S2','S3','S4','S5','S6','S7','S8','S9','S10','
S11','S12','S13','S14','S15','S16',
                          'S17','S18')##
  lag.N1 <- data.frame(N1)
  colnames(lag.N1) <-"x"

  n <- 13
  data.noi.temp<-setDT(lag.N1)[,paste0("x",1:n) :=shift(x,1:n)][]
  val.n.12<-tail(data.noi.temp[c(-1:-13),-14],1)##
  colnames(val.n.12) <- c('N1','N2','N3','N4','N5','N6','N7','N8','N9','N10','
N11','N12','N13')##

  val.12 <- cbind(val.t.12, val.s.12, val.n.12)##
    ck[[i]] <- val.12
}
# str(val.n.12)
big_data = do.call(rbind, ck)
testing.svr <- as.data.frame(cbind(data.testing, big_data))
colnames(testing.svr) <- c('Y','T1','S1','S2','S3','S4','S5','S6','S7','S8','S
9',
                     'S10','S11','S12','S13','S14','S15','S16','S17','S18',
                     'N1','N2','N3','N4','N5','N6','N7','N8','N9','N10','N11'
,'N12','N13')

library(e1071)
rmse <- function(errval)
{
  val = sqrt(mean(errval^2))
  return(val)
}

gammas = 2^(-8:3)
costs = 2^(-5:8)
epsilons = c(0.1, 0.01, 0.001)
```

```
set.seed(1001)
tune_radial <- tune(svm,
                     Y~.,
                     data= training.svr,
                     type = "eps-regression",
                     kernel = "radial",
                     ranges = list(gamma = gammas, cost = costs,
                                   epsilon = epsilons),
                     tunecontrol = tune.control(cross = 10)
)
grid_radial <- tune_radial$best.model
grid_radial
##
## Call:
## best.tune(method = svm, train.x = Y ~ ., data = training.svr, ranges = list
(gamma = gammas,
##     cost = costs, epsilon = epsilons), tunecontrol = tune.control(cross = 1
```

```
0),
##     type = "eps-regression", kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  256
##       gamma:  0.00390625
##     epsilon:  0.1
##
##
## Number of Support Vectors:  188
model_radial <- svm(Y~.,
                    data= training.svr,
                    cost= 256,
                    gamma= 0.00390625,
                    epsilon=0.1,
                    type="eps-regression",
                    kernel="radial")
pred_radial <- predict(model_radial, testing.svr[,-1])

err.radial <- testing.svr$Y - pred_radial
RMSE.radial <- rmse(err.radial)
cor_radial <- cor(testing.svr$Y, pred_radial,method = "pearson")
RMSE.radial; cor_radial
## [1] 0.1894508
## [1] 0.9029819
set.seed(1001)
tune_linear <- tune(svm,
                      Y~.,
                      data= training.svr,
                      type = "eps-regression",
                      kernel = "linear",
                      ranges = list(gamma = gammas, cost = costs,
                                    epsilon = epsilons),
                      tunecontrol = tune.control(cross = 10)
)
grid_linear <- tune_linear$best.model
grid_linear
##
## Call:
## best.tune(method = svm, train.x = Y ~ ., data = training.svr, ranges = list
(gamma = gammas,
##     cost = costs, epsilon = epsilons), tunecontrol = tune.control(cross = 1
0),
##     type = "eps-regression", kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  128
##       gamma:  0.00390625
##     epsilon:  0.01
##
## Number of Support Vectors:  228
```

```
model_linear <- svm(Y~.,
                    data= training.svr,
                    cost= 128,
                    gamma= 0.00390625,
                    epsilon=0.01,
                    type="eps-regression",
                    kernel="linear")
pred_linear <- predict(model_linear, testing.svr[,-1])
err.linear <- testing.svr$Y - pred_linear
RMSE.linear <- rmse(err.linear)
cor_linear <- cor(testing.svr$Y, pred_linear, method = "pearson")
RMSE.linear; cor_linear
## [1] 0.173663
## [1] 0.8870483
set.seed(1001)
tune_polinom <- tune(svm,
                     Y~.,
                     data= training.svr,
                     type = "eps-regression",
                     kernel = "polynomial",
                     ranges = list(gamma = gammas, cost = costs,
                                   epsilon = epsilons),
                     tunecontrol = tune.control(cross = 10)
)
grid_polinom <- tune_polinom$best.model
grid_polinom
##
## Call:
## best.tune(method = svm, train.x = Y ~ ., data = training.svr, ranges = list
(gamma = gammas,
##     cost = costs, epsilon = epsilons), tunecontrol = tune.control(cross = 1
0),
##     type = "eps-regression", kernel = "polynomial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  polynomial
##        cost:  0.125
##      degree:  3
##       gamma:  0.0625
##      coef.0:  0
##     epsilon:  0.1
##
##
## Number of Support Vectors:  189
model_polinom <- svm(Y~.,
                     data= training.svr,
                     cost= 0.125,
                     degree = 3,
                     coef.0 = 0,
                     gamma= 0.0625,
                     epsilon=0.1,
                     type="eps-regression",
                     kernel="polynomial")
```

```
pred_polinom <- predict(model_polinom, testing.svr[,-1])

err.polinom <- testing.svr$Y - pred_polinom
RMSE.polinom <- rmse(err.polinom)
cor_polinom <- cor(testing.svr$Y, pred_polinom, method = "pearson")

RMSE.polinom; cor_polinom
## [1] 0.3067434
## [1] 0.830467
RMSE.radial; RMSE.linear; RMSE.polinom
## [1] 0.1894508
## [1] 0.173663
## [1] 0.3067434
```

Lampiran 3 Syntax SSA-ARIMA

```
library(tseries)
library(lmtest)
library(nortest)
library(forecast)
noise <- N.train
# noise<-noise*100
#cek kestasioneran
adf.test(noise)
## Warning in adf.test(noise): p-value smaller than printed p-value
##
##  Augmented Dickey-Fuller Test
##
## data:  noise
## Dickey-Fuller = -9.2392, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
# noise = diff(noise)
Box.test(noise)
##
##  Box-Pierce test
##
## data:  noise
## X-squared = 20.848, df = 1, p-value = 4.973e-06
plot(noise)
```

```
library(MASS)
BoxCox.lambda(noise)
## Warning in guerrero(x, lower, upper): Guerrero's method for selecting a Box
-Cox
## parameter (lambda) is given for strictly positive data.
## [1] 1.006792
ggAcf(noise, lag.max = 50)
```

```
ggPacf(noise, lag.max = 50)
```

```
mod1 <- Arima(noise,order = c(1,0,1), method= "ML")
summary(mod1)
## Series: noise
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##           ar1     ma1     mean
##       -0.4888  0.2190  -0.0031
## s.e.   0.1466  0.1595   0.0134
##
## sigma^2 = 0.06784:  log likelihood = -17.09
## AIC=42.18   AICc=42.34   BIC=56.29
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.0001495414 0.2588985 0.1900066 158.8189 227.7004 0.6523288
##                      ACF1
## Training set 0.004636628
```

```
coeftest(mod1)
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.4887807  0.1465684 -3.3348 0.0008535 ***
## ma1        0.2189711  0.1595149  1.3727 0.1698359
## intercept -0.0030845  0.0133615 -0.2309 0.8174311
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#uji white noise pakai uji ljung box
Box.test(mod1$residuals,type = "Ljung")
##
##  Box-Ljung test
##
## data:  mod1$residuals
## X-squared = 0.0054823, df = 1, p-value = 0.941
#Uji Residual berdistribusi normal pakai Shapiro test
shapiro.test(mod1$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  mod1$residuals
## W = 0.95249, p-value = 2.466e-07
mod2 <- Arima(noise,order = c(1,0,2), method= "ML")
summary(mod2)
## Series: noise
## ARIMA(1,0,2) with non-zero mean
##
## Coefficients:
##           ar1     ma1     ma2     mean
##       -0.4105  0.1637  0.0838  -0.0029
## s.e.   0.1603  0.1566  0.1027   0.0144
##
## sigma^2 = 0.06795:  log likelihood = -16.79
## AIC=43.59   AICc=43.83   BIC=61.24
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.0002687874 0.2585888 0.1903216 139.2611 202.6137 0.6534101
##                      ACF1
## Training set -0.009706881
coeftest(mod2)
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## ar1       -0.4104851  0.1603311 -2.5602  0.01046 *
## ma1        0.1636673  0.1566098  1.0451  0.29599
## ma2        0.0838051  0.1027216  0.8158  0.41459
## intercept -0.0029317  0.0144105 -0.2034  0.83879
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#uji white noise pakai uji ljung box
Box.test(mod2$residuals,type = "Ljung")
##
##  Box-Ljung test
```

```
##
## data:  mod2$residuals
## X-squared = 0.024028, df = 1, p-value = 0.8768
#Uji Residual berdistribusi normal pakai Shapiro test
shapiro.test(mod2$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  mod2$residuals
## W = 0.95374, p-value = 3.396e-07
mod3 <- Arima(noise,order = c(1,0,3), method= "ML")
summary(mod3)
## Series: noise
## ARIMA(1,0,3) with non-zero mean
##
## Coefficients:
##          ar1      ma1     ma2      ma3     mean
##       0.0783  -0.3650  0.3391  -0.4968  -0.0023
## s.e.  0.1226   0.1043  0.0588   0.0668   0.0081
##
## sigma^2 = 0.06137:  log likelihood = -3.86
## AIC=19.72   AICc=20.07   BIC=40.9
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE    MAPE      MASE
## Training set -0.0001150937 0.2452502 0.1812114 250.0027 356.602 0.6221331
##                     ACF1
## Training set 0.007739629
coeftest(mod3)
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1        0.0782625  0.1226309  0.6382 0.5233461
## ma1       -0.3650391  0.1043366 -3.4987 0.0004676 ***
## ma2        0.3390692  0.0588052  5.7660 8.119e-09 ***
## ma3       -0.4968276  0.0668466 -7.4324 1.067e-13 ***
## intercept -0.0023294  0.0080784 -0.2884 0.7730750
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#uji white noise pakai uji ljung box
Box.test(mod3$residuals,type = "Ljung")
##
##  Box-Ljung test
##
## data:  mod3$residuals
## X-squared = 0.015276, df = 1, p-value = 0.9016
#Uji Residual berdistribusi normal pakai Shapiro test
shapiro.test(mod3$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  mod3$residuals
## W = 0.96862, p-value = 2.417e-05
mod4 <- Arima(noise,order = c(2,0,3), method= "ML")
summary(mod4)
```

```
## Series: noise
## ARIMA(2,0,3) with non-zero mean
##
## Coefficients:
##           ar1      ar2      ma1     ma2      ma3     mean
##       -0.1061  -0.3358  -0.1661  0.5576  -0.4425  -0.0030
## s.e.   0.1731   0.1326   0.1730  0.0992   0.0894   0.0101
##
## sigma^2 = 0.06017:  log likelihood = -0.86
## AIC=15.72   AICc=16.18   BIC=40.43
##
## Training set error measures:
##                         ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.0002634425 0.2423542 0.1788236 262.9788 355.8576 0.6139353
##                     ACF1
## Training set 0.01641201
coeftest(mod4)
##
## z test of coefficients:
##
##             Estimate Std. Error z value  Pr(>|z|)
## ar1       -0.1061319  0.1730730 -0.6132   0.53973
## ar2       -0.3357855  0.1326271 -2.5318   0.01135 *
## ma1       -0.1661086  0.1729567 -0.9604   0.33685
## ma2        0.5575836  0.0991541  5.6234 1.872e-08 ***
## ma3       -0.4425308  0.0893561 -4.9524 7.329e-07 ***
## intercept -0.0029508  0.0100871 -0.2925   0.76988
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#uji white noise pakai uji ljung box
Box.test(mod4$residuals,type = "Ljung")
##
##  Box-Ljung test
##
## data:  mod4$residuals
## X-squared = 0.068689, df = 1, p-value = 0.7933
#Uji Residual berdistribusi normal pakai Shapiro test
shapiro.test(mod4$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  mod4$residuals
## W = 0.96389, p-value = 5.652e-06
mod5 <- Arima(noise,order = c(3,0,3), method= "ML")
summary(mod5)
## Series: noise
## ARIMA(3,0,3) with non-zero mean
##
## Coefficients:
##           ar1     ar2     ar3      ma1     ma2      ma3     mean
##       -0.0016  0.1146  0.4941  -0.3019  0.1077  -0.8058  -0.0001
## s.e.   0.1568  0.0969  0.1096   0.1293  0.1237   0.0545   0.0012
##
## sigma^2 = 0.05756:  log likelihood = 3.74
## AIC=8.53   AICc=9.12   BIC=36.76
##
## Training set error measures:
```

```
##                       ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.006915868 0.2365678 0.1751588 149.3528 236.4543 0.6013533
##                     ACF1
## Training set 0.02083881
coeftest(mod5)
##
## z test of coefficients:
##
##              Estimate  Std. Error  z value  Pr(>|z|)
## ar1       -0.00163276  0.15684845  -0.0104   0.99169
## ar2        0.11458317  0.09690469   1.1824   0.23703
## ar3        0.49414312  0.10964424   4.5068 6.582e-06 ***
## ma1       -0.30194683  0.12931715  -2.3349   0.01955 *
## ma2        0.10771595  0.12366938   0.8710   0.38375
## ma3       -0.80576840  0.05453377 -14.7756 < 2.2e-16 ***
## intercept -0.00013372  0.00122507  -0.1092   0.91308
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#uji white noise pakai uji ljung box
Box.test(mod5$residuals,type = "Ljung")
##
##  Box-Ljung test
##
## data:  mod5$residuals
## X-squared = 0.11074, df = 1, p-value = 0.7393
#Uji Residual berdistribusi normal pakai Shapiro test
shapiro.test(mod5$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  mod5$residuals
## W = 0.9659, p-value = 1.036e-05
fore <- forecast(mod5, h = 24)
fore$mean
##              Jan         Feb         Mar         Apr         May         Ju
n
## 2021 0.242402075 0.119221806 0.093519216 0.133236894 0.069358278 0.06131279
6
## 2022 0.022981518 0.014793042 0.014260012 0.012975368 0.008870111 0.00846622
2
##              Jul         Aug         Sep         Oct         Nov         De
c
## 2021 0.073632736 0.041125565 0.038614680 0.040981818 0.024627053 0.02368425
4
## 2022 0.007361690 0.005288630 0.004965875 0.004183068 0.003122975 0.00287552
3
#Hasil Ramalan SSA ARIMA
SSAARIMA <- prediksi + fore$mean
SSAARIMA
##              Jan         Feb         Mar         Apr         May         Ju
n
## 2021 0.539999717 0.181591362 0.140962058 0.418150737 0.560101530 0.48027191
4
## 2022 0.242501147 0.145159783 0.191643227 0.312190655 0.364602344 0.30893205
6
##              Jul         Aug         Sep         Oct         Nov         De
c
```

```
## 2021 0.223298155 0.009574286 0.076087559 0.291736852 0.415456981 0.37997973
0
## 2022 0.227807431 0.198314741 0.200264005 0.197556742 0.218503444 0.27083887
1

#RMSE
sqrt(mean((data.testing - (prediksi + fore$mean))^2))
## [1] 0.3609707
#Plot Hasil Prediksi
Aktual <- data.testing
SSASVR <- pred_linear
SSAARIMA <- SSAARIMA

wana <- data.frame("Bulan"=c(1:24), cbind(Aktual, SSASVR, SSAARIMA))
wana
##    Bulan Aktual      SSASVR    SSAARIMA
## 1      1   0.26  0.21688490 0.539999717
## 2      2   0.10 -0.03675304 0.181591362
## 3      3   0.08  0.02649998 0.140962058
## 4      4   0.13  0.23251347 0.418150737
## 5      5   0.32  0.42288142 0.560101530
## 6      6  -0.16 -0.10783694 0.480271914
## 7      7   0.08 -0.02278952 0.223298155
## 8      8   0.03  0.13874675 0.009574286
## 9      9  -0.04 -0.16170434 0.076087559
## 10    10   0.12  0.10539246 0.291736852
## 11    11   0.37  0.34150554 0.415456981
## 12    12   0.57  0.45281111 0.379979730
## 13    13   0.56  0.47498536 0.242501147
## 14    14  -0.02 -0.03680012 0.145159783
## 15    15   0.66  0.26846386 0.191643227
## 16    16   0.95  0.72678751 0.312190655
## 17    17   0.40  0.44726293 0.364602344
## 18    18   0.61  0.48037859 0.308932056
## 19    19   0.64  0.56106565 0.227807431
## 20    20  -0.21 -0.12060186 0.198314741
## 21    21   1.17  0.99468090 0.200264005
## 22    22  -0.11  0.18012588 0.197556742
## 23    23   0.09 -0.42556770 0.218503444
## 24    24   0.66  0.66082523 0.270838871
library(reshape2)
data_long <- melt(wana, id.vars = "Bulan")    # Reshaping data to long format
head(data_long)                               # Head of long data
##   Bulan variable value
## 1     1   Aktual  0.26
## 2     2   Aktual  0.10
## 3     3   Aktual  0.08
## 4     4   Aktual  0.13
## 5     5   Aktual  0.32
## 6     6   Aktual -0.16
tail(data_long)
##    Bulan variable     value
## 67    19 SSAARIMA 0.2278074
## 68    20 SSAARIMA 0.1983147
## 69    21 SSAARIMA 0.2002640
## 70    22 SSAARIMA 0.1975567
```

```
## 71    23 SSAARIMA 0.2185034
## 72    24 SSAARIMA 0.2708389
library(ggplot2)
ggplot(data_long,                                # Draw ggplot2 time series plot
       aes(x = Bulan,
           y = value,
           col = variable)) +
  geom_line() +
  theme_classic() +
  facet_wrap(~ variable, ncol = 1) +
  theme(legend.position = "none")
```

Lampiran 4 Riwayat Hidup

A. Data Pribadi
   1. Nama : Nur Ikhwana
   2. Tempat, Tgl Lahir : Baliti, 1 Oktober 2000
   3. Alamat : Jl. S. Kelara
   4. Kewarganegaraan : Indonesia

B. Riwayat Pendidikan
   1. Tamat SMA Tahun 2017 di SMA Negeri 1 Makassar
   2. Sarjana (S1) Tahun 2021 di Universitas Negeri Makassar
   3. Magister (S2) Tahun 2024 di Universitas Hasanuddin

C. Karya ilmiah yang telah dipublikasikan

Ikhwana, N., Nusrang, M., & Sudarmin, S. (2021). Perbandingan metode PCA-SVM dan SVM untuk Klasifikasi Indeks Kepuasan Masyarakat Terhadap Layanan Pendidikan di Kabupaten Jeneponto. *VARIANSI Journal of Statistics and Its Application on Teaching and Research*, Vol 3 No (3), doi: https://doi.org/10.35580/variansiunm22988