ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет экономических наук

Образовательная программа «Экономика»

КУРСОВАЯ РАБОТА

«Сравнение алгоритмов прогнозирования рядов на базе M4»

Выполнила:

Студентка группы БЭК161
Волкова Анастасия Эдуардовна

Научный руководитель:

старший преподаватель
департамента прикладной экономики
Демешев Борис Борисович

Москва 2019

Содержание

1 Введение

В работе представлен обзор нового метода прогнозирования **M4metalearning**, занявшего второе место в соревнованиях **M4 Competition** в 2018 году.

Данный метод был разработан исследователем Пабло Монтеро-Мансо из университета Испании Ла-Корунья и его австралийскими коллегами из университета Монаша, включая известного статистика Роба Хиндмана.

Новизна алгоритма состоит в линейном комбинировании прогнозов 9 стандартных методов, веса для которого вычисляются с помощью градиентного бустинга решающих деревьев.

Комбинация составляется из 9 базовых методов прогнозирования:

- ARIMA модель, параметры p, d и q настраиваются автоматически.
- ETS модель, параметры настраиваются автоматически.
- NNAR. Нейронная сеть прямого распространения с одним скрытым слоем, обучающаяся на лагах. Количество лагов настраивается автоматически.
- TBATS. Модель включает в себя тригонометрические методы для распознования сезонности, которая может менятся со временем, трансформацию Бокса-Кокса от гетероскедастичности, ARMA ошибки, тренд и сезонность подобно ETS. Параметры настраиваются автоматически.
- STLM-AR. Метод сезонной трансформации с помощью LOESS с AR моделированием очищенных от сезонности рядов.
- THETA. Тета-метод выиграл M3 Forecasting Competition в 2000 году.
- Случайное блуждание. Наивный прогноз с помощью последнего наблюдения.
- Модель случайного блуждания со сдвигом. Частный случай ARIMA(0,1,0) с константой.
- Наивный прогноз с учетом сезонности.

В данной работе рассматриваемый метод с использованием предобученной модели был протестирован на данных из набора M4 и базе данных российских макроэкономических рядов **sophisthse**. Сравнение качества по MAPE показало, что исследуемый метод почти всегда опережает автонастраиваемые модели ARIMA и ETS. Кроме того, модель была обучена на части рядов из **sophisthse** и протестирована на остальных.

Все действия, к которым написан код, реализованы на языке программирования R, список необходимых пакетов и правила их установки можно найти в приложении.

2 Описание методов прогнозирования

Проведем краткий обзор методов прогнозирования, используемых в алгоритме M4metalearning. Для наглядности будем применять их к конкретному временному ряду. Например, возьмем данные по ежемесячным продажам кортикостероидных препаратов H02 в Австралии с 1992 по 2010 год.

```
glimpse(h02)
autoplot(h02) + ylab("млн $") + xlab("Месяц")+
ggtitle("Уровень продаж кортикостероидов в Австралии")
```

Рис. 1: Исходный ряд

Заметим, что ряд нестационарный, присутствует возрастающий тренд, а также небольшой рост дисперсии по мере увеличения уровня ряда. Данную проблему можно исправить с помощью логарифмирования. Также присутствует сезонность, посмотрим на нее в полярных координатах.

```
ggseasonplot(h02, polar = TRUE) + ylab("млн $") + xlab("Месяц")
```

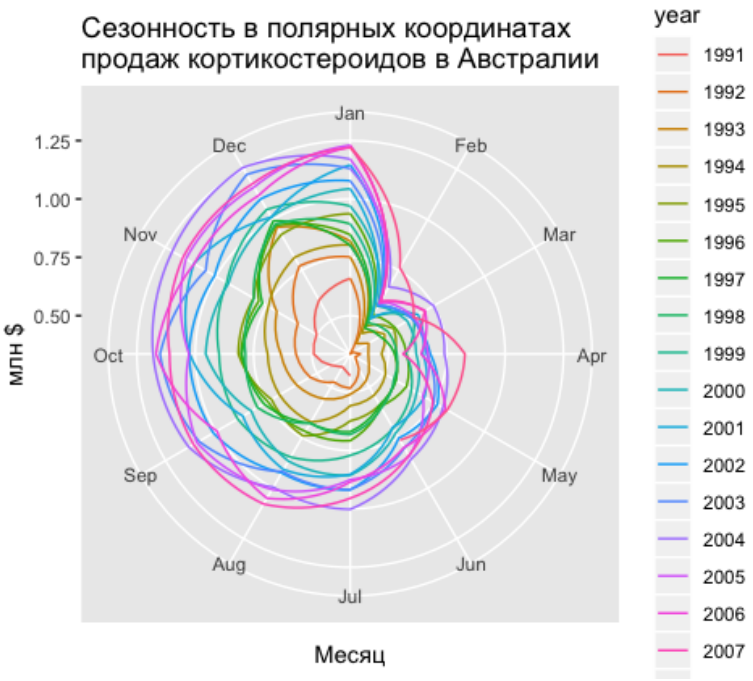


Рис. 2: Сезонность продаж кортикостероидов в Австралии в полярных координатах

Наличие сезонности, а именно ежегодно самый низкий уровень продаж в феврале можно отчасти попытаться объяснить как спецификой препарата, так и тем фактом, что в Австралии январь – последний теплый месяц и на него приходится пик туристического сезона. С февраля начинается новый учебный год, уровень расходов во всем секторе ритейла падает.

Поделим ряд на обучающую и тестовую выборки:

```
train <- window(h02, start = 1992, end = 2007)
test <- window(h02, start = 2007)
```

2.1 Случайное блуждание

Данный метод прогнозирует по принципу: «показатель будет такой же, как вчера». Часто используется как эталонный тест для сравнения качества.

$$\hat{y}_{T+h|T} = y_T \tag{1}$$

```
# Наивный прогноз
fcst_naive <- naive(train, h = 18)

# Проверим качество
accuracy(fcst_naive$mean, test)
#                    ME      RMSE       MAE       MPE     MAPE      ACF1
# Test set -0.3363762 0.3968374 0.3363762 -46.15849 46.15849 0.6911659
```

```
# Построим график
autoplot(fcst_naive) + xlab("Дата") + ylab("Объем продаж, млн $") +
ggtitle("Прогноз объема продаж кортикостероидов в Австралии с помощью Naive")
```

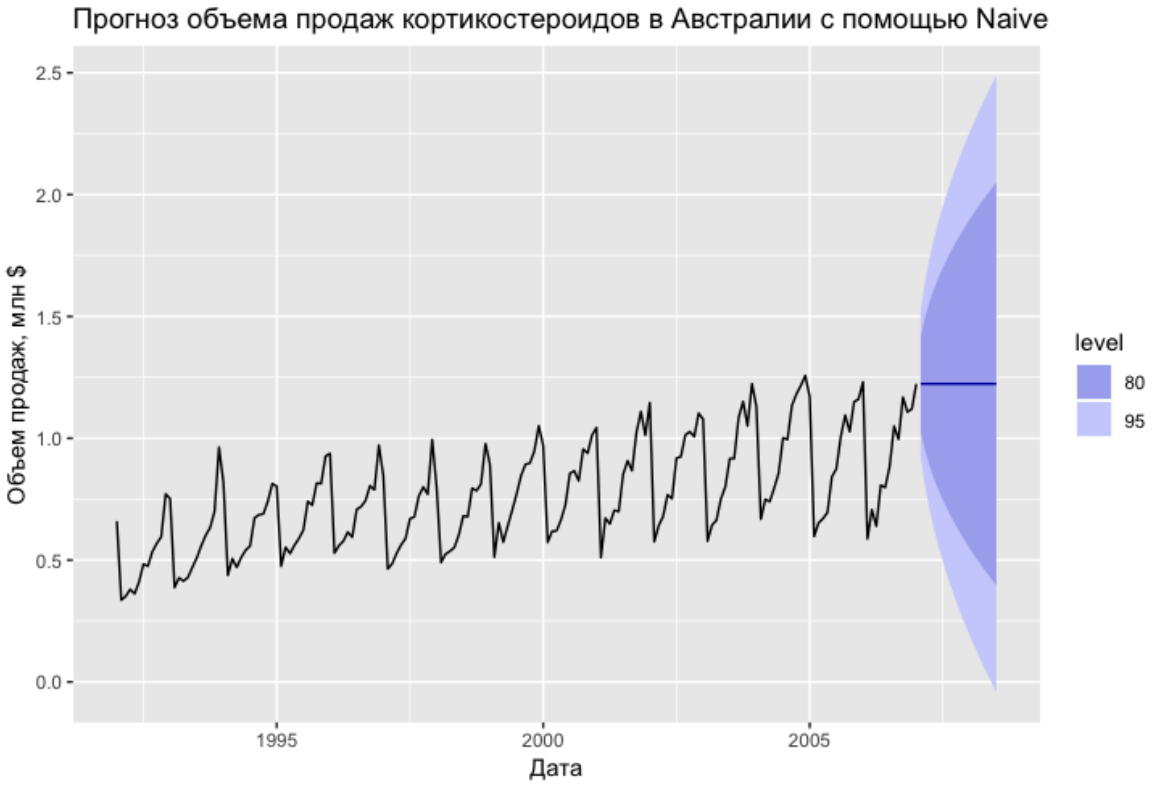


Рис. 3: Наивный метод

2.2 Наивный прогноз с учетом сезонности

Модификация наивного метода для данных с сезонностью. Работает по принципу: «показатель будет таким же, как в прошлом сезоне», в нашем случае «как в прошлом году»

$$\hat{y}_{T+h|T} = y_{T+h-m(k+1)}$$

```
# Наивный прогноз с учетом сезонности
fcst_s_naive <- snaive(train, h = 18)

# Проверим качество
accuracy(fcst_s_naive$mean, test)
                ME       RMSE        MAE      MPE     MAPE        ACF1
Test set 0.02649856 0.08077987 0.06151007 2.444995 7.345278 -0.3012539

# Построим график
autoplot(fcst_s_naive) + xlab("Дата") + ylab("Объем продаж, млн $") +
ggtitle("Прогноз объема продаж кортикостероидов в Австралии с помощью sNaive")
```

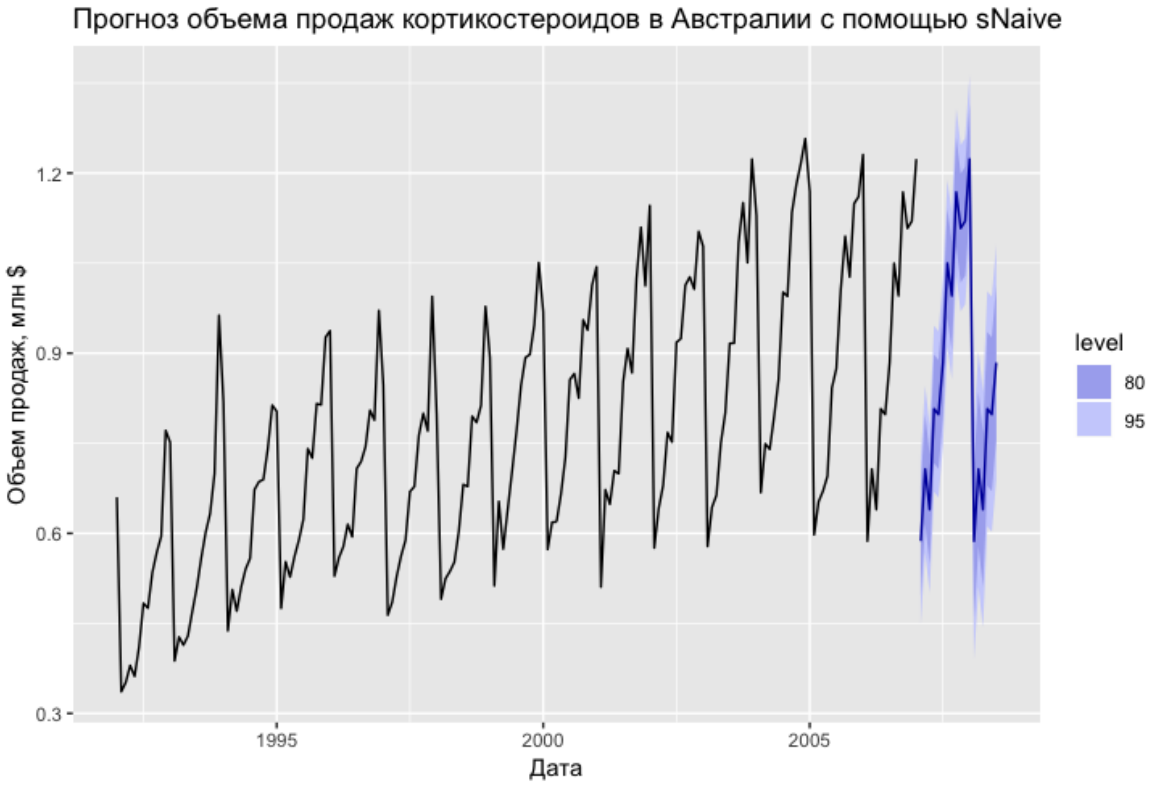


Рис. 4: Наивный метод с учетом сезонности

2.3 ARIMA

AR(Авторегрессия)

Множественная регрессия с последними p наблюдениями в качестве регрессоров.

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + e_t \tag{2}$$

MA(Скользящее среднее)

Множественная регрессия с последними q ошибками в качестве регрессоров.

$$y_t = c + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \tag{3}$$

ARMA процесс

Множественная регрессия с последними p наблюдениями и q ошибками в качестве регрессоров – $ARMA(p,q)$

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t \tag{4}$$

Данная модель работает только для стационарных рядов, поэтому сначала нужно продифференцировать данные, очистив их от тренда и сезонности.

ARIMA – AutoRegressive Integrated Moving Average

В отличии от $ARMA(p,q)$ данная модель включает в себя дополнительный параметр степени дифференцирования исходных данных d для приведения их к стационарному виду и записывается как $ARIMA(p,d,q)$.

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 e_{t-1} + \cdots + \theta_q e_{t-q} + e_t, \tag{5}$$

где y'_t – ряд с применением метода разностей.

```
# ARIMA
model_arima <- auto.arima(train)

summary(model_arima)
# Была выбрана
# Series: train
# ARIMA(1,0,2)(2,1,1)[12] with drift
fcst_arima <- forecast(model_arima, h = 18)

# Проверим качество
accuracy(fcst_arima$mean, test)
#                    ME       RMSE        MAE       MPE     MAPE         ACF1
# Test set 0.02345918 0.07143445 0.05690555 1.939375 6.971999 -0.09930066

# Построим график
fit %>%
  forecast(h = 18) %>%
  autoplot() + xlab("Дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью ARIMA")
```

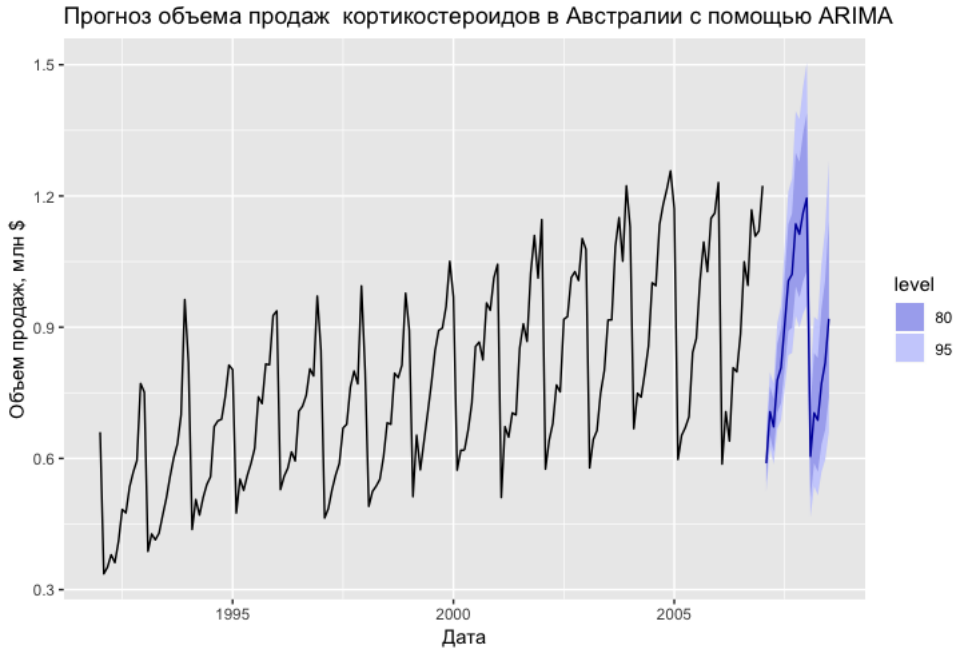


Рис. 5: Arima

2.4 ETS

Прогнозы, полученные с использованием методов экспоненциального сглаживания, представляют собой средневзвешенные значения прошлых наблюдений. Последние наблюдения имеют больший вес. Данная модель быстро строит надежные прогнозы для широкого диапазона временных рядов, что является большим преимуществом для решения различных бизнес-задач.

ETS расшифровывается как:

$$\text{ETS} = \{\text{Error, Trend, Seasonality}\}$$

Каждая составляющая данного метода может быть нескольких видов:

$$\text{Trend} = \{\text{None, Additive, Additive-damped}\}$$

$$\text{Seasonality} = \{\text{None, Additive, Multiplicative}\}$$

$$\text{Error} = \{\text{Additive, Multiplicative}\}$$

Наиболее подходящая модель может подбираться автоматически с помощью команды ets(), которая использует критерий AIC – Akaike's Information Criterion и метод максимального правдоподобия.

```
fcst_ets <- ets(train)
summary(fcst_ets)
# автоматически была подобрана модель ETS(M,A,M)
acc_ets <- forecast(fets, h = 18)

# Проверим качество
accuracy(forets$mean, test)
#                    ME       RMSE        MAE       MPE     MAPE         ACF1
# Test set -0.01814064 0.07401407 0.05845077 -3.146405 7.519454 -0.05680415

# Построим график
autoplot(fcst_ets) + xlab("Дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж кортикостероидов в Австралии с помощью ETS")
```

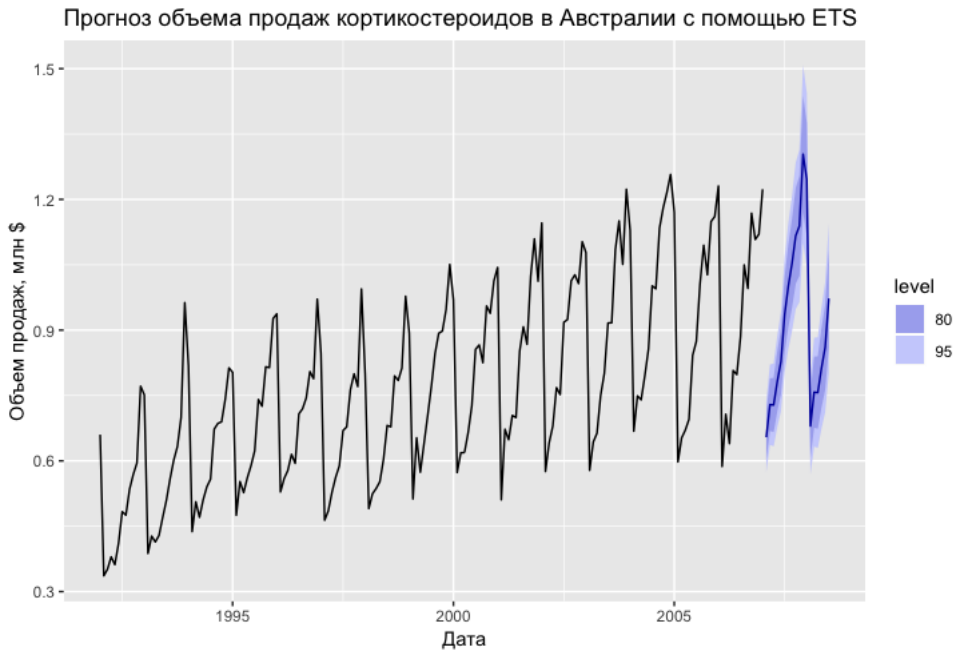
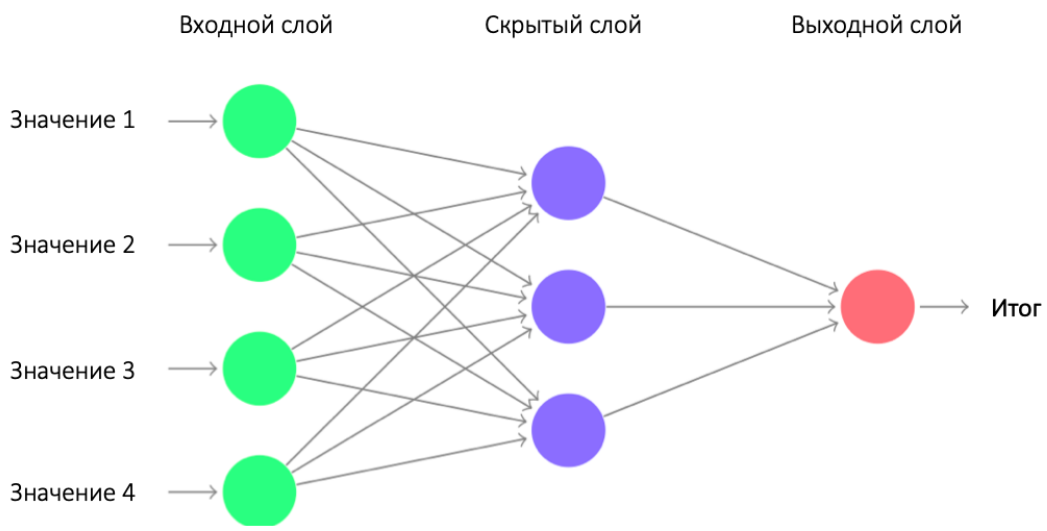


Рис. 6: ETS

2.5 NNETAR

Прямая нейронная сеть с одним скрытым слоем для прогнозирования одномерных временных рядов по их предыдущим значениям. Если представить нейронную сеть без скрытого слоя, она будет эквивалентна обычной линейной регрессии, в которой веса ко входным данным подбираются с помощью МНК. Если в нейронной сети появляется хотя бы один скрытый слой нейронов, то она становится нелинейной. Каждый скрытый нейрон преобразовывает данные с помощью какой-нибудь нелинейной функции, например, сигмоиды.

$$s(z) = \frac{1}{1 + e^{-z}} \tag{6}$$



$NNAR(p,k)$ – Neural Network AutoRegression

Во входной слой подаются p предыдущих значений ряда. Скрытый слой содержит k нейронов. Данная модель является базовой.

Для сезонных данных существует расширение $NNAR(p,P,k)_m$. Такая модель содержит данные за P предыдущих сезонов периодичностью m.

$$(y_{t-1},y_{t-2},\ldots,y_{t-p},y_{t-m},y_{t-2m},y_{t-Pm}) \tag{7}$$

Функция *nnetar()* автоматически подбирает параметры для модели $NNAR(p,P,k)_m$, p используя критерий AIC, число нейронов вычисляется как $k = (p+P+1)/2$, округленное до ближайшего целого числа.

```
#  Нейронная сеть
fcst_nnet <- nnetar_forec(train, h = 18)

# Построим график
autoplot(fcast) + xlab("дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью NNAR")

# Проверим качество
accuracy(fcst_nnet, test)
#                    ME       RMSE        MAE      MPE     MAPE         ACF1
# Test set 0.02170774 0.07206502 0.05740173 1.55178 7.095887 -0.00333016
```

Рис. 7: nnetar

В отличие от линейной авторегрессии, нейронная сеть может распознавать и прогнозировать ассиметричные циклы, это одно из ее преимуществ.

2.6 TBATS

Trigonometric terms for seasonality
Box-Cox transformations for heterogeneity
ARMA errors for short-term dynamics
Trend (possibly damped)
Seasonal (including multiple and non-integer periods)

Данная модель включает в себя множество различных более простых методов: для моделирования сезонности применяется тригонометрия, но в отличие от гармонической регрессии здесь сезонность может менять свою форму во времени, трансформацию Бокса-Кокса, ARMA ошибки для временных колебаний, тренд и уровень ряда подобно модели ETS. Все параметры подбираются автоматически.

$$w_t = \begin{cases} \log(y_t) & \text{если } \lambda = 0; \\ (y_t^\lambda - 1)/\lambda & \text{иначе.} \end{cases}$$

$$
\begin{aligned}
y_t &= \ell_{t-1} + \phi b_{t-1} + \sum_{i=1}^{M} s_{t-m_i}^{(i)} + d_t \\
\ell_t &= \ell_{t-1} + \phi b_{t-1} + \alpha d_t \\
b_t &= (1-\phi)b + \phi b_{t-1} + \beta d_t \\
d_t &= \sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \\
s_t^{(i)} &= \sum_{j=1}^{k_i} s_{j,t}^{(i)} \\
s_{j,t}^{(i)} &= s_{j,t-1}^{(i)} \cos\left(\lambda_j^{(i)}\right) + s_{j,t-1}^{*(i)} \sin\left(\lambda_j^{(i)}\right) + \gamma_1 d_t \\
s_{j,t}^{(i)} &= -s_{j,t-1}^{(i)} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t
\end{aligned}
$$

```r
 # TBATS model
fcst_tbats <- tbats_forec(train, h = 18)

# Проверим качество
accuracy(fcst_tbats, test)
#                    ME       RMSE        MAE       MPE     MAPE         ACF1
# Test set 0.02345918 0.07143445 0.05690555 1.939375 6.971999 -0.09930066

# Построим график
autoplot(fcst_tbats) + xlab("дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью TBATS")
```

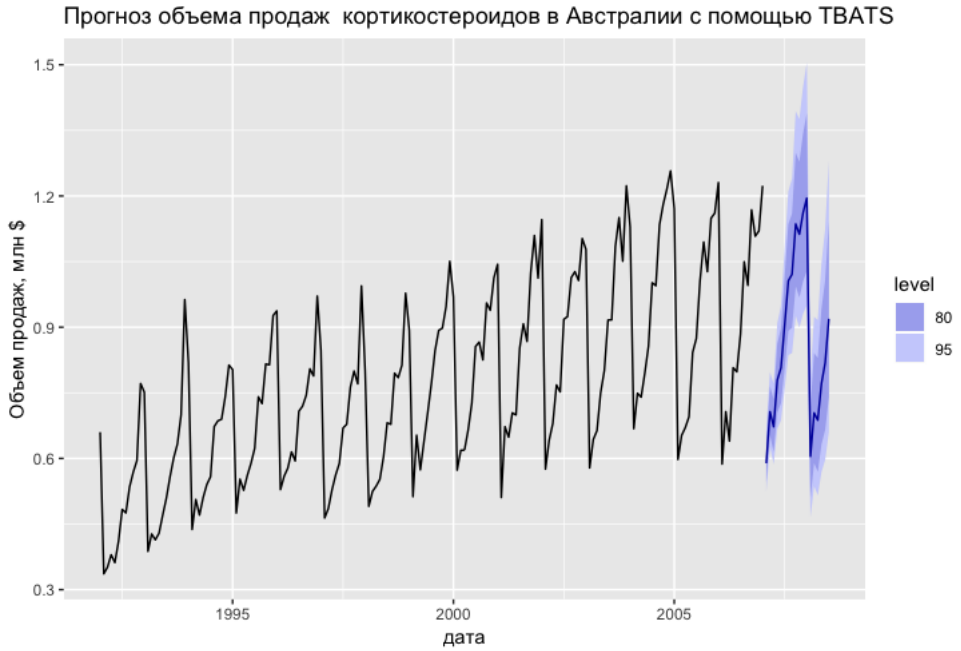


Рис. 8: TBATS

2.7 STLM-AR

Seasonal and
Trend decomposition using
Loess

STL – это универсальный и надежный метод разложения временных рядов, обнаруживающий любой вид сезонности (не только широковстречающиеся квартальные, месячные или годовые).

Предполагается, что $y_t = \hat{S}_t + \hat{T}_t + \hat{R}_t$ (или мультипликативный вид) это декомпозиция ряда, где каждая составляющая прогнозируется отдельно.

Функция stlm() включает в себя корректировку сезонности с помощью метода STL, бессезонную ETS модель, которая затем обратно сезонируется.

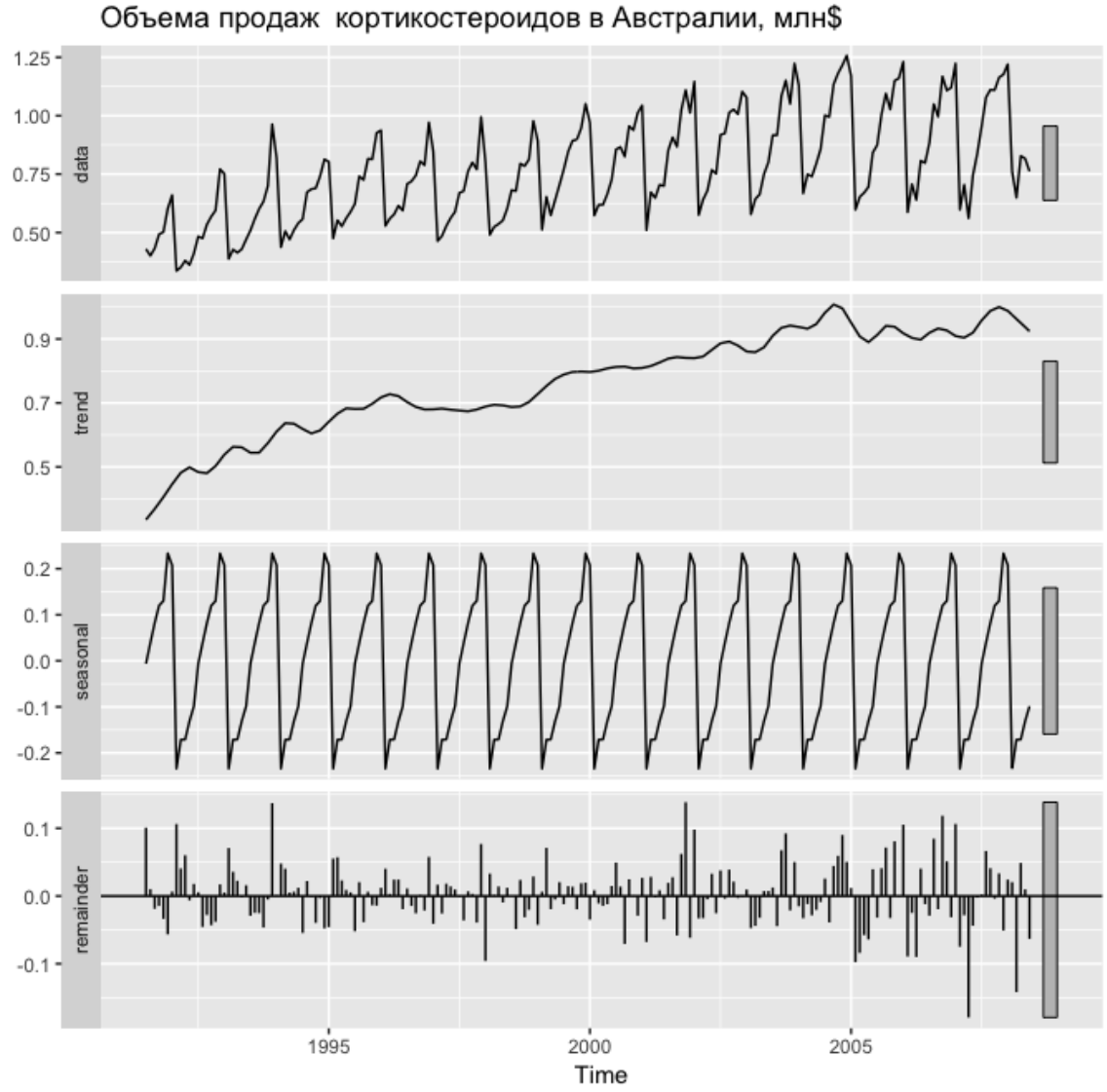


Рис. 9: STL

На рисунке 7 представлена декомпозиция исследуемого ряда. Двумя основ-

ными настраивыми параметрами при использовании STL являются окно тренда-цикла (t.window) и сезонное окно (s.window). Они контролируют, насколько быстро могут меняться трендовый цикл и сезонные компоненты. Меньшие значения допускают более быстрые изменения.

```
# STLM
model_stlm_ar <- stlm(train, modelfunction = ar)
fcst_stlm_ar <- forecast(model_stlm_ar, h = 18)

# Проверим качество
accuracy(fcst_stlm_ar$mean, test)
#                 ME       RMSE        MAE       MPE     MAPE      ACF1
# Test set 0.039341 0.08847129 0.07126412 3.172814 8.480873 0.2734496

# Построим график
autoplot(fcst_stlm_ar) + xlab("дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью STLM")
```

Рис. 10: stlm

2.8 RW-DRIFT

Данную модель можно рассматривать как частный случай $ARIMA(0,1,0)$.

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right)$$

```
# RW DRIFT
fcst_rw_drift <- rwf(train, h = 18, drift = TRUE)

# Проверим качество
accuracy(fcst_rw_drift$mean, test)
#                    ME      RMSE       MAE       MPE      MAPE      ACF1
# Test set -0.3645362 0.4195183 0.3645362 -49.39236 49.39236 0.6877662

# Построим график
autoplot(fcst_rw_drift) + xlab("дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью RW-Drift
```

Рис. 11: RW-Drift

2.9 THETAF

Theta метод применяется к несезонным или очищенным от сезонности временным рядам. Десезонализация обычно выполняется посредством мультипликативного классического разложения.

$$Y''_{t,\theta} = \theta X''_t = \theta(X_t - 2X_{t-1} + X_{t-2}) \tag{8}$$

Решение данного уравнения:

$$Y_{t,\theta} = a_\theta + b_\theta(t-1) + \theta X_t, \tag{9}$$

где a_θ и b_θ являются константами.

Y_t является линейной функцией от X_t с добавлением линии тренда. Параметры a_θ и b_θ подбираются минимизацией суммы квадартов ошибок $\sum_{i=1}^{t} = [X_t - Y_{t,\theta}]^2$

Метод разбивает исходные временные ряды на две новые линии через так называемые тета-коэффициенты, обозначаемые θ_1 и θ_2, которые применяются к данным, подвергнувшихся двукратному взятию разностей. Если θ равно нулю, новая линия является прямой. Когда $\theta > 1$, локальные кривизны увеличиваются, увеличивая кратковременные движения временного ряда. Полученные новые линии называются тета-линиями. Эти линии имеют то же среднее значение и наклон, что и исходные данные, но локальные кривизны либо отфильтровываются, либо усиливаются в зависимости от значения коэффициента θ.

В статье «Unmasking the Theta method» Роба Хиндмана приводится максимально краткое и понятное описание данного метода, а также отмечен факт сходства его прогнозов с прогнозами модели простого экспоненциального сглаживания со сдвигом.

```
# Theta method
fcst_theta <- thetaf(y = train, h = 18)

# Проверим качество
accuracy(fcst_theta$mean, test)
#                    ME       RMSE        MAE        MPE     MAPE        ACF1
# Test set 0.003803136 0.07383127 0.06249971 -0.7075656 7.806133 0.007525667

# Построим график
autoplot(fcst_theta) + xlab("дата") + ylab("Объем продаж, млн $") +
  ggtitle("Прогноз объема продаж  кортикостероидов в Австралии с помощью Theta-me
```

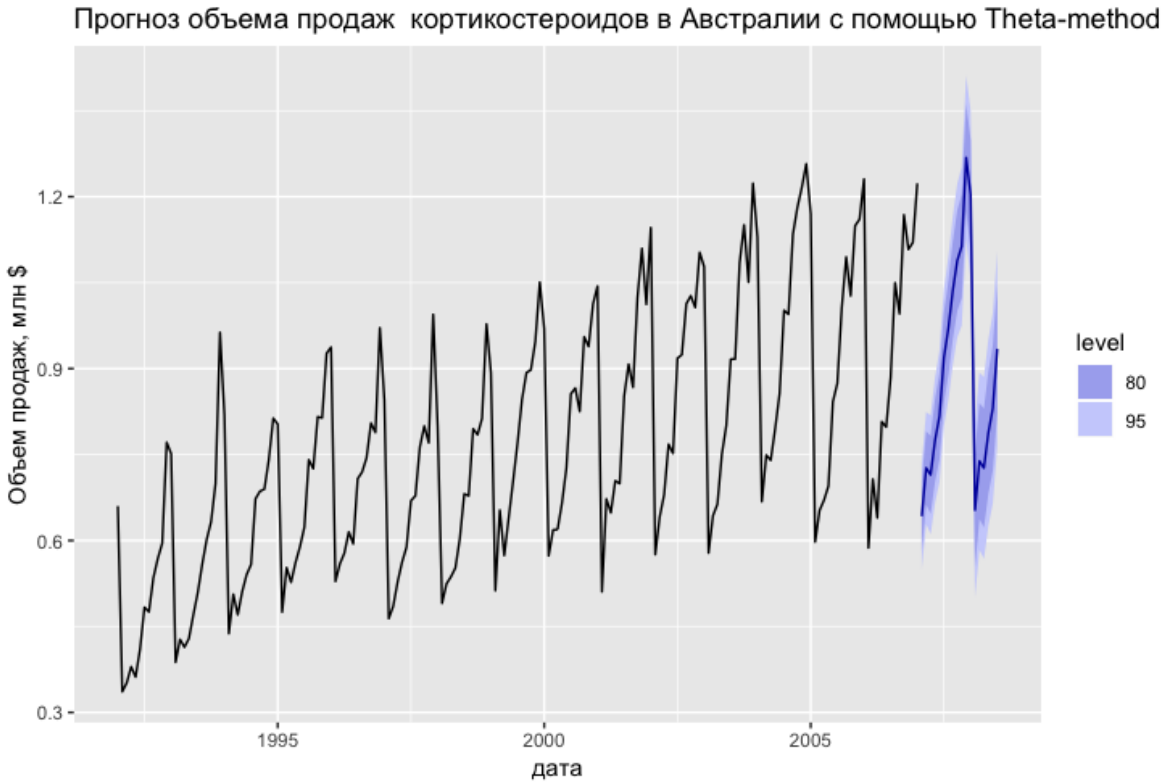


Рис. 12: Theta

Подведем промежуточный итог:

	Метод	MAPE
1	naive	46.16
2	snaive	7.35
3	ets	7.52
4	arima	6.97
5	rw-drift	49.39
6	theta	7.81
7	nnetar	7.1
8	tbats	6.97
9	stlm	8.48

Таблица 1: MAPE прогнозов базовых методов ряда H02

Худшее качество у моделей случайного блуждания, которые не учитывают сезонность. Остальные методы имеют относительно схожее качество. MAPE моделей arima и tbats странным образом оказались одинаковыми и имеют наилучшее качество.

Теперь представим все прогнозы на одном графике:

```
autoplot(h02) +
  autolayer(forets, series="ETS", PI=FALSE) +
```

```
autolayer(for_arima, series="ARIMA", PI=FALSE) +
autolayer(naiv, series="NAIVE", PI=FALSE) +
autolayer(snaiv, series="SNAIV", PI=FALSE) +
autolayer(fcast, series="NNETAR", PI=FALSE) +
autolayer(fc, series="TBATS", PI=FALSE) +
autolayer(forts, series="STLM", PI=FALSE) +
autolayer(rwd, series="RWD", PI=FALSE) +
autolayer(ftheta, series="THETA", PI=FALSE) +
xlab("Год") + ylab("млн$") +
ggtitle("Прогноз объема продаж кортикостероидов в Австралии")
```

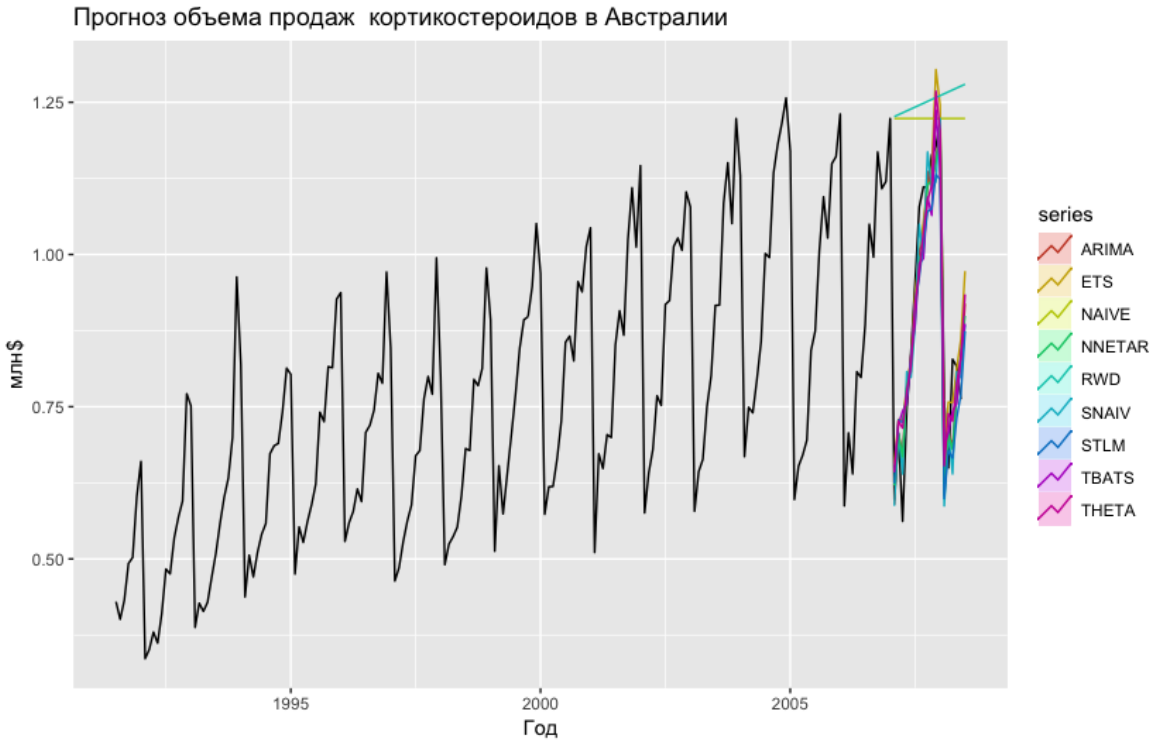


Рис. 13: Прогнозы ряда H02 базовых методов

2.10 Комбинация

Теперь попробуем построить прогноз с помощью исследуемого метода – взвешенной комбинации вышепредставленных алгоритмов.

```
# Combination
forec_result <- forecast_meta_M4(model_M4, train, h=18)
accuracy(f= forec_result$mean, test, test = NULL, d = NULL, D = NULL)
```

```
                    ME      RMSE       MAE       MPE     MAPE         ACF1
Test set -0.005720993 0.2140872 0.1345924 -4.213791 14.78935 -0.06484862
```

```r
# Посмотрим на график
plot(ts(c(train, forec_result$mean),
        start=start(train), frequency = frequency(train)),
     col="red", type="l", ylab="Объем продаж, млн$", xlab = "Год")
lines(train, col="black")
```

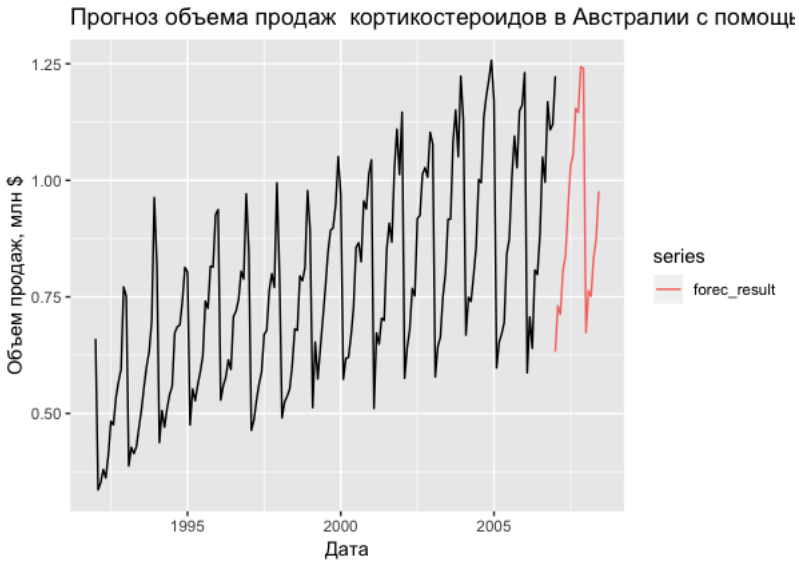


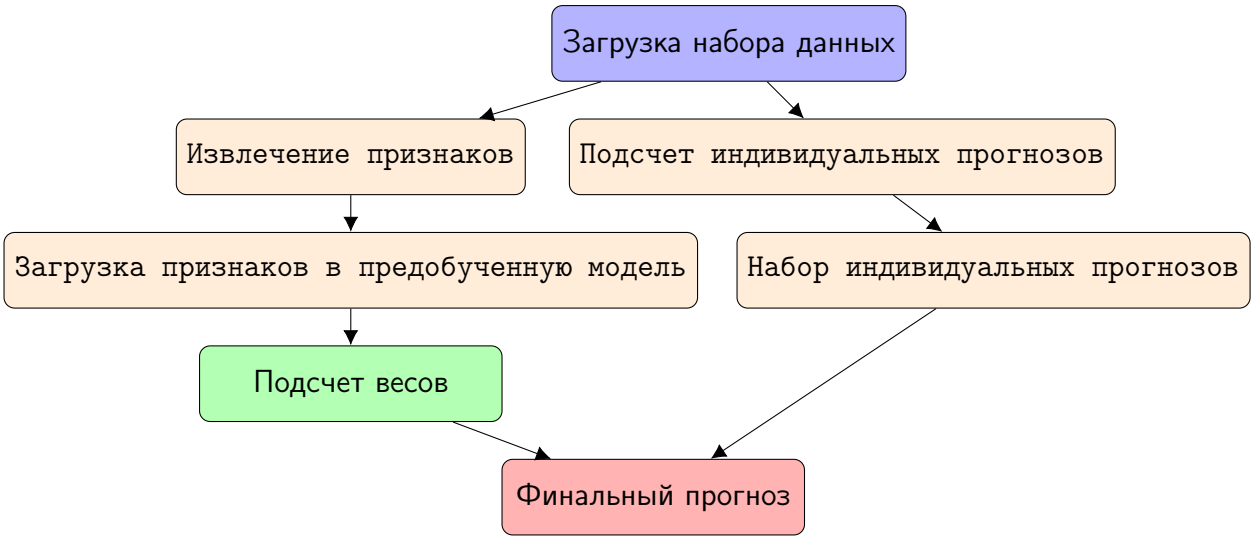
Рис. 14: Прогноз комбинированного метода

3 Алгоритм

Данную модель можно использовать двумя способами. Использовать при прогнозировании предобученную создателями модель, сосредоточенную в пакете M4metaresults или обучать ее заново на данных, которыми Вы располагаете, а затем применять ее. Второй способ занимает много времени и требует достаточного количества данных, поэтому начнем с первого, чтобы познакомиться с тем как работает данная модель.

3.1 Предобученная модель

Рассмотрим каким способом модель вычисляет веса для комбинирования индивидуальных методов прогнозирования.



Модель для получения средних прогнозов:

По сути, это основанный на характеристиках ряда градиентный бустинг решающих деревьев (xgboost), в котором функция потерь, которую нужно минимизировать, адаптируется к ошибке OWA (Overall Weighted Average – средневзвешенное между MASE и sMAPE), используемой в соревнованиях M4.

Модель для создания интервалов прогнозирования:

Данный метод использует средние прогнозы (результат нашего среднего подхода) в качестве центра интервала, и находит линейную комбинацию 95% интервалов прогнозов 3 методов: theta, naive и snaive.

3.2 Извлекаемые признаки

Данная модель испоьзует 42 признака ряда.

1. *x_acf* Коэффициент автокорреляции первого порядка
2. *x_acf10* Сумма квадратов первых десяти автокорреляционных коэффициентов
3. *diff1_acf1* Коэффициент автокорреляции первого порядка для первой разности ряда
4. *diff1_acf10* Сумма квадратов первых десяти автокорреляционных коэффициентов для первой разности ряда
5. *diff2_acf1* Коэффициент автокорреляции первого порядка для второй разности ряда
6. *diff2_acf10* Сумма квадратов первых десяти автокорреляционных коэффициентов для второй разности ряда
7. *seas_acf1* Коэффициент автокорреляции первого сезонного лага. Если сезонности нет, равен 0.

8. *ARCH.LM* Статистика, основанная на LM тесте авторегрессионной условной гетероскедастичности. R^2 авторегрессионной модели 12 лагов применен к x^2 после вычета среднего.

9. *crossing_point* Количество пересечений медианы временным

10. *entropy* Спектральная энтропия ряда. $H_s(x_t) = -\int_{-\Pi}^{\Pi} f_x(\lambda) \log f\_x(\lambda) d\lambda$, где плотность нормализована $\int_{-\pi}^{\pi} f_x(\lambda) d\lambda = 1$

11. *flat_spots* Число плоских участков в ряду, вычисленное путем дискретизации ряда на 10 равных интервалов и подсчета максимальной длины пробега в пределах любого отдельного интервала.

12. *arch_acf* После того, как ряд предварительно приведен к виду белого шума с помощью AR-модели и возведен в квадрат, сумма квадратов первых 12 автокорреляций.

13. *garch_acf* После предварительного приведения ряда к виду белого шума с помощью модели AR к нему применяется модель GARCH(1,1) и вычисляются остатки. Сумма квадратов первых 12 автокорреляция квадратов остатков.

14. *arch_r2* После того, как ряд предварительно очищен с помощью модели AR и возведен в квадрат, R^2 примененной к нему модели AR.

15. *garch_r2* После предварительного приведения ряда к виду белого шума с помощью модели AR к нему применяется модель GARCH(1,1) и вычисляется R^2 примененной к нему модели.

16. *alpha* Параметр сглаживания для уровня в модели ets(A,A,N).

17. *beta* Параметр сглаживания для тренда в модели ets(A,A,N), соответствующего ряда.

18. *hurst* Коэффициент Херста, указывающий уровень дробной разности временных рядов.

19. *lumpiness* Дисперсия дисперсий основана на делении ряда на неперекрывающиеся части. Размер части частота ряда, или 10, если ряд имеет частоту 1.

20. *nonlinearity* Нелинейность статистики на основе теста Terasvirta нелинейности временных рядов.

21. *x_pacf5* Сумма квадратов первых пяти частных автокорреляций ряда

22. *diff1x_pacf5* Сумма квадратов первых пяти частных автокорреляций первой разности ряда.

23. *diff2x_pacf5* Сумма квадратов первых пяти частных автокорреляций второй разности ряда.

24. *seas_pacf* Частная автокорреляция первого сезонного лага. Если сезонности нет, равен 0.

25. *nperiods* Количество сезонных периодов ряда.

26. *seasonal_period* Длина сезонного периода ряда.

27. *trend* В STL разложении ряда с r_t оставшейся части z_t с исключенной сезонностью ряда: max[0,1 − $-Var(rt)/Var(zt)$].

28. *spike* В STL разложении ряда с r_t остаток серии, дисперсия оставить один из дисперсий r_t

29. *linearity* В разложении STL ряда с T_t трендовой составляющей, квадратичная модель в зависимости от времени $T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t$ linearity β_1

30. *curvature* В разложении STL ряда с T_t трендовой составляющей, квадратичная модель в зависимости от времени $T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \epsilon_t$ curvature β_2

31. *e_acf1* Первый коэффициент автокорреляции остатков ряда в STL разложении.

32. *e_acf10* Сумма первых 10 квадратов коэффициентов автокорреляции остатков ряда в STL разложении.

33. *seasonal_strength* В разложении STL ряда, где rt - остатки ряда, xt - ряд, очищенный от тренда: max[0,1 − $-Var(rt)/Var(zt)$].

34. *peak* Расположение пика (максимального значения) в сезонной составляющей и разложение STL ряда.

35. *trough* Расположение минимального значения в сезонной составляющей и разложение STL ряда.

36. *stability* Дисперсия средних основана на делении ряда на неперекрывающиеся части. Длина части - частота ряда, или 10, если ряд имеет частоту 1.

37. *hw_alpha* α параметр модели ets (A,A,A), примененной к ряду.

38. *hw_beta* β параметр модели ets (A,A,A), примененной к ряду.

39. *hw_gamma* γ параметр модели ets (A,A,A), примененной к ряду.

40. *unitroot_kpss* Статистика по Kwiatkowski et al. тест единичного корня с линейным трендом и лагом 1.

41. *unitroot_pp* статистика для Z-alpha версии теста единичного корня Phillips&Perron с постоянным трендом и лагом 1.

42. *series_length* Длина ряда.

3.3 Обучение модели

Чтобы обучить модель самостоятельно, вам понадобится набор данных и список методов прогнозирования, которые вы хотите использовать. Самостоятельно выбираете количество последних элементов ряда, для которых будут строиться прогнозы во время обучения. Чтобы поделить выборку на train и test, достаточно выбрать h и применить функцию temp_holdout().

Далее происходит подсчет прогнозов calc_forecasts() каждого метода из списка forec_methods() (по умолчанию туда включены все методы, описанные ранее в работе, но можно самостоятельно внести изменения в перечень) и их ошибок calc_errors(). Параллельно с помощью функции THA_features() из ряда извлекаются признаки, описанные выше (42 признака). Функция create_feat_classif_problem() упорядочивает методы прогнозирования для каждого ряда в порядке убывания ошибки прогноза, расставляя метки.

Далее модель непосредственно обучается. Мы используем специальный пакет customxgboost, потому что R версия пакета xgboost не позволяет вводить пользовательские функции потерь для многоклассовых задач.

4 Тестирование на наборах данных

4.1 M4

Из набора данных M4 были выбраны по 50 рядов разной частоты: квартальные, месячные и годовые. В M4 данные уже поделены на тренировочную и тестовую выборки, которые обозначаются **x** и **xx** соответственно. Каждый ряд был спрогнозирован тремя методами: ETS, ARIMA и META (в данном случае так обозначен исследуемый метод, комбинирующий алгоритмы). Ошибки MAPE каждого алгоритма усреднялись.

	Type	ETS	ARIMA	META
1	Monthly series	7.42	7.73	7.02
2	Quaterly series	2.72	2.72	2.67
3	Yearly series	21.61	20.73	16.16

Таблица 2: MAPE прогнозов с помощью 3 основных методов

Из таблицы видно, что META стал лучшим во всех категориях, ETS опередил ARIMA на месячных рядах, а ARIMA показала хороший результат на годовых. Интересно, что диапазон значения ошибок не прямопропорционален их частоте.

Возможно, свою роль в победе алгоритма META сыграл тот факт, что представленная создателями модель обучалась на данных, содержащих прогнозируемые ряды. Однако это не отрицает тот факт, что модель, действительно, неплохая и может давать прогнозы, по качеству превосходящие наиболее популярные методы прогнозирования.

Попробуем протестировать алгоритм на временных рядах показателей российской экономики.

4.2 Sophisthse

Протестируем модель на не знакомых ей ранее рядах. Возьмем ряды макроэкономических показателей России из базы данных Sophisthse и модель, предобученную на рядах из M4 . Для загрузки данных существует специальный пакет, инструкцию по установке которго можно найти в приложении. Как и в предыдущем случае отфильтруем ряды по частоте и построим прогнозы тремя основными методами.

	Type	ETS	ARIMA	META
1	Monthly series	14.22	10.85	4.15
2	Quarterly series	13.97	11.36	10.19
3	Yearly series	15.91	12.70	11.86

Таблица 3: MAPE прогнозов sophisthse с помощью 3 основных методов

Из таблицы видно, что META-прогноз показал относительно лучшее качество во всех категориях.

	Type	ETS	ARIMA	META
1	Monthly series	24.8	14.95	5.36
2	Quarterly series	22.13	19.84	23.87
3	Yearly series	45.61	23.16	22.43

Таблица 4: Стандартное отклонение MAPE прогнозов sophisthse с помощью 3 основных методов

Сравним качество прогнозов для двух разных датасетов. Особенно хорошее качество мета-модель показала на месячных рядах Sophisthse, где так же можно заметить, что, чем больше частота ряда, тем хуже качество пронозов по MAPE.

```
accuracy_summ %>%
ggplot(aes(x = period, y = MAPE, label = method, colour = method)) +
facet_wrap(~dataset) +
geom_text() +
scale_colour_brewer(palette = "Set1") +
theme(legend.position = "none") +
labs(x = "", y = "Среднее MAPE", colour = "") +
ggtitle("Сравнение методов ETS, ARIMA и META",
        subtitle = "Наборы данных 'M4' и 'Sophisthse'")
```

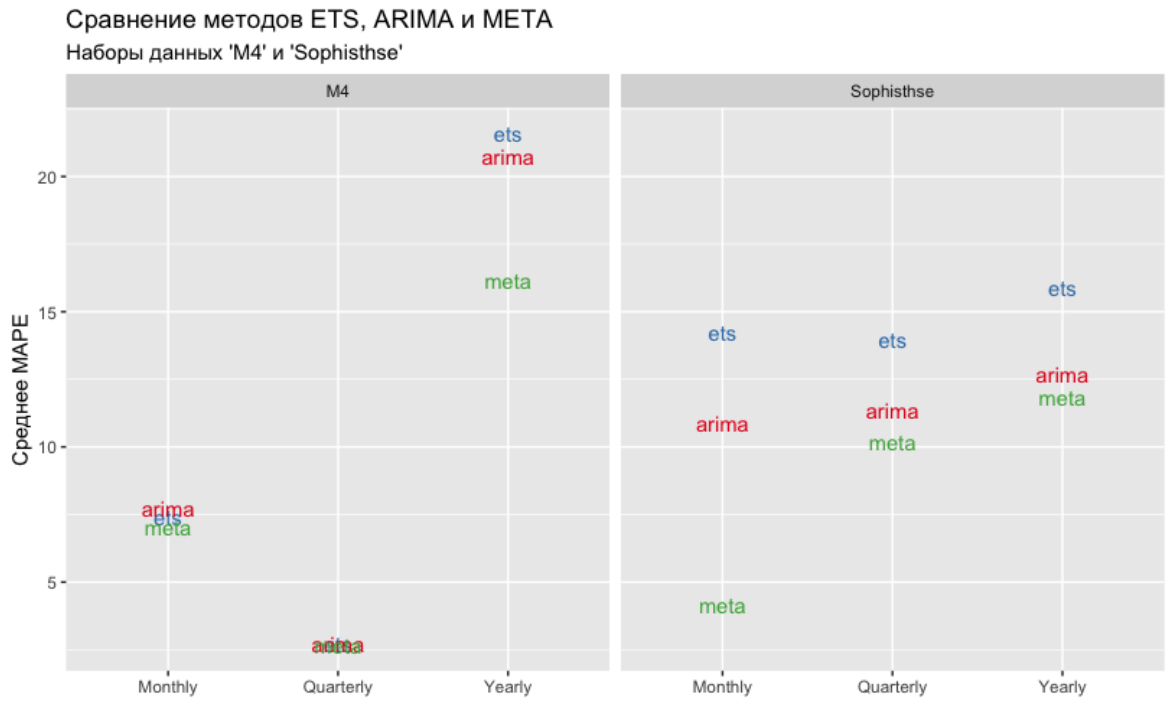


Рис. 15: Сравнение качества прогнозирования

5 Обучение модели

До этого момента мы использовали для прогнозирования предобученную модель M4metaresults, теперь попробуем самостоятельно обучить модель на рядах из Sophisthse. Возьмем только половину месячных рядов, поскольку процессобучения занимает достаточно много времени.

```
# возьмем предварительно подготовленный список рядов data_soph
meta_M4 <- temp_holdout(data_soph)

# Осторожно: это займет много времени
meta_M4 <- calc_forecasts(meta_M4, forec_methods(), n.cores=2)

# Вычисляем ошибки
meta_M4 <- calc_errors(meta_M4)

# Извлекаем признаки
meta_M4 <- THA_features(meta_M4, n.cores=2)

# Подбираем гиперпараметры
hyperparameter_search(meta_M4, filename = "M4_hyper.RData", n_iter=50)

# Выбираем лучшие найденные параметры
```

```
load("M4_hyper.RData")
best_hyper <- bay_results[ which.min(bay_results$combi_OWA), ]

# Обучаем модель на подобранных параметрах

train_data <- create_feat_classif_problem(meta_M4)
param <- list(max_depth=best_hyper$max_depth,
              eta=best_hyper$eta,
              nthread = 3,
              silent=1,
              objective=error_softmax_obj,
              num_class=ncol(train_data$errors),
              subsample=bay_results$subsample,
              colsample_bytree=bay_results$colsample_bytree)

meta_model <- train_selection_ensemble(train_data$data,
                                       train_data$errors,
                                       param=param)
```

Обученная на части данных из Sophisthse модель meta_model готова, попробуем с помощью нее спрогнозировать те же годовые, квартальные и месячные ряды из Sophisthse. Сравним результаты с предобученной моделью.

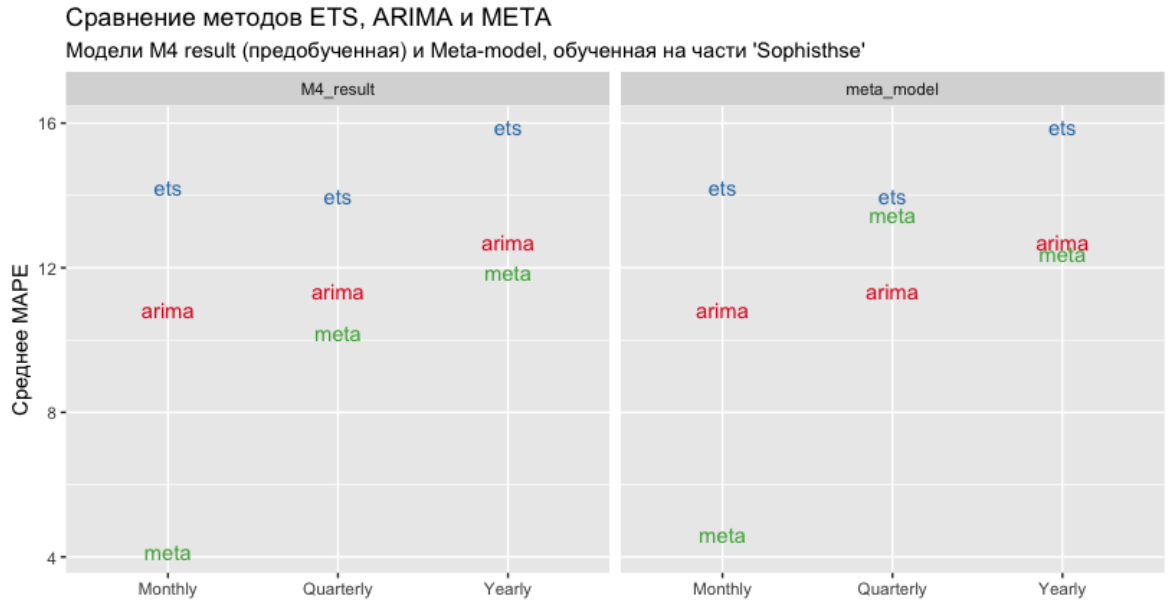


Рис. 16: Сравнение качества прогнозирования

В целом, качество мета алгоритма стало немного хуже по сравнению с моделью авторов, но, учитывая тот факт, что предобученная модель обучалась на 100000 рядах, а наша использовала меньше 100, то результат достаточно впечатляющий,

поскольку на месячных и годовых рядах мета по-прежнему обыгрывает ARIMA и ETS.

Здесь можно увидеть наиболее важные признаки рядов, по мнению нашей модели.

```
mat <- xgboost::xgb.importance (feature_names = colnames(train_data$data),
                                model = meta_model)
xgboost::xgb.plot.importance(importance_matrix = mat[1:20], cex=1.0)
```

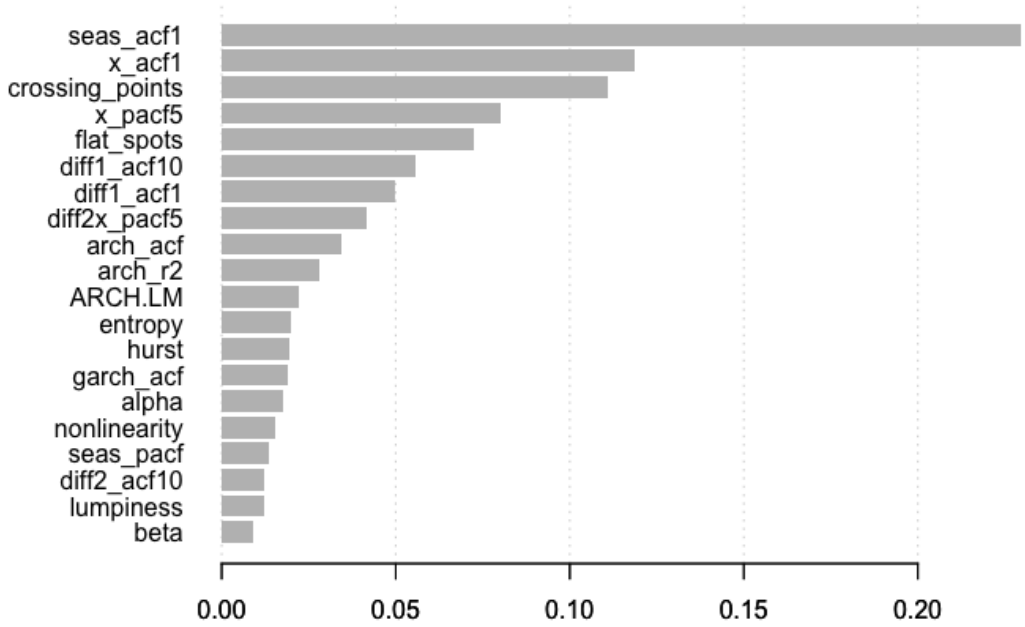


Рис. 17: Важность признаков

6 Итоги

Таким образом, разобравшись как работает модель и протестировав ее на разных наборах данных, можно сделать вывод, что новый метод прогнозирования Meta справляется лучше, чем автонастраиваемые Arima и ETS модели. Он требует больше времени на вычисление, но строит хорошие прогнозы.

Если есть возможность, рекомендуется обучать собственную модель применительно к данным, которые собираетесь прогнозировать. Так, если данных будет достаточно, можно добиться достаточно хорошего качества. Если возникают какие-то тружности, то предобученная модель тоже неплохая и неприхотлива в установке.

7 Приложение

7.1 Установка

Фрагменты кода и описание проблем, которые могут возникнуть, смотрите здесь.

Все манипуляции будем производить в RStudio. Для реализации кода необходимо установить и подключить следующие пакеты:

```
# Для новичков: если что-то не подключается, проверьте в Tools ~
# Install Packages... загружена ли библиотека

library("devtools")          # для загрузки пакетов
library("dplyr")             # инструменты для работы с массивами данных
library("forecast")          # базовые методы прогнозирования
library("ggplot2")           # визуализация
library("fpp2")              # временные ряды
library("xtable")            # для импорта таблиц
library("tseries")           # для анализа рядов
```

Теперь загрузим пакеты, подготовленные авторами исследуемого алгортима. Если вам для прогнозирования достаточно предобученной авторами модели, установите следующее:

```
devtools::install_github("pmontman/tsfeatures")
devtools::install_github("robjhyndman/M4metalearning")

# Обратите внимание, что пакет весит около 1Гб
install.packages("https://github.com/pmontman/M4metaresults/releases/download/
v0.0.0.9000/M4metaresults_0.0.0.9000.tar.gz", repos = NULL, type="source")

library("M4metalearning")    # исследуемый алгоритм
library("M4metaresults")     # предобученная модель
```

Если у вас имеется достаточно большое количество данных и вы хотите обучить на них исследуемую модель, необходимо установить специальную версию пакета xgboost (eXtreme Gradient Boosting Training), которая не встроена в RStudio.

```
devtools::install_github("pmontman/customxgboost")
```

Если у вас Macbook и возникла ошибка, как на картинке ниже, нужно будет сделать еще несколько манипуляций.

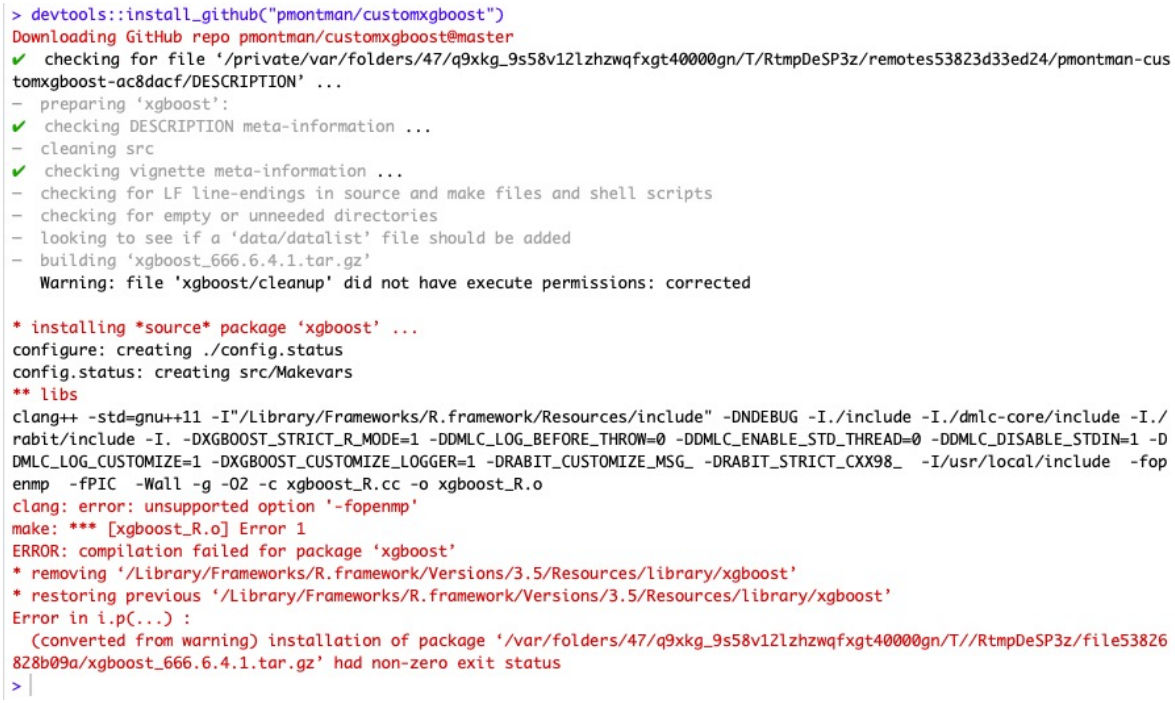


Рис. 18: Распространенная ошибка

Посмотрите в консоли RStudio какая у вас версия R и следуйте указаниям с сайта или воспользуйтесь советами пользователей, решивших данную проблему. Если хотите найти решение проблемы самостоятельно, обязательно учитывайте вашу версию RStudio.

Если вам удалось решить проблему с customxgboost, далее все должно работать нормально.

Также может возникнуть ошибка при скачивании данных M4comp2018. Пока неизвестно с чем это связано, следите за новостями.

```
devtools::install_github("carlanetto/M4comp2018")
```

Если у вас при использовании метода META возникает похожая ошибка (см.ниже), проверьте соотношение выборок train и test для ваших рядов. Удалите слишком короткие или увеличьте размер h. Все ряды нужно очищать от пропусков, например, с помощью na.remove или na.omit.

```
 Error in if (length(seriesentry$x) - seriesentry$h < max(2 * frq + 1,  :
argument is of length zero
```

Если неожиданно появляются проблемы с подключением к Sophisthse, которых раньше не было, проверьте подключение к интернету.

7.2 Реализация

7.2.1 Прогнозирование рядов различной частоты из M4

```
# Годовые данные
```

```r
monthly_M4 <- Filter(function(l) l$period == "Yearly", M4)

acc <- NULL
ts <- monthly_M4[1:20]
for (i in (1:10)) {
  forec_result <- forecast_meta_M4(model_M4, ts[[i]]$x, h = 12)
  temp <- accuracy(f = forec_result$mean, ts[[i]]$xx[1:12])
  acc <- rbind(acc, temp[5])
}

mean(acc)

acc1 <- NULL

for (i in (1:20)) {
  forec_result <- ets(ts[[i]]$x)
  forets_ets <- forecast(forec_result, h = 12)
  temp_ets <- accuracy(f = forets_ets$mean, ts[[i]]$xx[1:12])
  acc1 <- rbind(acc1, temp_ets[5])
}

mean(acc1)

acc2 <- NULL

for (i in (1:10)) {
  forec_result <- auto.arima(ts[[i]]$x)
  forets_ets <- forecast(forec_result, h = 12)
  temp_ets <- accuracy(f = forets_ets$mean, ts[[i]]$xx[1:12])
  acc2 <- rbind(acc_ets, temp_ets[5])
}

mean(acc2)

# Проверка на 50 месячных рядах

monthly_M4 <- Filter(function(l) l$period == "Monthly", M4)
ts <- monthly_M4[1:50]


acc_meta <- NULL
acc_ets <- NULL
acc_arima <- NULL

for (i in (1:50)) {
  meta_result <- forecast_meta_M4(model_M4, ts[[i]]$x, h = 12)
```

```
  meta <- accuracy(f = meta_result$mean, ts[[i]]$xx[1:12])
  acc_meta <- rbind(acc_meta, meta[5])

  ets_result <- ets(ts[[i]]$x)
  forets_ets <- forecast(ets_result, h = 12)
  ets <- accuracy(f = forets_ets$mean, ts[[i]]$xx[1:12])
  acc_ets <- rbind(acc_ets, ets[5])

  arima_result <- auto.arima(ts[[i]]$x)
  forets_arima <- forecast(arima_result, h = 12)
  tsrima <- accuracy(f = forets_arima$mean, ts[[i]]$xx[1:12])
  acc_arima <- rbind(acc_arima, tsrima[5])
}

mean(acc_meta)
mean(acc_ets)
mean(acc_arima)

# meta_month <- acc_meta
# ets_month <- acc_ets
# arima_month <- acc_arima


# Для квартальных данных

quarterly_M4 <- Filter(function(l) l$period == "Quarterly", M4)
ts <- quarterly_M4[1:50]


acc_meta <- NULL
acc_ets <- NULL
acc_arima <- NULL

for (i in (1:50)) {
  meta_result <- forecast_meta_M4(model_M4, ts[[i]]$x, h = 12)
  meta <- accuracy(f = meta_result$mean, ts[[i]]$xx[1:12])
  acc_meta <- rbind(acc_meta, meta[5])
}

for (i in (1:50)) {
  ets_result <- ets(ts[[i]]$x)
  forets_ets <- forecast(ets_result, h = 12)
  ets <- accuracy(f = forets_ets$mean, ts[[i]]$xx[1:12])
  acc_ets <- rbind(acc_ets, ets[5])

  arima_result <- auto.arima(ts[[i]]$x)
```

```r
  forets_arima <- forecast(arima_result, h = 12)
  tsrima <- accuracy(f = forets_arima$mean, ts[[i]]$xx[1:12])
  acc_arima <- rbind(acc_arima, tsrima[5])
}

mean(acc_meta)
mean(acc_ets)
mean(acc_arima)


# meta_quarter <- acc_meta
# ets_quarter <- acc_ets
# arima_quarter <- acc_arima

# Для годовых данных

yearly_M4 <- Filter(function(l) l$period == "Yearly", M4)
ts <- yearly_M4[1:50]


acc_meta <- NULL
acc_ets <- NULL
acc_arima <- NULL

for (i in (1:50)) {
  meta_result <- forecast_meta_M4(model_M4, ts[[i]]$x, h = 12)
  meta <- accuracy(f = meta_result$mean, ts[[i]]$xx[1:12])
  acc_meta <- rbind(acc_meta, meta[5])
}

for (i in (1:50)) {
  ets_result <- ets(ts[[i]]$x)
  forets_ets <- forecast(ets_result, h = 12)
  ets <- accuracy(f = forets_ets$mean, ts[[i]]$xx[1:12])
  acc_ets <- rbind(acc_ets, ets[5])

  arima_result <- auto.arima(ts[[i]]$x)
  forets_arima <- forecast(arima_result, h = 12)
  tsrima <- accuracy(f = forets_arima$mean, ts[[i]]$xx[1:12])
  acc_arima <- rbind(acc_arima, tsrima[5])
}

mean(acc_meta)
mean(acc_ets)
mean(acc_arima)
```

```
meta_yearly <- acc_meta
ets_yearly <- acc_ets
arima_yearly <- acc_arima


# Соединим полученные результаты в одной таблице

comparison <- cbind(
  Type = c("Monthly series", "Quarterly series", "Yearly series"),
  ETS = c(mean(ets_month), mean(ets_quarter), mean(ets_yearly)),
  ARIMA = c(mean(arima_month), mean(arima_quarter), mean(arima_yearly)),
  META = c(mean(meta_month), mean(meta_quarter), mean(meta_yearly))
)
```

7.2.2 Прогнозирование рядов различной частоты из Sophisthse

```
# ГОДОВЫЕ ДАННЫЕ
# Прогнозы для рядов из sophisthse с частотой 1 (годовые данные)

series <- filter(series_info, freq == 1)
series_1 <- series


# Создадим список названий рядов
label <- NA
for (i in c(1:length(series$table))) {
  label <- rbind(label, series$table[i])
}

# Удалим дубликаты
label <- unique(label)[-1]


# метод ETS

rus <- NA
rus1 <- NA
accuracy_ets <- NA


for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 2
```

```
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- ets(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_ets <- rbind(accuracy_ets, accuracy_result[5])
  }
}

# метод ARIMA

rus <- NA
accuracy_arima <- NA

for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 2
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- auto.arima(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_arima <- rbind(accuracy_arima, accuracy_result[5])
  }
}

sum(accuracy_arima)

# Комбинация META

rus <- NA
accuracy_meta <- NA


for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
```

```
    if (length(rus) > 4){
      h <- 2
      l <- length(rus) - h
      train <- ts(rus[1:l])
      test <- ts(rus[(l + 1):(l + h)])
      forecast_result <- forecast_meta_M4(model_M4, train, h = h)
      accuracy_result <- accuracy(f = forecast_result$mean, test)
      accuracy_meta <- rbind(accuracy_meta, accuracy_result[5])
    }

  }
}

c(i, j)
rus
length(rus)

# КВАРТАЛЬНЫЕ

series <- filter(series_info, freq == 4)
s_quart <- series

series$table[1]
length(series$table)

# Создадим список названий рядов
label <- NA
for (i in c(1:length(series$table))) {
  label <- rbind(label, series$table[i])
}

# Удалим дубликаты
label <- unique(label)[-1]


# метод ETS

rus <- NA
rus1 <- NA
accuracy_ets_4 <- NA


for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
```

```
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 4
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- ets(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_ets_4 <- rbind(accuracy_ets_4, accuracy_result[5])
  }
}

c(i, j)
# метод ARIMA

rus <- NA
accuracy_arima_4 <- NA

for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 4
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- auto.arima(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_arima_4 <- rbind(accuracy_arima_4, accuracy_result[5])
  }
}

sum(accuracy_arima)

# Комбинация

rus <- NA
accuracy_meta_4 <- NA


for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
```

```
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    if (length(rus) > 4){
      h <- 2
      l <- length(rus) - h
      train <- ts(rus[1:l])
      test <- ts(rus[(l + 1):(l + h)])
      forecast_result <- forecast_meta_M4(model_M4, train, h = h)
      accuracy_result <- accuracy(f = forecast_result$mean, test)
      accuracy_meta_4 <- rbind(accuracy_meta_4, accuracy_result[5])
    }

  }
}

c(i, j)
rus1



# МЕСЯЧНЫЕ
# Месячные данные

series <- filter(series_info, freq == 12)
series_12 <- series

series$table[1]
length(series$table)

# Создадим список названий рядов
label <- NA
for (i in c(1:length(series$table))) {
  label <- rbind(label, series$table[i])
}

# Удалим дубликаты
label <- unique(label)[-1]


# метод ETS

rus <- NA
rus1 <- NA
accuracy_ets_12 <- NA
```

```r
for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 12
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- ets(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_ets_12 <- rbind(accuracy_ets_12, accuracy_result[5])
  }
}

c(i, j)
# метод ARIMA

rus <- NA
accuracy_arima_12 <- NA

for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    h <- 12
    l <- length(rus) - h
    train <- rus[1:l]
    test <- rus[(l + 1):(l + h)]
    model <- auto.arima(train)
    forecast_result <- forecast(model, h = h)
    accuracy_result <- accuracy(f = forecast_result$mean, test)
    accuracy_arima_12 <- rbind(accuracy_arima_12, accuracy_result[5])
  }
}

sum(accuracy_arima)

# Комбинация
```

```
rus <- NA
accuracy_meta_12 <- NA


for (i in c(1:length(label))) {
  rus1 <- sophisthse(label[i])
  d <- dim(sophisthse(label[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    if (length(rus) > 4){
      h <- 2
      l <- length(rus) - h
      train <- ts(rus[1:l])
      test <- ts(rus[(l + 1):(l + h)])
      forecast_result <- forecast_meta_M4(model_M4, train, h = h)
      accuracy_result <- accuracy(f = forecast_result$mean, test)
      accuracy_meta_12 <- rbind(accuracy_meta_12, accuracy_result[5])
    }

  }
}
length(label)
c(i, j)
rus1



# Качество прогнозов годовых

accuracy_arima <- na.omit(accuracy_arima)
accuracy_arima <- accuracy_arima[is.finite(accuracy_arima)]
mean(accuracy_arima) # 12.69631

accuracy_ets <- na.omit(accuracy_ets)
accuracy_ets <- accuracy_ets[is.finite(accuracy_ets)]
mean(accuracy_ets) # 15.90514

accuracy_meta <- na.omit(accuracy_meta)
accuracy_meta <- accuracy_meta[is.finite(accuracy_meta)]
mean(accuracy_meta) # 11.85945

# Качество прогнозов квартальных

accuracy_arima_4 <- na.omit(accuracy_arima_4)
accuracy_arima_4 <- accuracy_arima_4[is.finite(accuracy_arima_4)]
```

```r
mean(accuracy_arima_4) # 11.35624

accuracy_ets_4 <- na.omit(accuracy_ets_4)
accuracy_ets_4 <- accuracy_ets_4[is.finite(accuracy_ets_4)]
mean(accuracy_ets_4) # 13.97002

accuracy_meta_4 <- na.omit(accuracy_meta_4)
accuracy_meta_4 <- accuracy_meta_4[is.finite(accuracy_meta_4)]
mean(accuracy_meta_4) # 10.19009

# Качество прогнозов месячных рядов

accuracy_arima_12 <- na.omit(accuracy_arima_12)
accuracy_arima_12 <- accuracy_arima_12[is.finite(accuracy_arima_12)]
mean(accuracy_arima_12) # 10.85314

accuracy_ets_12 <- na.omit(accuracy_ets_12)
accuracy_ets_12 <- accuracy_ets_12[is.finite(accuracy_ets_12)]
mean(accuracy_ets_12) # 14.21897

accuracy_meta_12 <- na.omit(accuracy_meta_12)
accuracy_meta_12 <- accuracy_meta_12[is.finite(accuracy_meta_12)]
mean(accuracy_meta_12) # 4.149181


# Соединим полученные результаты в одной таблице

comparison_soph <- cbind(
  Type = c("Monthly series", "Quarterly series", "Yearly series"),
  ETS = c(mean(accuracy_ets_12), mean(accuracy_ets_4), mean(accuracy_ets)),
  ARIMA = c(mean(accuracy_arima_12), mean(accuracy_arima_4), mean(accuracy_arima)
  META = c(mean(accuracy_meta_12), mean(accuracy_meta_4), mean(accuracy_meta))
)

# Стандартное отклонение

comparison_soph_sd <- cbind(
  Type = c("Monthly series", "Quarterly series", "Yearly series"),
  ETS = c(sd(accuracy_ets_12), sd(accuracy_ets_4), sd(accuracy_ets)),
  ARIMA = c(sd(accuracy_arima_12), sd(accuracy_arima_4), sd(accuracy_arima)),
  META = c(sd(accuracy_meta_12), sd(accuracy_meta_4), sd(accuracy_meta))
)
```

7.3 Обучение модели

```
# Создадим список названий рядов
label_meta <- NA
for (i in c(1:length(series$table))) {
  label_meta <- rbind(label_meta, series$table[i])
}

# Удалим дубликаты
label_meta <- unique(label_meta)[-1]

length(label_meta)
rus <- NA
data_soph <- list()
k <- 0

for (i in c(1:length(label_meta))) {
  rus1 <- sophisthse(label_meta[i])
  d <- dim(sophisthse(label_meta[i]))
  max <- d[2]
  for (j in c(1:max)) {
    rus <- na.remove(rus1[, j])
    if (length(rus) > 27) {
      k <- k + 1
      h = as.integer(6)
      ts <- ts(data = rus, frequency = 12)
      data_soph[[k]] <- list(x = ts, h = h)
    }
  }
}

meta_M4 <- temp_holdout(data_soph)


#calculate the forecasts of each method in the pool
#THIS WILL TAKE A LOT OF TIME (hours...)
meta_M4 <- calc_forecasts(meta_M4, forec_methods(), n.cores=2)

#calculate the OWA errors
meta_M4 <- calc_errors(meta_M4)
#extract the features
meta_M4 <- THA_features(meta_M4, n.cores=2)

#search for hyperparameters
hyperparameter_search(meta_M4, filename = "M4_hyper.RData", n_iter=50)

#get the best hyperparameter found
```

```
load("M4_hyper.RData")
best_hyper <- bay_results[ which.min(bay_results$combi_OWA), ]

#Train the metalearning model with the best hyperparameters found

train_data <- create_feat_classif_problem(meta_M4)
param <- list(max_depth=best_hyper$max_depth,
              eta=best_hyper$eta,
              nthread = 3,
              silent=1,
              objective=error_softmax_obj,
              num_class=ncol(train_data$errors),
              subsample=bay_results$subsample,
              colsample_bytree=bay_results$colsample_bytree)

meta_model <- train_selection_ensemble(train_data$data,
                                       train_data$errors,
                                       param=param)
```

8 Литература

1. Rob J Hyndman and George Athanasopoulos, Forecasting: Principles and Practice.
2. Pablo Montero-Manso, Forecasting Metalearning Example
3. Pablo Montero-Manso, Reproducibility: Combination of Forecast Methods by Feature-based Learning.
4. Makridakis, Spyros and Spiliotis, Evangelos and Assimakopoulos, Vassilis, The M4 Competition: Results, findings, conclusion and way forward, International Journal of Forecasting.
5. Rob J Hyndman, Baki Billah, Unmasking the Theta method.