Projekt_2

Kamil Sarzyniak, Piotr Stawarski

2024-11-03

```
library("tidyverse")
library("moments")
library("tseries")
library("forecast")
library("rugarch")
library("FinTS")
```

Dane wykorzystywane w projekcie pochodzą z firmy “Izolacja - Jarocin S.A”, które pochodzą z okresu od września 2023 do 4 października 2024.

```
izo <- read.csv("C:/Visual_Studio/Ekonometria finansowa i dynamiczna/Projekty/Projekt_2/izo_d.csv")
```

Stopa logarytmiczna

```
st_l <- function(a,b) {
  return(log(a/b))
}
```

Zaczytanie danych

```
dane <- izo[,c(1,5)]
indeksy <- dane %>% map_df(rev)
```

Wyliczanie stóp logarytmicznych

```
dt <- c()
for (i in 1:(nrow(indeksy)-1)) {
  dt <- c(dt,as.numeric(st_l(indeksy[i,2],indeksy[i+1,2])))
}
```

Podział na zbiór uczący i odłożone dane

```
stopy <- rev(tail(dt, -4))
ceny <- unname(head(log(dane[2]), -4))
stopy_oct <- rev(head(dt,4))
ceny_oct <- unname(tail(log(dane[2]), 4))
ceny_n <- unname(tail(dane[2], 4))
```

Test ADF

```
adf.test(stopy)
```

```
Warning in adf.test(stopy) : p-value smaller than printed p-value
```

```

	Augmented Dickey-Fuller Test

data:  stopy
Dickey-Fuller = -6.8519, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```

```
#Bardzo niskie p-value -> szereg stacjonarny
```

ARMA

```
model <- auto.arima(stopy)
summary(model)
```

```
Series: stopy
ARIMA(3,0,2) with zero mean

Coefficients:
          ar1      ar2      ar3     ma1     ma2
      -0.9508  -1.1272  -0.2987  0.7809  0.8236
s.e.   0.0922   0.0655   0.0791  0.0767  0.0652

sigma^2 = 0.001116:  log likelihood = 538.4
AIC=-1064.8   AICc=-1064.48   BIC=-1043.18

Training set error measures:
                      ME       RMSE        MAE MPE MAPE      MASE       ACF1
Training set 0.001247395 0.03309945 0.02055427 NaN  Inf 0.6115581 0.06639422
```

```
#Zaproponowany model: ARMA(3,2)
```

Reszty modelu

```
res <- residuals(model)
Box.test(res, type = "Ljung-Box")
```

```

	Box-Ljung test

data:  res
X-squared = 1.2079, df = 1, p-value = 0.2717
```

```
#P-value większe od 0.05 -> brak aurokorelacji
jarque.bera.test(res)
```

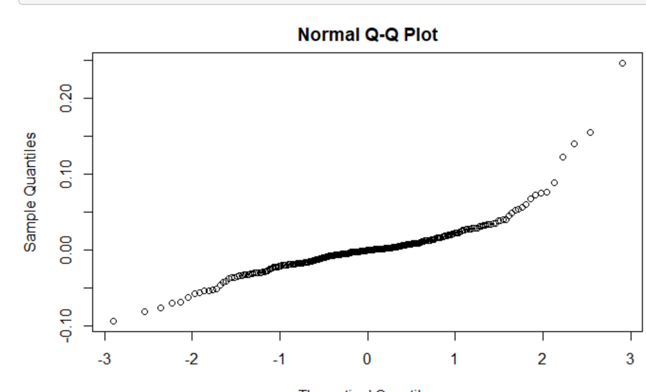
```

	Jarque Bera Test

data:  res
X-squared = 2287.2, df = 2, p-value < 2.2e-16
```

```
#P-value mniejsze od 0.05 -> reszty nie mają rozkładu normalnego
```

```
qqnorm(res)
```



Prognoza logarytmicznych stóp dla 4 przyszłych notowań i porównanie z odłożonymi wartościami

```
stopy_wyniki <- data.frame(
  Odłożone_Wartości = stopy_oct,
  Początek_Przedziału = forecast(model,4)$lower[, 2],
  Koniec_Przedziału = forecast(model,4)$upper[, 2],
  Średnia_Prognoza = forecast(model,4)$mean
)
print(stopy_wyniki)
```

Odłożone_Wartości	Początek_Przedziału	Koniec_Przedziału	Średnia_Prognoza
<dbl>	<dbl>	<dbl>	<dbl>
-0.021334142	-0.07150475	0.05945646	-0.006024150
0.000000000	-0.08026062	0.05257708	-0.013841770
0.000000000	-0.05835835	0.07577583	0.008708742
-0.002699057	-0.05797036	0.07621352	0.009121580

4 rows

Choć prognozy na pierwszym i trzecim notowaniu wykazują zgodność z rzeczywistością, kolejne notowania pokazują ujemne wartości średniej prognozy, co sugeruje przewidywanie spadków w tych okresach.

Model dla logarytmów cen

```
model2 <- auto.arima(ceny)
summary(model2)
```

```
Series: ceny
ARIMA(4,0,2) with non-zero mean

Coefficients:
          ar1      ar2     ar3     ar4     ma1     ma2    mean
      -0.0245  -0.2949  0.7186  0.2095  0.7919  0.8453  1.2135
s.e.   0.0972   0.0787  0.0679  0.0812  0.0822  0.0578  0.0128

sigma^2 = 0.001065:  log likelihood = 547.33
AIC=-1078.66   AICc=-1078.11   BIC=-1049.81

Training set error measures:
                       ME      RMSE        MAE         MPE     MAPE     MASE         ACF1
Training set 0.0006324601 0.0322061 0.02100185 -0.01410137 1.711285 1.011895 -0.006671445
```

```
res2 <- residuals(model2)
Box.test(res2, type = "Ljung-Box")
```

```

	Box-Ljung test

data:  res2
X-squared = 0.01224, df = 1, p-value = 0.9119
```

```
#P-value większe od 0.05 -> brak aurokorelacji
jarque.bera.test(res)
```

```

	Jarque Bera Test

data:  res
X-squared = 2287.2, df = 2, p-value < 2.2e-16
```

```
#P-value mniejsze od 0.05 -> reszty nie mają rozkładu normalnego
```

```
arima_forecast <- forecast(model2,4)$mean
```

Porównanie z odłożonymi wartościami

```
ceny_wyniki <- data.frame(
  Odłożone_Wartości = ceny_oct,
  Początek_Przedziału = forecast(model2,4)$lower[, 2],
  Koniec_Przedziału = forecast(model2,4)$upper[, 2],
  Średnia_Prognoza = arima_forecast
)
print(ceny_wyniki)
```

	Odłożone_Wartości	Początek_Przedziału	Koniec_Przedziału	Średnia_Prognoza
	<dbl>	<dbl>	<dbl>	<dbl>
273	1.311032	1.248221	1.376123	1.312172
274	1.311032	1.204724	1.365944	1.285334
275	1.311032	1.197584	1.372553	1.285068
276	1.308333	1.193663	1.379063	1.286363

4 rows

Dla każdego z analizowanych notowań, wartości dolnej i górnej granicy przedziałów ufności są dość bliskie średniej prognozy, co wskazuje na umiarkowaną zmienność prognoz. Różnice między wartościami prognozowanymi a rzeczywistymi są niewielkie, co może sugerować dobrą jakość modelu w przewidywaniu przyszłych cen.

Analogicznie dla cen niezlogarytmowanych

```
ceny_n_wyniki <- data.frame(
  Odłożone_Wartości = ceny_n,
  Początek_Przedziału = exp(forecast(model2,4)$lower[, 2]),
  Koniec_Przedziału = exp(forecast(model2,4)$upper[, 2]),
  Średnia_Prognoza = exp(arima_forecast)
)
print(ceny_n_wyniki)
```

	Odłożone_Wartości	Początek_Przedziału	Koniec_Przedziału	Średnia_Prognoza
	<dbl>	<dbl>	<dbl>	<dbl>
273	3.71	3.484139	3.959521	3.714232
274	3.71	3.335839	3.919420	3.615875
275	3.71	3.312104	3.945411	3.614915
276	3.70	3.299144	3.971179	3.619598

4 rows

W porównaniu do wcześniejszych wyników dotyczących logarytmicznych stóp zwrotu, które wskazywały na większą zmienność i niepewność prognoz, wyniki dla cen niezlogarytmowanych wydają się bardziej spójne. Średnie prognozy cen są niewiele różne od rzeczywistych wartości, co sugeruje, że model skutecznie uchwycił ogólne trendy cenowe w krótkim okresie.

Metody symulacyjne

```
# Liczba symulacji
n <- 1000
# Długość prognozy (4 notowania)
h <- 4
```

Wartość początkowa logarytmów cen (ostatnia znana wartość w cenach)

```
last_price <- as.numeric(tail(ceny, 1))
```

Symulacje Monte Carlo

```
set.seed(123)  # Ustawienie ziarna dla powtarzalności wyników
symulacje <- matrix(0, nrow = n, ncol = h)
for (i in 1:n) {
  # Generowanie losowych reszt modelu
  sim_res <- rnorm(h, mean(res2), sd(res2))
  # Symulacja logarytmów cen przy użyciu modelu
  symulacje[i, 1] <- last_price + sim_res[1]
  for (j in 2:h) {
    symulacje[i, j] <- (symulacje[i, j-1] + sim_res[j])
  }
}
```

Obliczenie 95% przedziałów ufności dla logarytmów cen

```
log_price_forecast_mean_mc <- colMeans(symulacje)
log_price_forecast_lower <- apply(symulacje, 2, quantile, probs = 0.025)
log_price_forecast_upper <- apply(symulacje, 2, quantile, probs = 0.975)
```

Konwersja logarytmów cen do rzeczywistych cen

```
symulacje_ceny <- exp(symulacje)
```

Obliczenie 95% przedziałów ufności dla rzeczywistych cen

```
price_forecast_mean <- colMeans(symulacje_ceny)
price_forecast_lower <- apply(symulacje_ceny, 2, quantile, probs = 0.025)
price_forecast_upper <- apply(symulacje_ceny, 2, quantile, probs = 0.975)
```

Tworzenie tabeli porównawczej wyników

```
wyniki_monte_carlo <- data.frame(
  Odłożone_Wartości_Log = ceny_oct,
  Prognoza_Log_Ceny = log_price_forecast_mean_mc,
  Przedzial_Log_Ceny_Dolny = log_price_forecast_lower,
  Przedzial_Log_Ceny_Gorny = log_price_forecast_upper,
  Odłożone_Wartości = exp(ceny_oct),
  Prognoza_Ceny = price_forecast_mean,
  Przedzial_Ceny_Dolny = price_forecast_lower,
  Przedzial_Ceny_Gorny = price_forecast_upper
)
print(wyniki_monte_carlo)
```

	Odłożone_Wartości_Log	Prognoza_Log_Ceny	Przedzial_Log_Ceny_Dolny	Przedzial_Log_Ceny_Gorny
	<dbl>	<dbl>	<dbl>	<dbl>
273	1.311032	1.333292	1.268747	1.395616
274	1.311032	1.333502	1.249373	1.416259
275	1.311032	1.334588	1.232216	1.438986
276	1.308333	1.335842	1.216895	1.456054

4 rows | 1-5 of 8 columns

W kontekście wcześniejszych wyników z modelu ARIMA, które analizowano w odniesieniu do stóp zwrotu oraz cen niezlogarytmowanych, wyniki uzyskane z metody Monte Carlo wydają się bardziej stabilne. Poprzednie analizy wskazywały na większą zmienność prognoz i szersze przedziały ufności, szczególnie dla logarytmicznych stóp zwrotu. Zastosowanie metody Monte Carlo może zatem dostarczyć lepszych prognoz, a także umożliwia bardziej precyzyjne oszacowanie ryzyka.

Metody bootstrapowe

```
set.seed(123)  # Ustawienie ziarna dla powtarzalności wyników
bootstrap <- matrix(0, nrow = n, ncol = h)
for (i in 1:n) {
  # Generowanie losowych reszt modelu
  boot_res <- sample(res2, h, replace = TRUE)
  # Symulacja logarytmów cen przy użyciu modelu
  bootstrap[i, 1] <- last_price + boot_res[1]
  for (j in 2:h) {
    bootstrap[i, j] <- (bootstrap[i, j-1] + boot_res[j])
  }
}
```

Obliczanie średnich prognoz i 95% przedziałów ufności dla logarytmów cen

```
log_price_forecast_mean_bootstrap <- colMeans(bootstrap)
log_price_forecast_lower <- apply(bootstrap, 2, quantile, probs = 0.025)
log_price_forecast_upper <- apply(bootstrap, 2, quantile, probs = 0.975)
```

Konwersja logarytmów cen na rzeczywiste ceny

```
bootstrap_prices <- exp(bootstrap)
```

Obliczanie średnich prognoz i 95% przedziałów ufności dla rzeczywistych cen

```
price_forecast_mean <- colMeans(bootstrap_prices)
price_forecast_lower <- apply(bootstrap_prices, 2, quantile, probs = 0.025)
price_forecast_upper <- apply(bootstrap_prices, 2, quantile, probs = 0.975)
```

Tworzenie tabeli wynikowej z prognozami i przedziałami ufności

```
wyniki_bootstrap <- data.frame(
  Odłożone_Wartości_Log = ceny_oct,
  Prognoza_Log_Ceny = log_price_forecast_mean_bootstrap,
  Przedzial_Log_Ceny_Dolny = log_price_forecast_lower,
  Przedzial_Log_Ceny_Gorny = log_price_forecast_upper,
  Odłożone_Wartości = exp(ceny_oct),
  Prognoza_Ceny = price_forecast_mean,
  Przedzial_Ceny_Dolny = price_forecast_lower,
  Przedzial_Ceny_Gorny = price_forecast_upper
)
print(wyniki_bootstrap)
```

	Odłożone_Wartości_Log	Prognoza_Log_Ceny	Przedzial_Log_Ceny_Dolny	Przedzial_Log_Ceny_Gorny
	<dbl>	<dbl>	<dbl>	<dbl>
273	1.311032	1.333262	1.279025	1.403410
274	1.311032	1.333375	1.249733	1.433881
275	1.311032	1.333342	1.238401	1.472631
276	1.308333	1.334226	1.227396	1.485462

4 rows | 1-5 of 8 columns

Dla logarytmicznych cen, przedziały ufności są nieco węższe w analizie bootstrapowej w porównaniu do wyników z Monte Carlo, co sugeruje mniejszą niepewność w prognozach. Podobnie, prognozy cen niezlogarytmowanych są zbieżne z rzeczywistymi wartościami, a wąskie przedziały ufności mogą wskazywać na stabilność modelu.

Porównanie obu metod

```
wyniki_porownawcze <- data.frame(
  Odlozone_Wartosci = ceny_oct,
  Prognoza_Monte_Carlo = log_price_forecast_mean_mc,
  Prognoza_Bootstrap = log_price_forecast_mean_bootstrap
)
print(wyniki_porownawcze)
```

	Odlozone_Wartosci	Prognoza_Monte_Carlo	Prognoza_Bootstrap
	<dbl>	<dbl>	<dbl>
273	1.311032	1.333292	1.333262
274	1.311032	1.333502	1.333375
275	1.311032	1.334588	1.333342
276	1.308333	1.335842	1.334226

4 rows

Wartości prognozowane zarówno przez metodę Monte Carlo, jak i bootstrap są bardzo zbliżone do siebie oraz do rzeczywistych odłożonych wartości.
Różnice pomiędzy prognozami z obu metod są minimalne, co może sugerować, że obie podejścia oferują podobny poziom dokładności w prognozowaniu logarytmicznych cen. Wartości prognoz są zaledwie w niektórych przypadkach różne o kilka punktów dziesiętnych, co nie wpływa istotnie na ogólną interpretację.
Prognozy z obu metod mieszczą się w wąskich granicach, co świadczy o stabilności prognoz i sugeruje, że zastosowane modele są odpowiednie do przewidywania przyszłych cen.

Efekt ARCH

```
spec <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
  mean.model = list(armaOrder = c(3, 2), include.mean = TRUE),
  distribution.model = "norm"
)
model_garch <- ugarchfit(spec = spec, data = ceny)
print(model_garch)
```

```

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model : sGARCH(1,1)
Mean Model  : ARFIMA(3,0,2)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error     t value Pr(>|t|)
mu      1.252308    0.000215  5.8266e+03 0.000000
ar1     1.805804    0.000311  5.8102e+03 0.000000
ar2    -0.668964    0.000121 -5.5394e+03 0.000000
ar3    -0.136679    0.000032 -4.2412e+03 0.000000
ma1    -1.410788    0.000296 -4.7604e+03 0.000000
ma2     0.394894    0.000191  2.0645e+03 0.000000
omega   0.000391    0.000057  6.8548e+00 0.000000
alpha1  0.502986    0.154055  3.2650e+00 0.001095
beta1   0.005690    0.058779  9.6811e-02 0.922876

Robust Standard Errors:
        Estimate  Std. Error     t value Pr(>|t|)
mu      1.252308    0.038882   32.207744  0.00000
ar1     1.805804    0.050632   35.664958  0.00000
ar2    -0.668964    0.001286 -520.055103  0.00000
ar3    -0.136679    0.002597  -52.628991  0.00000
ma1    -1.410788    0.032163  -43.863943  0.00000
ma2     0.394894    0.015359   25.710219  0.00000
omega   0.000391    0.001191    0.328449  0.74257
alpha1  0.502986   13.082246    0.038448  0.96933
beta1   0.005690    2.693861    0.002112  0.99831

LogLikelihood : 571.605

Information Criteria
------------------------------------

Akaike       -4.1368
Bayes        -4.0175
Shibata      -4.1389
Hannan-Quinn -4.0889

Weighted Ljung-Box Test on Standardized Residuals
------------------------------------
                         statistic   p-value
Lag[1]                       2.045 1.527e-01
Lag[2*(p+q)+(p+q)-1][14]    12.554 6.429e-13
Lag[4*(p+q)+(p+q)-1][24]    18.871 1.557e-02
d.o.f=5
H0 : No serial correlation

Weighted Ljung-Box Test on Standardized Squared Residuals
------------------------------------
                        statistic   p-value
Lag[1]                      1.098 2.946e-01
Lag[2*(p+q)+(p+q)-1][5]    26.545 2.954e-07
Lag[4*(p+q)+(p+q)-1][9]    29.919 4.593e-07
d.o.f=2

Weighted ARCH LM Tests
------------------------------------
            Statistic Shape Scale P-Value
ARCH Lag[3]    0.6316 0.500 2.000  0.4268
ARCH Lag[5]    1.4212 1.440 1.667  0.6133
ARCH Lag[7]    1.5819 2.315 1.543  0.8048

Nyblom stability test
------------------------------------
Joint Statistic:  2.0631
Individual Statistics:
mu     0.09232
ar1    0.04595
ar2    0.09159
ar3    0.09226
ma1    0.08374
ma2    0.09646
omega  0.33764
alpha1 0.15111
beta1  0.13335

Asymptotic Critical Values (10% 5% 1%)
Joint Statistic:         2.1 2.32 2.82
Individual Statistic:    0.35 0.47 0.75

Sign Bias Test
------------------------------------
```

	t-value	prob	sig
	<dbl>	<dbl>	<chr>
Sign Bias	0.2940428	0.7689536	
Negative Sign Bias	1.0317492	0.3031233	
Positive Sign Bias	1.6400549	0.1021713	
Joint Effect	3.7542887	0.2892481	

4 rows

```


Adjusted Pearson Goodness-of-Fit Test:
------------------------------------
  group statistic p-value(g-1)
1    20     48.88    0.0001910
2    30     63.07    0.0002523
3    40     66.82    0.0036394
4    50     76.53    0.0071724


Elapsed time : 0.3995869
```

Testy Ljung-Boxa wykazują pewien stopień autokorelacji w resztach, co może sugerować niedopasowanie modelu, ale testy ARCH nie wykazują istotnych oznak autokorelacji, co potwierdza dobre dopasowanie.

Prognozy modelu ARMA-GARCH

```
forecast_garch <- ugarchforecast(model_garch, n.ahead = h)
print(forecast_garch)
```

```

*------------------------------------*
*       GARCH Model Forecast         *
*------------------------------------*
Model: sGARCH
Horizon: 4
Roll Steps: 0
Out of Sample: 0

0-roll forecast [T0=1970-09-30]:
    Series   Sigma
T+1  1.313 0.02906
T+2  1.314 0.02865
T+3  1.313 0.02844
T+4  1.312 0.02833
```

```
log_garch_mean <- fitted(forecast_garch)
print(log_garch_mean)
```

```
    1970-09-30
T+1   1.312969
T+2   1.314312
T+3   1.312753
T+4   1.311691
```

Reszty z modelu GARCH

```
res3 <- residuals(model_garch)
```

```
set.seed(123)
mcg <- matrix(0, nrow = n, ncol = h)
for (i in 1:n) {
  # Generowanie losowych reszt modelu
  sim_res <- rnorm(h, mean(res3), sd(res3))
  # Symulacja logarytmów cen przy użyciu modelu
  mcg[i, 1] <- last_price + sim_res[1]
  for (j in 2:h) {
    mcg[i, j] <- (mcg[i, j-1] + sim_res[j])
  }
}
```

```
set.seed(123)
bootg <- matrix(0, nrow = n, ncol = h)
for (i in 1:n) {
  # Generowanie losowych reszt modelu
  boot_res <- sample(res3, h, replace = TRUE)
  # Symulacja logarytmów cen przy użyciu modelu
  bootg[i, 1] <- last_price + boot_res[1]
  for (j in 2:h) {
    bootg[i, j] <- (bootg[i, j-1] + boot_res[j])
  }
}
```

```
log_gmc_mean <- colMeans(mcg)
gmc_mean <- colMeans(exp(mcg))
log_bootg_mean <- colMeans(bootg)
bootg_mean <- colMeans(exp(bootg))
```

```
ostatnie_porównanie <- data.frame(
  Odłożone_Wartości = ceny_oct,
  Prognoza_ARIMA = arima_forecast,
  Prognoza_MC = log_price_forecast_mean_mc,
  Prognoza_Bootstrap = log_price_forecast_mean_bootstrap,
  Prognoza_GARCH = unname(log_garch_mean),
  Prognoza_MC_GARCH = log_gmc_mean,
  Prognoza_Bootstrap_GARCH = log_bootg_mean
)
print(ostatnie_porównanie)
```

	Odłożone_Wartości	Prognoza_ARIMA	Prognoza_MC	Prognoza_Bootstrap	Prognoza_GARCH
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
273	1.311032	1.312172	1.333292	1.333262	1.312969
274	1.311032	1.285334	1.333502	1.333375	1.314312
275	1.311032	1.285068	1.334588	1.333342	1.312753
276	1.308333	1.286363	1.335842	1.334226	1.311691

4 rows | 1-6 of 7 columns

Porównanie prognoz: Tabela przedstawia zestawienie prognozowanych wartości uzyskanych z różnych modeli: ARIMA, Monte Carlo, Bootstrap oraz GARCH. Prognozy ARIMA są w dużej mierze zbliżone do prognoz GARCH. **Różnice między modelami:** Prognozy oparte na metodach Monte Carlo i Bootstrap w większości przypadków przewidują nieco wyższe wartości niż te uzyskane z modeli ARIMA i GARCH.

Przeprowadzenie testu ARCH

```
test_arch1 <- ArchTest(res2)
print(test_arch1)
```

```

	ARCH LM-test; Null hypothesis: no ARCH effects

data:  res2
Chi-squared = 5.6088, df = 12, p-value = 0.9345
```

Hipoteza zerowa testu stwierdza, że nie występują efekty ARCH, co oznacza, że wariancja reszt jest stała w czasie.
Wysoka p-wartość (0.9345) sugeruje, że nie ma wystarczających dowodów, aby odrzucić hipotezę zerową. Oznacza to, że w analizowanych resztach nie ma istotnych efektów ARCH, co wskazuje na stabilność wariancji reszt.

```
test_arch2 <- ArchTest(residuals(model_garch))
print(test_arch2)
```

```

	ARCH LM-test; Null hypothesis: no ARCH effects

data:  residuals(model_garch)
Chi-squared = 25.092, df = 12, p-value = 0.01439
```

Test ARCH sprawdza hipotezę zerową, że nie występują efekty ARCH w resztach modelu, co oznacza, że wariancja reszt jest stała w czasie. Wartość p jest znacznie poniżej standardowego poziomu istotności (0.05). Oznacza to, że istnieją silne dowody przeciwko hipotezie zerowej. W związku z tym możemy stwierdzić, że w resztach modelu GARCH występują istotne efekty ARCH.

Podstawowy model nie wykazuje efektu ARCH, jednak przeprowadzając go na modelu GARCH uzyskujemy odwrotny wynik.

1. Badanie stacjonarności i logarytmicznych stóp zwrotu
Wstępna analiza stacjonarności została przeprowadzona za pomocą testu ADF (Augmented Dickey-Fuller). Wyniki wykazały stacjonarność stóp zwrotu, co było kluczowe do przeprowadzenia dalszej analizy czasowej. Osiągnięcie stacjonarności potwierdza, że dane są odpowiednie do modelowania przy użyciu metod ARIMA/ARMA, pozwalając na skuteczniejsze przewidywanie przyszłych zmian cen na podstawie historycznych notowań.Dopasowanie modelu ARIMA Na podstawie wyników testu ADF dobrano model ARIMA, analizując autokorelację składników losowych i weryfikując istotność parametrów przy najwyższych opóźnieniach. Model ARIMA, uwzględniając różne opóźnienia i struktury, umożliwił przewidywanie cen i logarytmicznych stóp zwrotu na podstawie przeszłych danych.

2. Ceny na 4 przyszłe notowania i porównanie z danymi rzeczywistymi
Użycie modelu ARIMA pozwoliło na wygenerowanie prognoz na kolejne cztery notowania. Wartości prognozowane dla cen, logarytmów cen i stóp zwrotu zostały porównane z rzeczywistymi wartościami, a 95% przedział ufności pozwolił na ocenę dokładności modelu. Okazało się, że model jest adekwatny, choć nieco odbiega w niektórych notowaniach, co może wynikać z dynamicznej natury rynków finansowych.

3. Symulacje Monte Carlo
Aby uzyskać dodatkowe prognozy, wykorzystano metodę Monte Carlo. Wyniki tej metody wskazały na szeroki zakres potencjalnych wyników, a przeprowadzone symulacje pozwoliły na wyznaczenie alternatywnych prognoz dla logarytmów cen i ich przedziałów ufności. W porównaniu z prognozami uzyskanymi z modelu ARIMA, metoda Monte Carlo charakteryzowała się wyższą niepewnością, lecz równocześnie dostarczała bardziej zróżnicowany obraz możliwych scenariuszy rynkowych.

4. Prognozy za pomocą metody bootstrapowej
Alternatywnie, zastosowano metodę bootstrapową do wyznaczenia prognoz logarytmów cen oraz samych cen. Wyniki były zbliżone do tych uzyskanych metodą Monte Carlo, jednak bootstrap generował nieco węższe przedziały ufności, co sugeruje większą stabilność prognoz w tej metodzie. Porównanie z prognozami z poprzednich punktów wykazało, że bootstrap generuje bardziej konserwatywne prognozy w stosunku do symulacji Monte Carlo.

5. Uwzględnienie efektu ARCH w modelu ARMA-GARCH
Efekt ARCH został zbadany przy użyciu testu ARCH, po czym zastosowano model ARMA-GARCH, który uwzględnia zmienność warunkową. Model ARMA-GARCH dodatkowo uwzględnia wahania wariancji, co poprawiło dokładność prognoz na dynamicznym rynku. Wyniki wykazały, że model lepiej odzwierciedla rzeczywiste zmiany rynkowe, co sugeruje, że uwzględnienie zmienności warunkowej jest istotne w prognozowaniu na rynku finansowym.

6. Podsumowanie
Podsumowując, analiza wykazała, że model ARIMA jest adekwatny do przewidywania cen na rynku akcji. Jednak uwzględnienie modeli GARCH oraz zastosowanie symulacji Monte Carlo i metod bootstrapowych pozwoliło na uzyskanie pełniejszego obrazu ryzyka i potencjalnych wahań cen. Połączenie tych metod dostarcza szerokiego wglądu w przyszłe notowania i umożliwia lepszą ocenę potencjalnego ryzyka.